Verificação de Requisitos de Transparência em Modelos iStar

Joás Weslei Baía, José Luis Braga, Leonardo Fonseca de Carvalho

Universidade Federal de Viçosa
Campus UFV, 36570-000 Viçosa, MG, Brasil.
joas.baia@gmail.com; zeluis@dpi.ufv.br; leofdecarvalho@gmail.com

Resumo. A transparência de software é um requisito não funcional que os engenheiros de software precisarão demonstrar à medida que a sociedade exigir transparência nas relações com seus representantes, pois essas relações são automatizadas por software. Nesse sentido a verificação da presença de transparência em modelos de requisitos de software é a abordagem utilizada nesse trabalho. Esses modelos representados com o *framework* iStar são verificados utilizando a linguagem CLIPS através da aplicação de regras de produção implementadas a partir das características do iStar que possibilitam identificar os requisitos de transparência. O modelo de requisito é inicialmente representado no formato iStarML e transformado em fatos formando assim a base de dados do sistema de verificação. Essa base de dados é submetida juntamente com a base de conhecimento ao motor de inferência CLIPS para extrair da especificação de requisitos o conhecimento implícito sobre os requisitos de transparência.

Palavras-chave: Framework iStar, transparência de software, verificação de modelos de requisitos.

1 Introdução

Nas sociedades democráticas é crescente a demanda por informação de qualidade, consistente e 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. Atualmente nas democracias é crescente a preocupação com a transparência dos atos do poder público, e esforços estão sendo feitos no sentido de evitar abusos dos governantes, sem que a informação fique disponível. Obviamente nos regimes de governo autoritários os princípios das sociedades abertas não são bem aceitos [1].

A sociedade carece de sistemas computacionais de qualidade em todos os meios de produção. Modelos de negócio são automatizados por sistemas computacionais, e a necessidade por transparência no *software* surge devido à necessidade de

transparência nas organizações. Essa pressão social se transfere para as organizações, que precisarão demonstrar transparência em sua estrutura organizacional [2].

A transparência de *software* é um requisito não funcional [2] que engenheiros de *software* terão que atender e incluir nos projetos de *software*, 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.

Nesse trabalho a abordagem utilizada para verificar se o *software* atende a requisitos de transparência é baseada na verificação de modelos de requisitos de *software* para identificar a presença de atributos de transparência. O modelo de requisitos de *software* representado com o framework iStar [5] é inspecionado em busca da ocorrência de requisitos de transparência. A versão da linguagem iStar utilizada nesse trabalho é a iStar Wiki [17].

O sistema de verificação proposto nesse trabalho é composto por dois módulos: a base de conhecimento e a base de dados (base de fatos). A base de conhecimento representada pelas regras de produção é implementada a partir das características do iStar que possibilitam identificar os requisitos de transparência. A base de dados é composta pela especificação de requisitos iStar transformada em fatos da linguagem CLIPS [11].

Este texto está organizado em seis seções. Na segunda seção são apresentados os requisitos de transparência juntamente com as características do framework iStar que permitem identificá-los. A terceira seção discute a abordagem escolhida para a verificação da presença de requisitos de transparência a partir de modelos iStar. A quarta seção mostra a inferência de requisitos de transparência a partir de especificações iStar. Na quinta seção discutimos os trabalhos correlacionados à nossa abordagem. Por fim, na sexta seção apresentamos as conclusões.

2 Características do framework iStar que permitem identificar requisitos de transparência.

A transparência de *software* tem sido explorada como um requisito não funcional [2], [3], [4]. A tabela 1 mostra as características do framework istar [5] que permitem identificar os requisitos de transparência [4].

Os requisitos de transparência são agrupados em: acessível, usável, informativo, entendível e auditável. Esse agrupamento surgiu porque os atributos de transparência estão relacionados entre si, e eles são considerados graus de transparência que contribuem para alcançar a transparência [3].

A tabela 1 mostra os requisitos de transparência agrupados segundo suas correlações. Além disso ela mostra o relacionamento entre esses requisitos e as características do framework iStar que possibilitam identificá-los. Essas características são usadas como conhecimento na implementação de regras de produção para verificar se especificações de requisitos incorporam os requisitos de transparência.

A coluna Relação da tabela 1 expressa a aderência das características do iStar identificados pelos algarismos de 1 a 8 na coluna ID, aos requisitos de transparência.

Verificabilidade, por exemplo, é relacionada com as características 1, 2, 3, e 4 do iStar.

A engenharia de requisitos baseada em objetivos surgiu como uma proposta para análise de requisitos com base nas intenções dos atores 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 [6]. A intencionalidade contribui para aumentar a transparência pois através dela é possível obter rastreabilidade, verificabilidade e auditabilidade das ações realizadas pelos atores para atingir seus objetivos no ambiente organizacional [4].

Table 1. Características que identificam requisitos de transparência. Adaptado de [3] [4] e [7].

Graus de transparência	Requisitos de transparência	Relação	ID	Características iStar
Auditabilidade	Verificabilidade	1, 2, 3, 4	1	Atores, metas, tarefas e recursos
	Controlabilidade	2	2	Alternativas de operacionalização
	Rastreabilidade	1, 2, 5, 8	3	Estrutura organizacional
	Validação	2	4	Dependência estratégica
	Intencionalidade detalhada	6	5	Contribuições +/-
Entendibilidade	Compositividade	1, 4	6	Decomposição de tarefas
	Divisibilidade	3, 6	7	Meta flexível
	Extensibilidade	2	8	Intencionalidade
	Dependência	4		
	Clareza	1, 4, 5, 7		
Informatividade	Integridade	2		
	Completude	7		
	Acurácia	7		
	Divulgação	3		
Acessibilidade	Disponibilidade	3		
	Desempenho	3		
	Operabilidade	2		

O iStar é um framework de modelagem de requisitos intencional orientado a agentes [5] que representa elementos intencionais tais como: metas, metas flexíveis, tarefas e recursos. Esses elementos são organizados em torno de atores estratégicos, que por sua vez podem ser especializados em agentes, papéis e posições. Além disso, ele fornece dois modelos de requisitos que utilizam esses elementos: o modelo de razões estratégicas e o modelo de dependência estratégica. Esse representa o relacionamento de dependência entre os vários atores, enquanto aquele descreve as intenções dos atores estratégicos [8]. Adicionalmente o modelo de atores estratégicos descreve os conceitos de agente, papéis e posições [9].

O modelo de razões estratégicas do iStar fornece as características: 1, 2, 5, 6, 7 e 8, conforme relacionado na tabela 1, enquanto o modelo de dependência estratégica provê a característica dependência estratégica [4]. Além disso, o modelo de atores estratégicos representa a estrutura organizacional e a taxonomia de atores [7].

O modelo de razões estratégicas apresenta quatro características que permitem identificar requisitos de transparência: atores intencionais, metas flexíveis explícitas, intencionalidade detalhada e alternativas de operacionalização. A intencionalidade dos atores contribui para os requisitos de *verificabilidade* e *rastreabilidade*. As metas flexíveis (*softgoals*), relacionam os requisitos de *completeza*, *clareza* e *acurácia*. A *intencionalidade detalhada* descrita pela característica decomposição de tarefas identifica os requisitos de *compositividade* e *decomposição*. Por fim a representação das alternativas de operacionalização permitem relacionar os requisitos de *integridade*, *extensibilidade* e *validade* [4].

O modelo de dependência estratégica representa os atores envolvidos no processo bem como seus relacionamentos. Esse modelo permite identificar os atores e seus objetivos, as metas flexíveis e recursos necessários para a execução de tarefas através das dependências de meta, meta flexível, tarefa e recurso, respectivamente [7].

O modelo de atores estratégicos permite identificar o requisito de transparência *verificabilidade* através da descrição da estrutura organizacional. Além disso ele representa a classificação de atores que possibilita identificar *disponibilidade*, *desempenho* e *divisibilidade* [7].

Os requisitos que formam o grau de transparência *usabilidade* [3] não estão relacionados na tabela 1 pois eles não têm relação direta com as características providas pelo framework iStar [4], [7].

3 Verificação de requisitos de transparência em modelos iStar.

A abordagem utilizada neste trabalho para verificar a presença de requisitos de transparência em modelos iStar, consiste em criar regras e fatos utilizando a linguagem CLIPS - *C Language Integrated Production System* [11]. As regras de produção atuam sobre os fatos para inferir o conhecimento sobre requisitos de transparência. Esse conhecimento capturado pelo framework iStar é implementado nas regras de produção. O sistema de verificação proposto nesse trabalho é composto por dois módulos: a base de conhecimento e a dase de dados (base de fatos). A base de conhecimento representada pelas regras de produção é implementada a partir das características iStar que possibilitam identificar os requisitos de transparência. A base de dados é composta pela especificação de requisitos iStar transformada em fatos da linguagem CLIPS. A separação entre a base de dados e a base de conhecimento facilita a expansão do sistema [16]. Novas regras de produção podem ser adicionadas à base de conhecimento à medida que o sistema evolui.

A figura 1 mostra um modelo de razões estratégicas do iStar no domínio de empréstimos bancários, representando a contratação de um cliente para obter financiamento. O objetivo principal do cliente é obter um financiamento, ele faz a proposta de crédito e comprova sua renda. Adicionalmente ele deseja obter o

financiamento adequado ao cenário econômico, onde são analisados as melhores taxas de juros e prazos. A instituição financeira tem como objetivo principal obter lucro em suas operações finaceiras. Para alcançar esse objetivo ela deseja evitar a inadimplência e monitora possíveis fraudes nessas transações. Nesse sentido, ela realiza duas tarefas: analisa a renda do proponente e verifica se o mesmo não consta em cadastros de inadimplentes. Ela utiliza o recurso lista de inadimplentes na execução dessa tarefa.

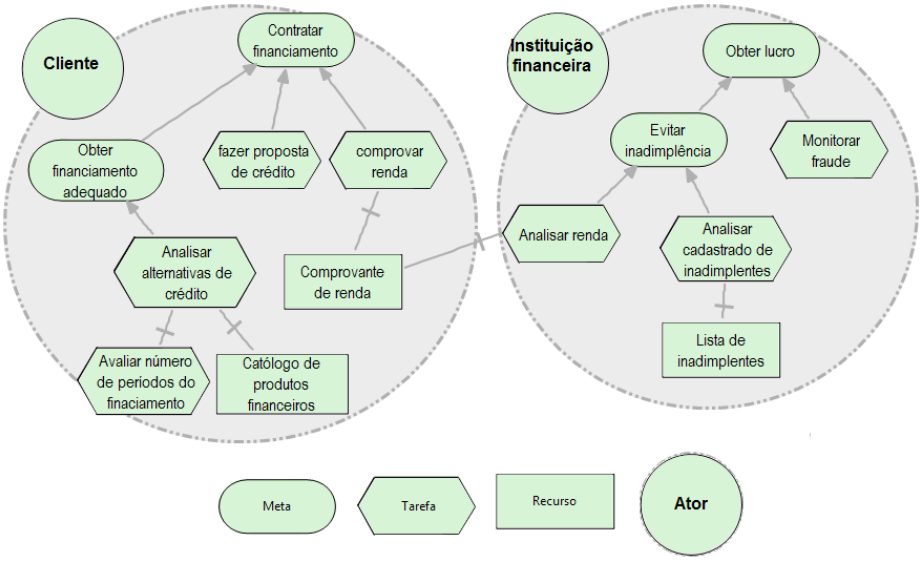

Fig. 1. Modelo de razão estratégica do domínio empréstimo bancário

3.1 Mapeando um modelo iStar em iStarML e CLIPS

A figura 2 mostra as etapas realizadas para verificar a presença de requisitos de transparência em especificações de requisitos. Inicialmente os requisitos são representados com o framework iStar, conforme exemplo na figura 1, em seguida eles são representados no formato iStarML [10], um formato de intercâmbio de modelos iStar baseado em XML (*Extensible Markup Language*). O iStarML é um formato para descrever diagramas iStar proposto para permitir o compartilhamento de especificações de requisitos entre várias ferramentas de modelagem iStar [12] disponíveis.

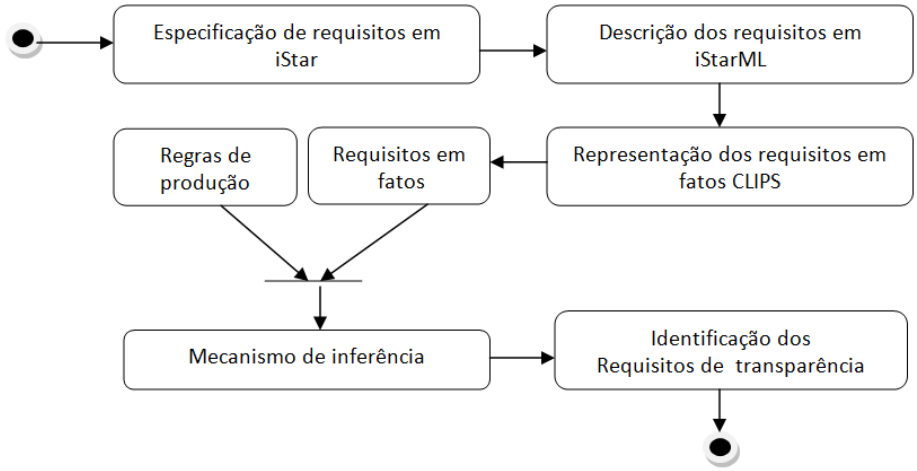

Fig. 2. Sequência de tarefas para verificar requisitos de transparência em especificações iStar.

A figura 3 ilustra parcialmente o modelo de requisitos da figura 1 representado em iStarML. O diagrama está aninhado dentro da tag <istarml/> conforme as linhas 2 a 15, semelhantemente os elementos intencionais do iStar são aninhados pela respectiva tag <actor/>, indicando assim que cada ator no modelo tem uma fronteira, linhas 5 à 11, e dentro dessa há os elementos intencionais do framework iStar [10].

```
1  <?xml version="1.0"?>
2  <istarml version="1.0">
3      <diagram name="Análise de crédito">
4          <actor id="01" name="Cliente">
5              <boundary>
6                  <ielement id="02" type="goal"
7                  name="Contratar financiamento">
8                      <ielementLink
9                      type="means-end" iref="03"/>
10                 </ielement>
11             </boundary>
12         </actor>
13         <actor id="15" name="Instituição financeira">
14     </diagram>
15 </istarml>
```

Fig. 3. Parte do modelo de requisitos da figura 1 representado no formato iStarML.

A partir de um modelo de requisitos no formato iStarML, é realizado sua transformação para a sintaxe da linguagem CLIPS, que é uma linguagem desenvolvida para implementação de sistemas especialistas com base de conhecimento e motor de inferência para realização de deduções e extração de conhecimento implícito. Ela funciona sobre três pilares: regras, fatos e inferência. O motor de inferência atua sobre o conjunto de fatos através da aplicação das regras de produção para extrair o conhecimento sobre o domínio em questão [11], portanto é necessário que o modelo de requisitos seja representado em fatos e o conhencimento

utilizado para verificar os requisitos de transparência implementado nas regras de produção para ocorrer a inferência do conhecimento implícito na especificação de requisitos. A transformação para o formato de fatos CLIPS ocorre conforme o mapeamento descrito na tabela 2. A implementação das regras de produção utiliza as informações disponíveis na tabela 1. Conforme ela mostra, o framework iStar possui características que permitem identificar os requisitos de transparência [4] [7].

Table 2. Mapeamento 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/>	-

A figura 4 ilustra a definição de fatos em CLIPS utilizando o mapeamento da tabela 2. As linhas 277 à 282 mostram a implementação da tag <ielement/> em CLIPS. Na linhas 283 e 284 está implementado um exemplo do mapeamento da tag <ielementLink/>.

```
276 (deffacts requirements "Requisitos do modelo iStar"
277  (element (id "01") (name "Instituição financeira")
278           (type "actor")                              )
279  (element (id "02") (name "Cliente")
280           (type "actor")                              )
281  (element (id "03") (name "Evitar inadimplencia")
282           (type "goal") (idActor "01")                )
283  (elementLink (type "means-end") (source "03")
284           (target "08")                               )
```

Fig. 4. Definição de fatos em CLIPS

A figura 5 mostra a implementação da regra de produção que verifica se o modelo possui o requisito de transparência *intencionalidade*. Uma regra de produção é composta por duas partes: antecedente e consequente [11]. Na parte antecedente estão as condições necessárias para que a regra seja disparada, linhas 98 à 100, e na parte consequente, linhas 102 a 109, as ações que serão realizadas quando ocorrer seu disparo. Porém, o disparo de uma regra ocorre somente quando as condições da parte antecedente forem satisfeitas, ou seja, há fatos na base de dados que satisfazem às restrições da regra. Nesse sentido a *regra de intencionalidade* da figura 5 mostra quais são as metas dos respectivos atores do modelo de requisitos iStar.
A submissão ao motor de inferência CLIPS dos fatos do modelo de requisitos, figura 4, juntamente com a regra de intencionalidade, figura 5, permitiu a identificação das seguintes intencionalidades:

- Cliente deseja obter financiamento adequado.

- Cliente deseja contratar financiamento.
- Instituição financeira deseja obter lucro.
- Instituição financeira deseja evitar inadimplência.

```
(defrule ruleIntencionalidade "Intencionalidade"
    (element (id ?idGoal)   (name ?nameGoal)
             (type "goal") (idActor ?idActor ) )
    (element (id ?idActor) (name ?nameActor)   )
    =>
    (assert (intencionalidade
                (idActor ?idActor)
                (actor   ?nameActor)
                (idGoal  ?idGoal)
                (goal    ?nameGoal)
            )
    )
)
```

Fig. 5. Implementação da *regra de intencionalidade.*

A modelagem das alternativas de operacionalização permite identificar os requisitos de transparência *controlabilidade*, *validação*, *extensibilidade*, *integração* e *operabilidade*, conforme relaciona a coluna Relação da tabela 1.

```
(defrule ruleAlternativaDeOperacionalizacao
    ( element     (id ?idGoal)       (name ?nameGoal)
                  (type "goal")      (idActor ?idActor ) )
    ( element     (id ?idElement)    (name ?nameElement)
                  (type ?typeElement)(idActor ?idActor ) )
    ( elementLink (type "means-end") (target ?idGoal)
                  (source ?idElement)                   )
    =>
    ( assert (alternativaDeOperacionalizacao
                (idGoal         ?idGoal)
                (nameGoal       ?nameGoal)
                (idElement      ?idElement)
                (nameElement    ?nameElement)
                (typeElement    ?typeElement)
                (idActorElement ?idActor )
            )
    )
)
```

Fig. 6. Regra de alternativas de operacionalização.

A regra *alternativa de operacionalização* implementa as informações sobre as alternativas de operacinalização conforme mostra a figura 6. Ela avalia três condições na parte antecedente: nas linha 126 à 131, o mecanismo de inferência testa se há elementos intencionais do tipo *goal*. A segunda condição seleciona todos os elementos do modelo que estão na fronteira de um mesmo ator. Nesse sentido ela utiliza o atributo *idActor* com a variável *?idActor*. Por fim a terceira condição verifica se há um *link* entre um elemento do tipo *goal* e os demais elementos. Caso afirmativo a regra é disparada e o fato *alternativaDeOperacionalização* é criado com os atributos do elemento do tipo *goal* e de sua *alternativa de operacionalizaçã*o.

Como resultado da aplicação da regra *alternativaDeOpercionalizacao* ao modelo de razão estratégica da figura 1 representado em fatos, resulta nas seguintes *alternativas de operacionalização*:

- *Goal*: Obter financiamento adequado *Task*: Analisar alternativas de crédito.
- *Goal*: Contratar financiamento. *Task*: Comprovar renda.
- *Goal*: Contratar financiamento. *Task*: Fazer proposta de crédito.
- *Goal*: Contratar financiamento. *Goal*: Obter financiamento adequado.
- *Goal*: Evitar inadimplência. *Task*: Analisar cadastro de inadimplentes.
- *Goal*: Evitar inadimplência. *Task*: Analisar renda.
- *Goal*: Obter lucro. *Task*: Monitorar fraude.
- *Goal*: Obter lucro. *Goal*: Evitar inadimplência.

A implementação das demais regras para verificar a ocorrência de requisitos de transparência em modelos iStar seguiu o mesmo processo usado para definir as regras de verificação de intencionalidade e alternativas de operacionalização, ou seja, as características do iStar que fornecem transparência ao modelo de requisitos, conforme tabela 1, são usadas na parte antecedente de cada regra. A tabela 3 mostra a relação entre as regras implementadas e as características que foram utilizadas em sua implementação.

A *regra verificabilidade* analisa se no modelo há o requisito de transparência *verificabilidade*, avaliando se os atores possuem intencionalidade, quais os meios pelos quais eles alcançam seus objetivos, a estrutura organizacional em questão e as dependências entre os mesmos.

A *regra de dependência estratégica* verifica as dependências entre os atores. Ela considera os quatro tipos de dependência modelados com o iStar: dependência entre tarefas, metas, recursos e metas flexíveis.

Table 3. Regras de produção x características iStar

Regras de produção	Características iStar
ruleVerificabilidade	Atores, metas, tarefas e recursos
ruleAlternativaOpercionalizacao	Alternativas de operacionalização
ruleEstruturaOrganizacional ruleTaxonomiaAgentes	Estrutura organizacional
ruleDependenciaEstrategica	Dependência estratégica
ruleContribuicao	Contribuições +/-
ruleDecomposicao	Decomposição de tarefas
ruleIntencionalidade	Intencionalidade

A regra que inspeciona a estrutura organizacional analisa os elementos do modelo de atores estratégicos: agentes, papéis e posições. Semelhantemente a regra de taxonomia de agentes atua sobre a base de fatos a procura de *links* "*is_a*", que relaciona dois atores com base no conceito de especialização-generalização. Dessa forma a regra extrai a classificação de atores presente no modelo.

A regra contribuição atua sobre os *links* do tipo "*contribution*" identificando as contribuições positivas e negativas entre os elementos intencionais do framework iStar.

Finalmente a regra decomposição atua sobre os *links* "*decomposition*" relacionando os elementos que decompõe uma tarefa.

A partir da identificação dos requisitos de transparência é possível verificar o grau de transparência [3] que o modelo de requisitos captura através da implementação de regras de produção que atuam sobre o conhecimento produzido pelas regras da tabela 3. A figura 08 mostra a regra que identifica o grau de transparência auditabilidade.

```
234 (defrule ruleAuditabilidade "auditabilidade"
235     ( verificabilidade )
236     ( contribuicao     )
237     ( decomposicao     )
238     ( intencionalidade )
239     =>
240     (printout t " >>> Grau de transparência:
241         auditabilidade " crlf)
242 )
```

Fig. 7. Regra para verificar o grau de transparência auditabilidade.

Na parte antecedente da *regra auditabilidade*, linhas 235 a 235 da figura 7, são verificadas a presença dos requisitos de *verificabilidade*, *contribuições positivas e negativas*, *decomposição de tarefas* e *intencionalidade*. Uma vez satisfeitas essas condições a regra é disparada sinalizando que o *grau de transparência auditabilidade* foi alcançado. A informação utilizada para implementar essa regra tem origem na primeira e segunda colunas da tabela 1, juntamente com a informação produzida ao aplicar as regras da tabela 3. Portanto as regras implementadas para verificar os graus de transparência foram: *Auditabilidade*, *Entendibilidade*, *Informatividade* e *Acessibilidade*.

A verificação de um modelo de requisitos representado com o framework iStar ocorre a partir de sua transcrição para o formato iStarML, e então é realizado o mapeamento entre esse formato e a representação apresentada na linguagem CLIPS. Finalmente esses fatos são submetidos junto com as regras de produção ao mecanismo de inferência dessa linguagem.

4 Inferindo requisitos de transparência a partir de modelo iStar com CLIPS

No modelo de razões estratégicas da figura 1, na fronteira da instituição financeira, a tarefa monitorar fraude não foi decomposta em subtarefas ou recursos. A decomposição de tarefas é uma característica do framework iStar que permite identificar o requisito de transparência intencionalidade detalhada [4], esse requisito não foi demonstrado totalmente no modelo da figura 1.

A abordagem para tratar esse problema consiste em criar regras de produção implementadas na linguagem CLIPS que atuam sobre os fatos para identificar pontos onde os requisitos de transparência são necessários. A regra de sugestão de decomposição da figura 8 identifica quais são as tarefas que não sofreram decomposição.

```
225 (defrule ruleSugereDecomposicao "Decomposição"
226     (element (id ?idTask)  (name ?nameTask)
227              (type "task")                  )
228     (not (decomposicao
229              (idTask ?idTask)
230              (nameTask ?nameTask)         ) )
231     =>
232     (printout t " >>> A tarefa : "?nameTask
233              " Não foi decomposta " crlf)
234 )
```

Fig. 8. Regra para verificar a falta de decomposição.

O funcionamento da regra *ruleSugereDecomposicao* consiste em verificar a ausência de decomposição para cada tarefa. Na parte antecedente são analisados dois fatos: os elementos do tipo tarefa e as decomposições existentes. Quando a regra de decomposição é disparada ela cria o fato *decomposicao* contendo o *id* e *nome* da tarefa e do elemento relacionado à decomposição. Por exemplo, a tarefa analisar cadastro de inadimplentes é decomposta no recurso lista de inadimplentes. Quando a regra de decomposição analisa o fato contendo essa tarefa ela identifica a decomposição da mesma através do *link* de decomposição ligando o elemento lista de inadimplentes. A partir dos fatos de decomposição a regra *ruleSugereDecomposicao* identifica a falta de decomposição pela interseção entre o conjunto de tarefas e o conjunto de tarefas decompostas. Quando ocorre o disparo dessa regra o sistema de inferência informa ao usuário quais as tarefas que não sofreram decomposição, linhas 232 e 233 da figura 8.

A aplicação da regra que sugere decomposição de tarefas ao modelo de requisitos da figura 1 representado em fatos constatou que as tarefas analisar renda, monitorar fraude, fazer proposta de crédito, comprovar renda e avaliar número de períodos do financiamento não foram decompostas. Consequentemente os requisitos de

intencionalidade detalhada e *divisibilidade*, não foram demonstrados totalmente nesse modelo de requisitos, conforme relaciona a tabela 1.

A identificação da ausência dos requisitos de transparência em especificações iStar permite que o usuário seja informado em quais partes do modelo precisam reunir novas informações sobre o ambiente organizacional para que essas especificações represente os requisitos de transparência. A base de conhecimento implementada em regras de produção extraiu do modelo de requisitos da figura 01 os requisitos de transparência nele presente, além disso, ela identificou quais requisitos de transparência estão ausentes nesse modelo.

O sistema de verificação de requisitos proposto limita-se a identificar requisitos de transparência, outros aspectos de verificação de requisitos tais como ambiguidade, completude e rastreabilidade não estão implementados na base de conhecimento.

O sistema de verificação de requisitos implementado nesse trabalho poderá ser utilizado juntamente com as ferramentas de suporte à modelagem de requisitos iStar disponíveis [12], principalmente as ferramentas que fazem exportação automática no formata iStarML.

5 Trabalhos correlacionados

Nessa seção apresentamos três abordagens para verificação de requisitos correlacionada ao nosso trabalho. A primeira abordagem utiliza a tecnologia XML/XSLT para verifcar especificações de requisitos sob os aspectos de ambiguidade, completude e rastreabilidade [13]. A verificação dos requisitos é realizada através da aplicação de folhas de estilos sobre o modelo de requisitos representado em XML. Em nossa abordagem utilizamos a tecnologia XML somente para representar o modelo de requisitos e obter interoperabilidade entre ferramentas iStar. Para executar a verificação dos requisitos utilizamos regras de produção que atuam sobre o modelo de requisitos representado em fatos da linguagem CLIPS.

A segunda abordagem utiliza técnicas formais de análise de requisitos [14]. A linguagem Tropos é empregada para verificar os aspectos dinâmicos de dependência entre atores estratégicos. O modelo de especificação de requisitos iStar é representado em Tropos onde atores, metas, tarefas, recursos e dependências são declarados juntamente com as restrições usadas para verificar o modelo. Nossa abordagem difere dessa 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. Temos dois módulos distintos para realizar a verificação: a base de fatos e a base de regras.

A terceira abordagem de verificação de especificações de requisitos é baseada no processamento de linguagem natural [15], onde o objetivo principal é identificar ambiguidades em especificações de requisitos. Inicialmente a especificação descrita em linguagem natural é submetida ao procedimento de medição de ambiguidades, onde as sentenças potencialmente ambíguas são identificadas. Posteriormente é identificado o motivo que tornaria a sentença ambígua auxiliando o trabalho engenheiro de requisitos na remoção de tais ambiguidades. Nossa abordagem também

difere dessa na forma como os requisitos são representados, pois descrevemos os requisitos com o framework iStar e posteriormente representamos o conhecimento que esse modelo captura na forma de fatos na linguagem CLIPS.

A principal diferença entre a abordagem utilizada nesse trabalho e as outras três consiste no objetivo da verificação. Estamos interessados na identificação de requisitos 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 Stekeholders à medida que ela for exigida pelos diversos segmentos da sociedade.

6 Conclusões

A abordagem utilizada nesse trabalho para identificar requisitos de transparência em modelos iStar utiliza o formato de intercâmbio de modelos iStarML juntamente com a linguagem CLIPS que é usada para implementar as regras de produção e para representar o modelo de requisitos em fatos.

A utilização do formato iStarML provê uma interface de comunicação entre as ferramentas de modelagem de requisitos iStar, tornando essa abordagem independente de plataforma. O mapeamento entre as tags iStarML e fatos da linguagem CLIPS possibilita que os dados sobre o modelo de requisitos sejam submetidos ao motor de inferência desta linguagem.

As características iStar que permitem identificar requisitos de transparência são implementadas através das regras de produção. Consequentemente a verificação de modelos de requisitos iStar torna-se possível através da representação desse modelo em fatos e posteriormente pela submissão desses fatos ao mecanismo de inferência da linguagem CLIPS.

O processo de verificação de requisitos de transparência em modelos de requisitos iStar ocorre em duas etapas, na primeira verifica-se quais os requisitos de transparência estão presentes nesse modelo e a segunda são identificados partes do modelo iStar onde os requisitos de transparência são necessários para tornar o software mais aderente à transparência.

A abordagem baseada em regras de produção em um sistema de inferência como o CLIPS proporcionou a obtenção de um sistema flexível e expansível, que permite a utilização tanto para verificar especificações de transparência quanto para sugerir a inclusão de novos requisitos em especificações analisadas pelo sistema. Essa abordagem 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.

Agradecemos à Capes, CNPq, Fapemig, Funarbe e Sydle pelo apoio recebido neste projeto.

Referências Bibliográficas

1. Holzner B., Holzner L., Transparency in Global Change: The Value of Open society. Universidad of Pittsburgh Press; 1 edition, 206.
2. Cappelli, C., Pádua, A., Leite, J.C.S.P, Exploring Business Process Transparency Concepts, RE 207, IEEE Computer Society Press, pp. 389-390, 207.
3. CAPPELLI, C., LEITE, J. C. S. P. Transparência de Processos Organizacionais. Universidade Federal Fluminense, LATEC. II Simpósio Internacional de Transparência nos Negócios. 208.
4. Leite, J.C.S.P, Cappelli, C., Exploring i* Characteristics that Support Software Transparency. In Proceedings of the 3rd International i* Workshop, CEUR Workshop Proceedings, Vol. 322, 208, pp. 51-54. <Disponível em: http://CEUR-WS.org/Vol-322/> <Acesso em 24/09/211>.
5. E. Yu. Modelling Strategic Relationships for Process Reengineering Ph.D. Thesis. Dept. of Computer Science, University of Toronto. 1995.
6. Lapouchnian, A. Goal-Oriented Requirements Engineering: An Overview of the Current Research. Department of Computer Science University Of Toronto. 205.
7. Oliveira, A. P. A. ; Cappelli, C. ; Cunha, H. S. ; Leite, J. C. P. ; WERNECK, Vera M. B. . Engenharia de software Intencional: Tornando o Software Mais Transparente. In: XXI Simpósio Brasileiro de Engenharia de Software, 207, Rio de Janeiro. XXI Simpósio Brasileiro de Engenharia de Software, 207.
8. Eric Yu, Markus Strohmaier, Xiaoxue Deng, Exploring Intentional Modeling and Analysis for Enterprise Architecture, Proceedings of the 10th IEEE on International Enterprise Distributed Object Computing Conference Workshops, p.32, October 16-20, 206.
9. Leite, J. C. S. P. ; Werneck, V. ; Padua, A. ; Cappelli, C. ; Cerqueira, Ana ; Cunha, H.S. ; González-Baixauli, B. Understanding the Strategic Actor Diagram: An Exercise of Meta Modeling. In:. Proceedings of the 10th Workshop on Requirements Engineering. Toronto : York University, 207. v. 10. p. 2-12 (wer.inf.puc-rio.br/WERpapers).
10. Cares, C., Franch, X., Perini, A., Susi, A., iStarML: an XML-based Interchange Format for i* Models. Proceedings of the 3rd International i* Workshop, Recife, Brazil, February 11-12, 208.
11. CLIPS Version 5.1 User's Guide, NASA Lyndon B. Johnson Space Center, Software Technology Branch, Houston, TX, 1991.
12. iStar tools. Disponível em <http://istar.rwth-aachen.de/tiki-index.php?page=i*%20Tools>. Acesso em 13 de novembro de 211.
13. A. Duran, B. Bernardez, A. Ruiz, and T. Toro, "An XML-based approach for the automatic verification of software requirements specifications," in Proc. of the Fourth Workshop on Requirements Engineering (WER01), 201, pp. 181-194.
14. Fuxman, A.; Mylopoulos, J.; Pistore, M.; Traverso, P. Model Checking Early Requirements Specifications in Tropos. RE-2001, the 9th IEEE International Requirements Engineering Conference. Toronto, Canada, (201).
15. Requirements for Tools for Ambiguity Identification and Measurement in Natural Language Requirements Specifications. *Nadzeya Kiyavitskaya, Nicola Zeni, Luisa Mich, Daniel M. Berry.* Anais do WER07 - Workshop em Engenharia de Requisitos, Toronto, Canada, May 17-18, 207, pp 197 – 206.
16. NEGNEVITSKY, M. , Artificial Intelligence: A guide to Intelligent Systems. Addison Wesley; 2nd ed., 205, pp. 50.
17. Grau, G.: iStar Guide. Disponível em <http://istar.rwth-aachen.de/tiki-view_articles.php>. Acesso em 03 de março de 212.