Pontifícia Universidade Católica
do Rio de Janeiro

André Luiz de Castro Leal

Análise de Conformidade de Software com Base em Catálogos de Requisitos não Funcionais: Uma Abordagem Baseada em Sistemas Multi-Agentes

Tese de Doutorado

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio.

Orientador: Prof. Julio Cesar Sampaio do Prado Leite

Rio de Janeiro
Abril de 2014

Pontifícia Universidade Católica
do Rio de Janeiro

André Luiz de Castro Leal

Análise de Conformidade de Software com Base em Catálogos de Requisitos não Funcionais: Uma Abordagem Baseada em Sistemas Multi-Agentes

Tese apresentada como requisito parcial para obtenção do grau de Doutor pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

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

Prof. Carlos José Pereira de Lucena
Departamento de Informática – PUC-Rio

Prof. Alessandro Fabricio Garcia
Departamento de Informática – PUC-Rio

Profª. Vera Maria Benjamim Werneck
Departamento de Informática e Ciência da Computação – UERJ

Profª. Claudia Cappelli
Universidade Federal do Estado do Rio de Janeiro - Unirio

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

Rio de Janeiro, 10 de Abril de 2014

André Luiz de Castro Leal

Atua no mercado de TI desde 1992. Prof. da Universidade Federal Rural do Rio de Janeiro no curso de Sistemas de Informação. Mestre em Ciência da Computação pela Universidade Federal de Viçosa. Especialista em Ciência da Computação pela Universidade Federal de Viçosa. Especialista em Gestão de TI pela Faculdade Machado Sobrinho de Juiz de Fora.

Ficha Catalográfica

Leal, André Luiz de Castro

Análise de Conformidade de Software com Base em Catálogos de Requisitos não Funcionais: Uma Abordagem Baseada em Sistemas Multi-Agentes; orientador: Julio Cesar Sampaio do Prado Leite. - 2014.

206 f. ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2013.

Inclui bibliografia.

1. Informática – Tese. 2. Análise de Conformidade 3. Requisito não Funcional. 4. Sistemas Multi-Agentes. 5. Modelagem Orientada à Meta. I. Leite, Julio Cesar Sampaio do Prado Leite. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática.

CDD: 004

Ao meu filho Lucas, minhas desculpas pela ausência nessa longa caminhada e ao meu filho Tiago, que chega ao mundo no término desta, mas nos acompanhará em um novo ciclo de vida.

Agradecimentos

Aos amigos Henrique Sousa e Eduardo Almentero pelos diversos momentos de troca de experiências fundamentais para a aquisição do conhecimento e motivação para seguir nessa caminhada.

Aos colegas do Grupo de Pesquisas de Engenharia de Requisitos da PUC-Rio pelas discussões enriquecedoras sobre os temas e que sem elas esse trabalho não poderia ter sido concluído.

Aos membros da banca de qualificação e de defesa da tese pela disponibilidade de tempo e observações apresentadas para o enriquecimento do trabalho.

À FAPERJ e PUC-Rio pelo consentimento de bolsa de estudos e isenção sem as quais não seria possível a realização desse curso.

Aos funcionários da secretaria do departamento de informática, em especial à Regina Zanon, pela pronta atenção e solução de problemas do cotidiano de estudante ao longo desses anos.

Em especial ao Prof. Julio Leite pela paciência, confiança e direcionamento da pesquisa.

Resumo

Leal, André Luiz de Castro; Leite, Julio Cesar Sampaio do Prado. **Análise de Conformidade de Software com Base em Catálogos de Requisitos não Funcionais: Uma Abordagem Baseada em Sistemas Multi-Agentes.** Rio de Janeiro, 2014. 206p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A análise de requisitos não funcionais (RNF) é um desafio e vem sendo explorado na literatura científica. Tal iniciativa deve-se ao fato da existência do problema de se verificar o uso das operacionalizações desse tipo de requisito no *software* construído. Nessa tese apresenta-se um método, com técnicas e ferramentas de apoio, que analisam se um software está em conformidade com padrões de RNFs estabelecidos em catálogo como alternativa para o problema de análise de RNF. A estratégia adotada nessa tese utiliza agentes autônomos para análise de conformidade de *software* em relação a operacionalizações de RNF. Para isso, utiliza uma base de conhecimentos de padrões persistidos em um catálogo. Os resultados parciais são indicativos de que a proposta de solução é aplicável. A avaliação da validade dá-se por demonstração de que um método parcialmente automatizado é eficaz na identificação de conformidades. Um diferencial do trabalho apresentado é a ligação dos RNFs a sua efetiva implementação. Para demonstração da tese aplicou-se e customizou-se uma técnica de padrões de RNFs, baseados em orientação a metas, em estudos de caso de exemplos do cotidiano prático de *software.* Apresentamos também a construção de um *framework* de agentes, que operam sob notações XML para identificar conformidades de *software* em relação a um catálogo de RNF.

Palavras-chave

Analise de Conformidade; Requisito não Funcional; Sistemas Multi-Agentes; Modelagem Orientada à Meta.

Abstract

Leal, André Luiz de Castro; Leite, Julio Cesar Sampaio do Prado (Adivisor). **Software Compliance Analysis Based on Softgoal Catalog: A Multi-Agents Systems Approach.** Rio de Janeiro, 2014. 206p. DSc Thesis – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

The analysis of non-functional requirements (NFR) is a challenge and has been explored in the literature. This initiative is due to the fact of the existence of the problem of analysis the use of the NFR's operationalization in software. In this thesis we present a method, with supporting tools and techniques, that checks, if a software complies with standards of non-functional requirements as described in a catalog, as an alternative to the NFR analysis problem. The strategy adopted in this thesis uses autonomous agents to check software compliance regarding the operationalization of an NFR, by using a knowledge base of patterns persisted in a catalog. Initial results show that the proposed solution is applicable. The evaluation of the validity is given by the demonstration that a partially automated method is effective in identifying compliance. This work differs form others by linking NFRs to their effective implementation. A method based on patterns NFRs was used in common software, as to show the application of the proposed strategy. An agent based framework, working with XML descriptions, for checking software compliance with respect to a NFR catalog was built.

.

Keywords

Analysis of Compliance; Non-Funcional Requirements; Mult-Agent Systems; Goal Oriented.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ADF	*Agent Definition File*
ASCii	*American Standard Code for Information Interchange*
BDI	*Belief Desire Intention*
C&L	Cenários e Léxico
CTS	Catálogo de Transparência de *Software*
ER	Engenharia de Requisitos
FIPA	*Foundation for Intelligent Physical Agents*
GORE	*Goal-Oriented Requirements Engineering*
GQM	*Goal-Question-Metric*
GQO	*Goal-Question-Operationalization*
i*	iStar, iEstrela ou Intencionalidade Distribuída
IP	*Intentional Painel*
JADE	*Java Agent Development Environment*
JADEX	*Java Agent Development Environment eXtension*
LEL	Léxico Estendido da Linguagem
LS	Lattesscholar
NCSA	*National Center for Supercomputing Applications*
NFR	*Non-Functional Requirements*
RNF	Requisito Não-Funcional
SADT	*Structured Analysis and Design Technique*
SD	*Strategic Dependency*
SIG	*Softgoals Interdependency Graph*
SMA	Sistema Multi-Agentes
SR	Strategic Rationale
UdI	Universo de Informação
XML	*eXtensible Markup Language*

1.
Introdução

"Somos o que fazemos, principalmente o que fazemos para mudar o que somos"
Eduardo Hughes Galeano

Este trabalho apresenta um método sistêmico para análise de RNF com metas definidas em catálogo e implementadas em software. O método é baseado na abordagem de modelos GORE mapeados para estruturas XML que servem como regras para Sistemas Multi-agentes (SMA) atuarem na análise de conformidade do software com o catálogo estabelecido. A pesquisa foi feita a partir da evolução da aplicação de SMA em um catálogo de padrões de RNFs desde metas definidas em modelo conceitual até operacionalizações funcionais que são indicadas para implementação. Nessa seção, expõe-se a motivação do trabalho, os problemas relacionados frente à análise de qualidades, subjetivas e dependentes do olhar do espectador, descreve-se sucintamente a solução proposta, apresentam-se as contribuições e finalmente a organização do documento.

1.1
Motivação

Definições, relações, procedimentos, catálogo, análise e processos relativos a requisitos não funcionais (RNFs) são amplamente estudados no meio científico conforme constatado em [Chung et al. 2000], [Supakkul et al. 2010], [Leite 2006], [Cappelli 2009]. Particularmente, nos estudos de Leite (2006), o autor sugere a necessidade de aprofundamento de estudos na avaliação de RNFs no contexto de Transparência. Segundo o autor, Transparência tem uma peculiaridade em sua decomposição em outros RNF fortemente inseridos no contexto de *software*. Em [Leite 2006] argumenta-se sobre os desafios dos Engenheiros de *Software* frente a necessidade de entendimento sobre avaliação da Transparência, tomada como RNF em *software* [Leite et al. 2010].

A Transparência é abordada por Leite (2006) sob diversos aspectos: aborda os meios de tornar *software* transparente e os pontos de vista nesse contexto de avaliação; apresenta as demandas de cientistas, sociedade e as contribuições que o tema pode trazer para a comunidade; aborda em quais locais a Transparência pode ser importante em produtos, serviços e processos, aborda inclusive em diferentes contextos de *software*, como os embarcados; coloca a questão da periodicidade, frequência e duração, ou seja, trata de questões temporais do uso da Transparência;

apresenta as fontes das demandas por transparência em *software*, cita a comunidade, os Engenheiros de *Software* e diferentes atores interessados (agências reguladoras, ONGs, cidadãos); demonstra sua preocupação de como se pode criar métodos, processos, definições de conceitos, níveis de abstração para que o RNF de Transparência possa ser tratado com base em critérios técnicos. Leite (2006) argumenta que a Transparência, dada como RNF [Leite e Cappelli 2010], é um requisito peculiar, uma vez que sua análise depende de uma caracterização sistemática que envolve a sua decomposição em outros RNFs e também suas correlações. Algumas decomposições de Transparência são propostos em [Cappelli 2009] a partir de requisitos como: acessibilidade, usabilidade, informativo, entendimento, auditabilidade.

É a partir dessa contextualização de Transparência e sua decomposição em diversos outros RNFs que se indica um caminho a ser investigado, ou seja, utilizar de tais características consolidadas sistematicamente para prover se RNF está implementado em software. Algumas abordagens teóricas indicam um caminho a ser estudado, principalmente abordagens de *Goal-Oriented Requirements Engineering* (GORE) dentre essas pode-se destacar: o *framework* de Chung et al. (2000) a respeito da modelagem de RNF; o método ERi*c [Oliveira et al. 2008]; Transparência em Processos Organizacionais Utilizando Aspectos [Cappelli 2009]; conceitos e abordagens sobre modelos intencionais, estudos esses efetuados a partir da literatura de KAOS [Lamsweerde 2001], i* *Framework* [Yu 1995], V-graph [Yu et al. 2004], GBRAM [Anton 1997], Tropos [Castro 2002] e [Bresciani 2004].

Diante dessa peculiaridade de Transparência como RNF pesquisas iniciadas avançaram na criação do Catálogo de Transparência de *Software* (CTS) [CTS 2013] e demonstraram sua forte relação com outros RNFs como Acessibilidade, Usabilidade, Informativo, Entendimento e Auditabilidade [Cappelli 2009] definidos no catálogo como *softgoals* de Transparência, além de um conjunto de questões para cada meta flexível. Essas questões se relacionam às alternativas funcionais para operacionalizar as metas flexíveis e foram obtidas como resultado de uma adaptação do método *Goal-Question-Metric* (GQM) [Basili 1992] para um método *Goal-Question-Operationalization* (GQO) [Serrano e Leite 2011a].

Com um catálogo pré-concebido e as abordagens orientadas à metas surge a oportunidade da pesquisa de como analisar RNFs a partir de suas decomposições e operacionalizações estabelecidas em catálogo e analisados por SMA. A proposta desse trabalho pretende ser uma alternativa para a análise de RNFs a partir da associação de *patterns à* modelagem conceitual de metas flexívies e operacionalizações de respostas funcionais que possam satisfazer à RNFs.

Como se trata de um método baseado em metas (*goals*), ou seja, os RNFs analisados no *software* são definidos como metas a serem atingidas, uma possível aplicação seria operacionalizar o método a partir de SMA [Wooldridge 2002] [Jennings e Wooldridge 2002] com agentes fazendo a coleta de dados em diferentes tipos de artefatos para que possam ser comparados com um catálogo de metas pré-estabelecidas. O uso de SMA justifica-se pela sua proposta de utilização de bases de conhecimento, modelos baseados em raciocínio e autonomia dos agentes, bem como a possibilidade de sua arquitetura orientada a objetivos [Wooldridge 2002].

- **É tema dessa tese um método sistêmico para análise de conformidades de RNF implementados em *software*.**

1.2
Caracterização do Problema

Requisitos de qualidade em *software* propõem características em alto nível de abstração e sua avaliação possui grau de subjetividade que depende do ponto de vista do interessado [Fenton e Pfleeger 1997]. Por exemplo, dizer se é transparente ou não, tomando Transparência como RNF [Leite e Cappelli 2010], pode não ser adequado, pois irá depender do grau de subjetividade de quem está na expectativa da informação. Além disso, conforme apresentado no CTS (2013), a Transparência possui decomposições em outros RNFs e elos de relação entre eles que podem sugerir um impacto de julgamento ainda mais complexo, uma vez que tais relações podem ser positivas ou negativas [Cappelli 2009]. Essas decomposições são tratadas de *softgoal* a partir do *framework* de Chung et al. (2000) e também possuem grau de subjetividade de julgamento.

Enquanto um requisito funcional (RF) é uma declaração do que o *software* deve ou não deve fazer, normalmente expressa na forma: se uma dada condição é, em seguida, o *software* deve responder apropriadamente; um RNF é uma qualidade ou restrição inerente ao comportamento do *software* diante dessa resposta [Sommerville 2011]. A análise dessas condições torna-se mais efetiva para os RFs, uma vez que dada a condição, a análise e validação da resposta dada pelo *software* é menos passível de controvérsia, pois é determinística. Já para RNFs a validação de sua implementação é algo mais suscetível ao olhar de quem analise. A implementação de uma determinada qualidade pode satisfazer ao julgamento de um usuário, mas pode não estar adequada às expectativas de outro usuário. Além disso é composta por características que não podem ser implementadas diretamente, em vez disso são satisfeitas em alguns casos pela combinação de diversos RFs [Sommerville 2011].

A literatura apresenta que RNFs são difíceis de se expressar e consequentemente de verificar e validar nas diversas fases do *software* porque: certas restrições estão relacionadas com a solução de desenho, que é desconhecido na fase de requisitos [Fidge e Lister 1993]; certas restrições são altamente subjetivas e só podem ser determinadas através de avaliações empíricas complexas [Matoussi e Laleau 2008]; RNFs podem se relacionar a um ou mais RFs e possuem dessa maneira uma característica de ortogonalidade [Leite e Cappelli 2010]; RNFs tendem a conflitos e contradições, uma vez que são sucetíveis à percepção de quem o avalia [Kaiya e Kaijiri 1999]; não há regras "universais" e diretrizes para determinar quando RNFs são plenamente atendidos [Glinz 2007].

Analisar RNF implementado em *software* a partir de sua natureza semântica ou conceitual é uma tarefa difícil, portanto faz-se necessário refinar esses requisitos até que se tornem elementos mais concretos de julgamento [Chung et al. 2000]. Fenton e Pfleeger (1997) consideram que RNFs devam ser subdivididos em um conjunto de características funcionais, para que essas possam ser implementados em *software* para atender às expectativas do usuário com relação aos RNFs elicitados.

Técnicas de análise têm sido amplamente utilizadas para verificar e validar os requisitos. Lamsweerde (2001) e Lamsweerde e Letier (2000) sugerem especificações hierárquicas de decomposição de metas produzindo estados a serem alcançados e do comportamento do sistema necessário para atingir esses estados para assim verificar RNFs. Lamsweerde (2001) argumenta que de modo considerável o refinamento de RNFs é necessário para que o raciocínio automatizado possa ser aplicado.

Estabelecer um método que faça uma união de abordagens de definição de RNFs, seus refinamentos em questões e operacionalizações transformados em catálogo, para minimizar o grau de subjetividade do julgamento dos RNFs é um desafio. O termo catálogo de RNF é utilizado a partir da especialização em camadas de padrões (*patterns*) que envolvem desde requisitos de alto nível de abstração, como qualidades, até RFs de mais baixo nível ou operacionalizações [Chung et al. 2000] [Supakkul et al. 2010]. O objetivo de sua criação em *patterns* é possibilitar reuso de seu conhecimento. Supakkul et al. (2010) propõem diferentes tipos de *patterns* tais como: padrões objetivos, padrões problema, padrões alternativas e padrões seleção que serão apresentados em seções posteriores.

Chung et al. (2000) propõem um *framework* para representar RNFs e suas relações. Serrano e Leite (2011a) adaptam a proposta de Supakkul et al. (2010) quando exploram no modelo GQO com a inserção de padrões do tipo questão para servir de elo entre as operacionalizações e os *softgoals*.

Um ponto pouco explorado na literatura é operacionalizar *patterns* de RNFs em uma estrutura única para que possa servir de base para automação da análise da implementação de RNFs. Tal abordagem mostra-se importante, uma vez que analisar os desmembramentos dos RNFs e suas relações com alternativas de operacionalizações que os satisfaçam podem não ser triviais necessitando assim de um mecanismo automático que auxilie na análise. A partir de um catálogo pode ser possível uma análise no sentido de saber se as operacionalizações providas pelo catálogo foram implementadas no *software*.

Diante dessas considerações, surgem as seguintes questões de pesquisa:

- **É possível estabelecer um método sistêmico que contenha regras para análise de conformidade de implementação de RNF em *software*?**
- **Como estabelecer um método único, a partir de padrões conceituais e operacionais, que possibilide análise de conformidade de implementação de RNF em *software*?**
- **Como um mecanismo automático pode ser utilizado para analisar padrões conceituais e operacionais com objetivo de identificar conformidades de implementação de RNF em *software*?**

1.3 Enfoque da Solução

No sentido de tornar possível um método sistematizado para análise de RNF, o objetivo primeiro da tese é estabelecer tal método a partir de uma abordagem *TOP-DOWN* ao especificar RNF a partir de modelos conceituais e focar seu detalhamento em direção às operacionalizações, até que possam ser analisadas por agentes autônomos. O trabalho utiliza, para estudos de caso, os RNFs sistematizados no catálogo de *software* [CTS 2013]. Estão presentes no catálogo requisitos do tipo Acessibilidade, Usabilidade, Informativo, Entendimento, Auditabilidade e seus desdobramentos em outras qualidades [Leite e Cappeli 2010]. Dessa forma, o catálogo proposto em CTS (2013) disponibiliza uma série de RNFs caracterizados conceitual e operacionalmente que podem permitir uma análise da conformidade desses requisitos implementados em *software*.

A elicitação, especificação e refinamento dos requisitos presentes no CTS é concebido a partir da proposta do NFR *Framework*, tratado no trabalho como RNF *framework*, de Chung et al (2000). O *framework* propõe a representação das qualidades e seus elos de contribuição positivos ou negativos, além de seu nível de

satisfação (satisfeito a contento). Tais qualidades são tratadas por Chung et al. (2000) como *softgoals* (metas-flexíveis), que são representados graficamente no chamado *Softgoal Interdependency Graph* (SIG).

No SIG também é possível representar operacionalizações e relacioná-las aos *softgoals* correspondentes. Essas operacionalizações (metas concretas) são maneiras ou funcionalidades das quais pode-se satisfazer os *softgoals*.

Apesar da proposta de Chung et al. (2000) sobre o uso de operacionalizações, adota-se nessa tese a proposta de operacionalizações do GQO de Serrano e Leite (2011a), por entendermos que GQO é uma ferramenta que minimiza a identificação de operacionalização e suas relações aos *softgoals* por meio do uso de questões.

Assim como o GQM [Basili 1992], o GQO procura elencar elementos que possam tornar as metas, ou no caso RNFs, susceptíveis a avaliação. O GQO utiliza de uma abordagem do uso de boas práticas de *software* em sua camada *Operacionalization* (O). As operacionalizações, segundo a proposta do GQO, devem representar práticas concretas com menor subjetividade de representação para facilitar análise, julgamento e implementação que possam satisfazer as metas definidas no SIG. Como exemplo, pode-se utilizar alternativas, camada *Operationalizations* (O), de RFs que dêem suporte aos RNFs estabelecidos nas camadas de *Goal* (G).

O presente trabalho de tese apresenta uma proximidade com a abordagem de SMA, uma vez que trata de objetivos a serem atingidos. Agentes em SMA interagem com intuito de solucionar demandas de sistema, ou metas (*goals*) definidas. O uso de SMA também propõe a autonomia das decisões dos agentes para atingir metas definidas, além de contribuição mútua dos agentes para atingir metas em comum. O trabalho propõe o uso de agentes com orientação baseada em intencionalidade, agentes intencionais segundo Bratman (1999). Os agentes baseados em modelos intencionais possuem melhor capacidade de raciocínio e capacidade de aprendizado de agentes inteligentes a partir de bases de conhecimento ou heurísticas, por exemplo. Essas ações inteligentes são baseadas em conceitos como crença, desejo e intenção (*Belief-Desire-Intention* - BDI) [Braubach et al. 2003] [Rao 1996]. Portanto, o uso de SMA justifica-se pelos seguintes características [Wooldridge 2002] [Jennings e Wooldridge 2002]:

- capacidade pró-ativa, possuem a capacidade de atuar diante às alterações de ambiente e possuem comportamento orientado à objetivos;

- comportamento social, onde atuam na interação com outros agentes para resolver seus problemas ou auxiliar na resolução de um objetivo comum. Para isso possuem a capacidade de comunicar seus requisitos internos a outros agentes e

arquitetura interna orientada à decisão que permite definir quais interações são mais apropriadas;

- autonomia, pois executam grande parte de suas ações sem a necessidade de interferência humana ou outros agentes computacionais, possui controle de suas ações e estado interno. Possui a capacidade de agir segundo seus próprios objetivos;

- coordenação entre seus agentes para trabalhar a dependência das ações os agentes envolvidos, os indivíduos não solucionam os problemas sozinhos, necessidadem respeitar a regras globais para execução de tarefas.

Esse trabalho propõe: o uso de agentes para atuar em ambiente com objetivos definidos (*goal-oriented*), um conjunto de agentes distribuídos com objetivos distindos, especializados em suas tarefas, um conjunto de relações necessárias para atingir aos objetivos, um conjunto de operações, operadores que representam os resultados das operações e as reações do ambiente a eles.

Como a especificação do problema da análise dos RNFs estão modelados sobre uma abordagem orientada a objetivos, entende-se que uma arquitetura do SMA em BDI seja apropriada, pois, objetivos, *softgoal* e tarefas serão mapeados para metas, desejos, intenções e crenças conforme proposta de Serrano e Leite (2010b).

Diante do exposto, a hipótese do trabalho pode ser resumida como:

- **A análise de conformidade de RNFs implementados em *software* pode ser feita desde que esses requisitos estejam definidos em catálogo a partir de metas flexíveis e operacionalizações.**

Nesse trabalho foram feitos estudos de caso a partir de RNFs definidos no CTS de Transparência na intenção de validar o esboço teórico e consequentemente contribuir para os estudos da análise de RNF e de Transparência de *Software*. Essa tese construiu o método, suas técnicas e ferramentas de apoio foram aplicados em exemplos de *software* reais. Os resultados, parciais, são indicativos de que a proposta de solução é aplicável. Para demonstração da tese aplicou-se e customizou-se a técnica de padrões de RNFs, construi-se um framework de agentes, utilizou-se notações XML tanto na definição do catálogo, como na configuração dos artefatos de *software* objeto da análise, construiu-se instâncias para a base de conhecimento de RNFs (SIG) e aplicou-se estudos de caso.

Uma das principais contribuições esperadas com a tese é que de maneira geral as propostas de análise de RNF, com foco em verificação, são fortemente voltadas à visão funcional. A proposta aqui apresentada é inovadora na ligação dos RNFs a sua efetiva implantação, sendo o RNF o requisito de primeiro plano a ser considerado para sua posterior operacionalização.

As contribuições esperadas com o trabalho são: a) definição de um método sistêmico, baseado em modelos conceituais definidos por *patterns*, que possibilite agentes autônomos analisarem conformidades de RNFs implementados em *software*; b) estabelecer arquitetura operacional para o CTS (2013), a partir de padrões estabelecidos [Supakkul et al. 2010] [Serrano et al. 2010], para que o mesmo possa ser analisado por automação; c) fazer a análise de conformidade de RNF em artefatos de software, traços de execução e código fonte, a partir de catálogo definido por padrões; d) utilizar um mecanismo automático baseado em SMA para analisar os valores de propagação, elos do tipo HELP [Chung et al. 2000] e ANSWER [CTS 2013] das relações entre operacionalizações e *softgoals* de RNFs. Tal mecanismo, baseado em SMA, teve sua primeira versão implementada conforme Leal et al. (2013a) e estendido nesse trabalho de tese; e) inserir na abordagem de avaliação dos RNFs, estruturas suportadas por proveniência (*provenance*) [Miles 2007] para que sirvam de base de conhecimento para otimização dos processos de análise dos agentes do SMA. O uso da proveniência, com foco na rastreabilidade, utilizado no método provê mecanismo para otimização do processo de análise uma vez que rastros históricos de análises anteriores passam a ser consideradas para diagnóstico de análises futuras. [Leal et al., 2013a].

1.4
Organização da Tese

A tese está organizada em 6 (seis) capítulos:

- no capitulo 2 são apresentados os pressupostos teóricos que fundamentam o desenvolvimento do trabalho de pesquisa, são apresentados temas como GORE, SMA, modelagem intencional e padrões de RNFs;

- no capítulo 3 são descritos os passos do método sistêmico para análise de RNFs. Inicialmente descreve-se um contexto histórico da pesquisa para inserir o leitor sobre alguns trabalhos já desenvolvidos, em seguida é apresentado o método, desde sua estrutura central, a partir de modelos conceituais até seus desdobramentos em operacionalizações a partir do SMA. Nas etapas de descrição do método são elaborados modelos em SADT [Ross 1997] para facilitar a compreensão a partir de modelos visuais e em seguida suas descrições (*rationales*).

- no capítulo 4 são apresentados um modelo conceitual dos *patterns* de RNF, a estrutura do catálogo XML, as estruturas criadas a partir de léxicos e a arquitetura dos agentes do SMA, elaborada a partir da modelagem intencional, bem como a arquitetura de implementação dos agentes;

- no capítulo 5 são apresentados os estudos de caso onde demonstra-se a aplicação do método para análise de RNF em diferentes artefatos de *software*, com diferentes propósitos de avaliação de RNFs, por exemplo, em tempo de projeto e em tempo de execução;

- no capítulo 6 são apresentadas as conclusões, como também trabalhos correlatos, uma revisão das contribuições da pesquisa, as limitações e trabalhos futuros.

2.
Fundamentação Teórica

Nesse capítulo são apresentados os pressupostos teóricos que dão suporte à tese. Aqui estão em destaque os principais conceitos das abordagens utilizadas na criação do método para análise de RNF. Outro ponto importante de deixar estabelecido é o uso de métodos baseados na abordagem GORE, que é utilizada para modelar a natureza do problema ou o objeto da análise.

2.1
Goal-Oriented Requirements Engineering - GORE

Em GORE, a abordagem de aquisição de conhecimento inicial é realizada pela descoberta das metas (*goals*) dos interessados, portanto, o conhecimento é adquirido inicialmente a partir da análise das metas desses interessados no projeto de *software*. Dardenne et al. (1993) coloca que as metas provêem uma fonte de informação básica para detectar e resolver conflitos que surgem a partir de diferentes pontos de vista.

De outro lado, Lapouchnian (2005), Rumbaugh et al. (1991), Ross (1977) e DeMarco (1978) argumentam que as abordagens tradicionais no contexto de software focam o desenvolvimento apenas na análise dos processos e dados e não capturam o conhecimento (*rationale*) para o que é desenvolvido. Além disso, argumentam que os projetos de *software* tratam a definição do produto isoladamente, não se preocupando com o contexto a que ele está inserido, dessa forma, os RNFs são tratados perifericamente nas especificações de requisitos.

Em GORE há uma preocupação de se tornar as metas, organizacionais ou de *software*, o foco principal da modelagem e a partir deles efetuar refinamentos até que se possam reduzir tais metas em um conjunto de requisitos funcionais (RF) [Mylopoulos 2006].

As metas podem ser representadas a partir de objetivos (*hard goals*) ou como metas-flexíveis (*softgoals*) [Mylopoulos 2006]. Para os primeiros há uma maior possibilidade de formalização por se tratarem de requisitos que permitem uma análise e avaliação precisa de ser ou não alcançada. Os demais, como qualidades, são menos passíveis de julgamento e permitem apenas uma avaliação de que são satisfeitos a contento (*satisficed*) [Mylopoulos 2006] [Cunha 2007] e não como satisfeitos (*satisfied*) [Mylopoulos 2006]. Mylopoulos (1992) e Chung (2000) tratam as qualidades como *softgoals* ou *fuzzy goals* que possam a partir de formalização receber uma avaliação.

Para Mylopoulos (2006), as principais vantagens de GORE estão no tratamento de forma sistemática da derivação dos requisitos a partir das metas, que provêem *rationales* para os requisitos, como também no refinamento da estrutura das metas em estruturas compreensíveis para a documentação de requisitos a partir de seus refinamentos.

2.2
RNF *Framework*

O RNF framework de Chung et al. (2000) procura estabelecer de forma sistemática uma visão de necessidade de elicitar qualidades já no início da construção do *software*. O *framework* procura tratar a aquisição de conhecimento sobre requisitos a partir de metas (*goals*). Em GORE, um dos problemas principais das metas é a identificação de como elas são alcançadas, principalmente em se tratando de RNF que possuem subjetividade de julgamento, seja na sua completude de definição ou mesmo na avaliação, que dependerá do ponto de vista de quem análisa [Dardenne et al. 1993].

Chung et al. (2000) definem o conceito de: RNF-*softgoal* (meta flexível) que representa o objetivo de qualidade que se quer atingir; a operacionalização que representa as ações a serem realizadas para institucionalização de determinado RNF-*softgoal*; e as contribuições que mapeiam os relacionamentos entre os outros dois elementos. Esses elementos são relacionados a partir do modelo SIG. O modelo permite uma visão sistemática dos *softgoals* devidamente decompostos, hierarquizados e operacionalizados, além de relacioná-los a partir de decomposições do tipo *and* e *or e* por meio de elos de contribuição, positivos ou negativos. A Figura 1 apresenta os elos de contribuição propostos no *framework.*

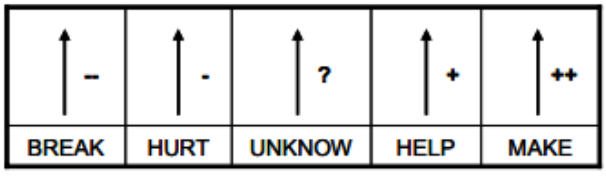


Figura 1: Diferentes tipos de relação possíveis no RNF framework [Chung et al. 2000].

Os tipos representam: a) *Break*: provê contribuição negativa suficiente para que a característica superior não seja atendida; b) *Hurt*: provê contribuição negativa parcial para não atendimento da característica superior; c) *Unknown*: provê contribuição, porém não se sabe se negativa ou positiva; d) *Help*: provê contribuição positiva parcial para atendimento da característica superior; e) *Make*: provê contribuição positiva suficiente para que a característica superior seja atendida [Chung et al. 2000]. Esses relacionamentos possibilitam uma análise de propagação de como um *softgoal*

influencia o outro, uma vez que todos os *softgoals* estão relacionados, mesmo sendo uma análise de cunho qualitativo. Para relacionar *softgoals* de diferentes grupos ou na mesma camada, é possível usar duas representações: *Some* + : que tem por objetivo representar contribuições positivas, tais como *Help* e *Make*; e *Some* - : que tem por objetivo representar contribuições negativas, tais como *Hurt* e *Break* [Chung et al. 2010].

A Figura 2 apresenta um exemplo de modelo baseado no *framework* com decomposições e elos do tipo HELP. O modelo é um recorte sobre a representação do *softgoal* Auditabilidade, retirado de [CTS 2013], onde o tópico é *software* (representado por [*software*]). Nele é possível ver a decomposição de Auditabilidade em outros cinco *softgoals: validade, controlabilidade, verificabilidade, rastreabilidade* e *explicação*. Entende-se que esses *softgoals* contribuem positivamente com Auditabilidade a partir de seus elos de ligação do tipo *Help.*

As operacionalizações, representadas por nuvens em destaque, no caso negrito, estão alocadas abaixo de seus *softgoals* correspondentes e denotam contribuições para com esses. A propagação entre os diversos elos do grafo irão denotar as contribuições positivas ou negativas para as camadas superiores ou para elementos relacionados a partir dos elos.

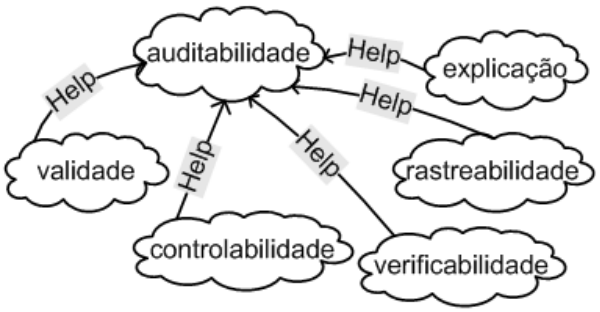


Figura 2: SIG Auditabilidade [Cappelli 2009].

Os significados dos atributos podem ser vistos na Tabela 1.

Tabela 1: Detalhamento dos atributos de Auditabilidade. Fonte: [Cappelli 2009]

Graus de transparência	Atributos	Descrição do atributo
Auditabilidade	Validável	Capacidade de ser testado por experimento ou observação para identificar se o que está sendo feito é correto.
Auditabilidade	Controlabilidade	Capacidade de domínio.
Auditabilidade	Verificabilidade	Capacidade de identificar se o que está sendo feito é o que deve ser feito.
Auditabilidade	Rastreabilidade	Capacidade de seguir o desenvolvimento de um processo ou a construção de uma informação, suas mudanças e justificativas.

Auditabilidade	Explicável	Capacidade de informar a razão de algo.

O recorte da representação de SIG de *Auditabilidade* é extraído do SIG de *Transparência* defendido por Cappelli (2009). A autora define um conjunto de características que podem contribuir com transparência, conceituadas pela autora como Graus de Transparência (GT). São cinco graus na camada mais alta de uma hierarquia que auxilia a maturidade em transparência, que são denominados: a) Acessibilidade: a transparência é realizada através da capacidade de acesso. Esta capacidade é identificada através da aferição de práticas que implementam características de portabilidade, operabilidade, disponibilidade, divulgação e desempenho; b) Usabilidade: a transparência é realizada através das facilidades de uso. Esta capacidade é identificada através da aferição de práticas que implementam características de uniformidade, intuitividade, simplicidade, amigabilidade e compreensibilidade; c) Informativo: a transparência é realizada através da qualidade da informação. Esta capacidade é identificada através da aferição de práticas que implementam características de clareza, acurácia, completeza, corretude, consistência e integridade; d) Entendimento: a transparência é realizada através do entendimento. Esta capacidade é identificada através da aferição de práticas que implementam características de composição, concisão, divisibilidade, dependência, adaptabilidade e extensibilidade; e) Auditabilidade: a transparência é realizada através da auditabilidade. Esta capacidade é identificada através da aferição de práticas que implementam características de explicação, rastreabilidade, verificabilidade, validade e controlabilidade. A Figura 3 apresenta os tipos de contribuição, atribuídos dos GT e suas relações.

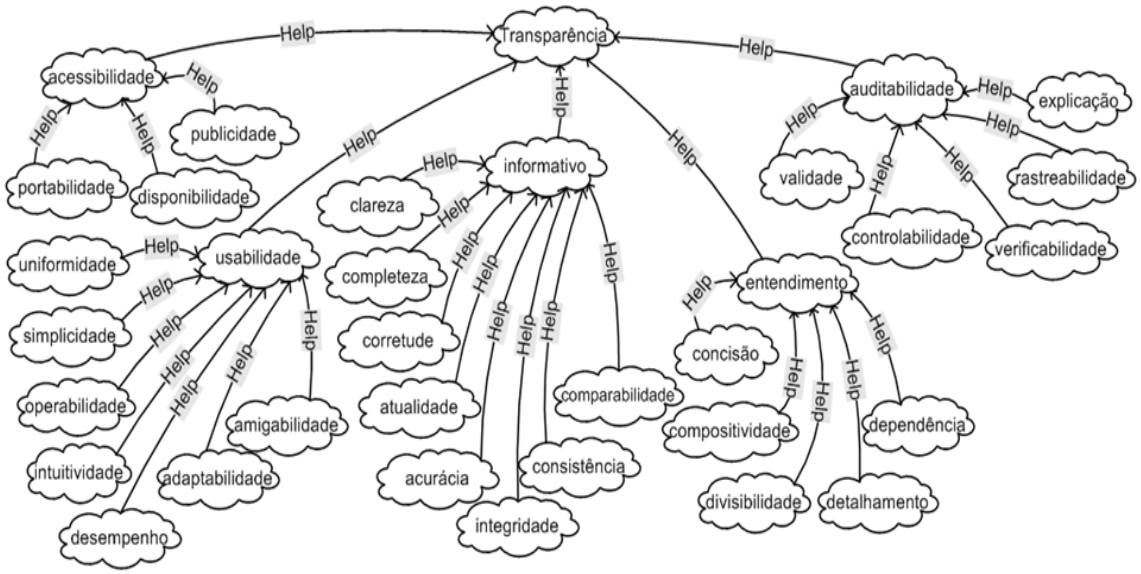


Figura 3: SIG de Transparência [Cappelli 2009].

A Figura 4 apresenta um detalhamento de Auditabilidade em operacionalizações, nuvens em negrito, que possam satisfazer os *softgoals* iniciais.

Figura 4: Modelo de Auditabilidade. Adaptado de [CTS 2013].

2.3
Goal-Question-Operacionalization

O método GQO proposto pelo Grupo ER PUC-Rio descrito no trabalho de [Serrano e Leite 2011a] apresenta uma proposta baseada no GQM [Basili 1992]. A proposta de Basili (1992) é uma abordagem orientada a metas para a mensuração de produtos e processos de *software.* Essa abordagem foi desenvolvida a partir da aplicação *top-down* de um programa de mensuração e a análise dos dados a partir da interpretação *bottom-up*. No GQM se estabelecem metas (*goals* - *G*) a serem atingidas e para cada meta são identificadas questões (*questions* - *Q*) para que se possa refinar e determinar se os objetivos podem ser alcançados [Basili 1992]. A partir dessa abordagem qualitativa há uma determinação de elementos com características quantitativas que irão servir de respostas para as perguntas estabelecidas. Esses elementos são determinados como medições (*metrics* ou *measurements* - *M*). A Figura 5 apresenta a estrutura proposta do GQM.

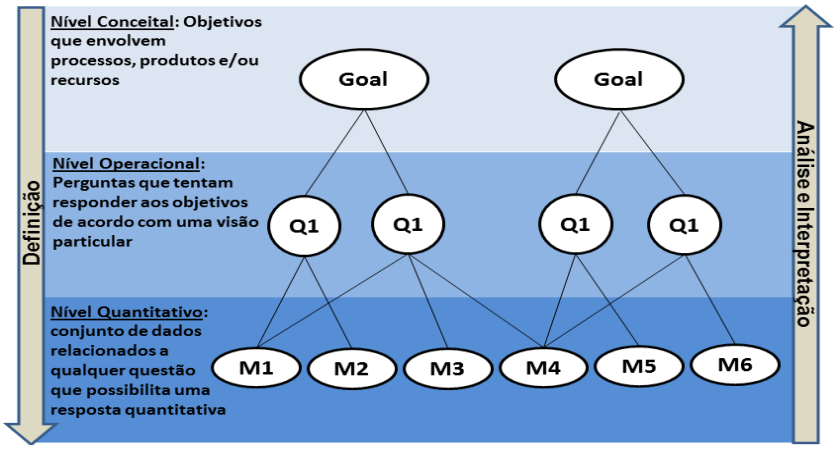


Figura 5: Estrutura GQM [Basili 1992].

O GQO possui a mesma estrutura, mas em sua camada de medição (M) utiliza uma alternativa de Operacionalização (*operacionalization* - *O*). Em tal camada, Serrano e Leite (2011a) estabelecem que os elementos de operacionalização sejam estabelecidos a partir de boas práticas (técnicas, métodos, ferramentas e procedimentos) da ES para que se possam determinar respostas para as questões.

2.4
Padrões de RNFs

Os padrões de RNFs focam no reuso do conhecimento de RNF a partir de uma série de padrões especializados com o propósito de definir e operacionalizar tais requisitos. O objetivo da criação dos padrões é o seu reaproveitamento para RNFs distintos. Supakkul et al (2010) trabalham a partir de quatro tipos de padrões para o reuso do conhecimento sobre RNFs: “Padrões Objetivos” (*Objective Patterns*) para capturar a definição do RNF em termos de *softgoals*; “Padrões Problema” (*Problem Patterns*) para representar obstáculos na satisfação dos softgoals; “Padrões Alternativas” (*Alternative Patterns*) que dispõe de diferentes alternativas para satisfazer os *softgoals*; e Padrões Seleção (*Selection Patterns*) que auxiliam na escolha das diversas alternativas por meio de análise quantitativa e qualitativa.

Particularmente em CTS (2003) usam-se os padrões de RNFs para Transparência de *Software* onde o requisito é representado desde sua definição a partir da sua decomposição em outras qualidades, transformadas em *softgoals* pela proposta do *framework* de Chung et al. (2000), até a operacionalização a partir de alternativas que satisfaçam as metas elaboradas.

Padrões de RNFs são utilizados para representar o conhecimento a partir do uso de uma estrutura baseada em *Inicial-Resultados-Refinamento Regras* (*Initial-Result-Refinement Rules*), onde Inicial capta um conceito-chave para dar contexto

para reutilização, Resultado para capturar o conhecimento disponível sobre o RNF e as Regras de Refinamento para capturar refinamentos modelo individuais e sua ordem de aplicação [Supakkul et al. 2010]. Essa abordagem propõe a captura e o registro do conhecimento a partir de um método *top-down* em que inicialmente são criados modelos conceituais para representação do conhecimento e ao longo do seu desenvolvimento são criadas estruturas funcionais para representar operacionalizações ou regras que possam satisfazer ao modelo inicial. Dessa forma, a proposta pretende tornar o conhecimento reutilizável, fácil de entender e possibilita uma seleção de operacionalizações, ou refinamentos, que possam atender a domínios ou necessidades específicas.

A Figura 6 apresenta o *pattern* Questão do *softgoal* de Desempenho presente no catálogo de Transparência [CTS 2013]. Nele é possível visualizar o *pattern* dividido em quadros. Na seção Resultado as questões são agrupadas de acordo com um Grupo para que fiquem juntas em um grupo de mesma finalidade. O catálogo tem a representação de R1 que indica que os grupos fazem parte de Desempenho fazendo com que o mesmo seja satisfeito de acordo com as respostas às perguntas desse agrupamento. Em [Serrano et al. 2010] é sugerida a abordagem GQO para que sejam identificadas operacionalizações que possam servir de respostas alternativas para as diversas questões do catálogo.

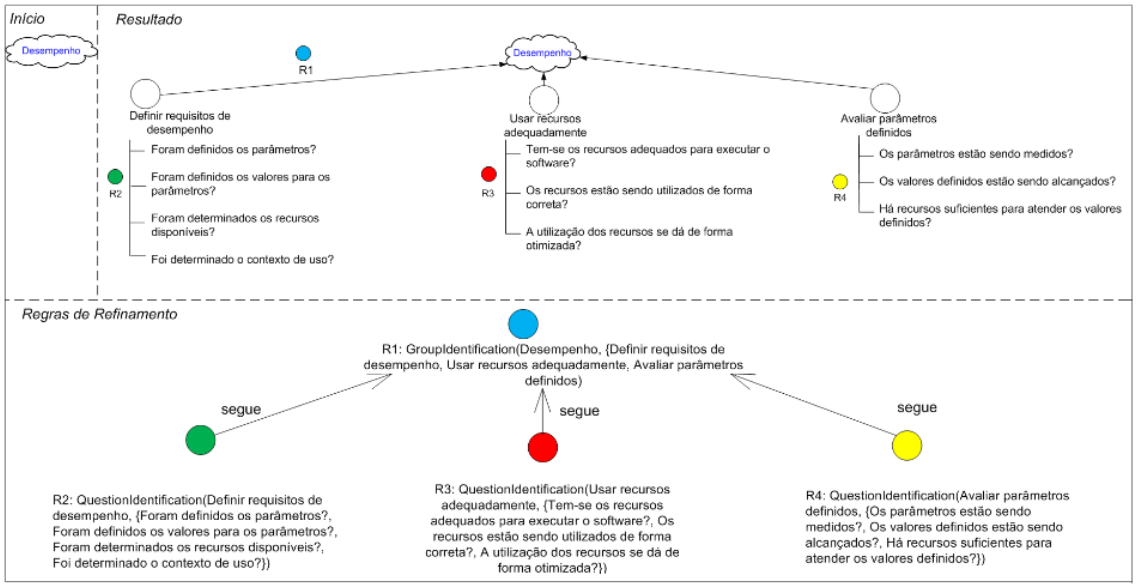


Figura 6: *Pattern* definido no catálogo de Transparência para o *softgoal* Desempenho [CTS 2013].

Diante do explicitado há uma junção de tipos *patterns* propostas pela aliança das abordagens de Supakkul et al. (2010) e Serrano et al. (2010) que propõem uma estrutura em camadas para os *patterns* conforme pode ser visto na Figura 7.

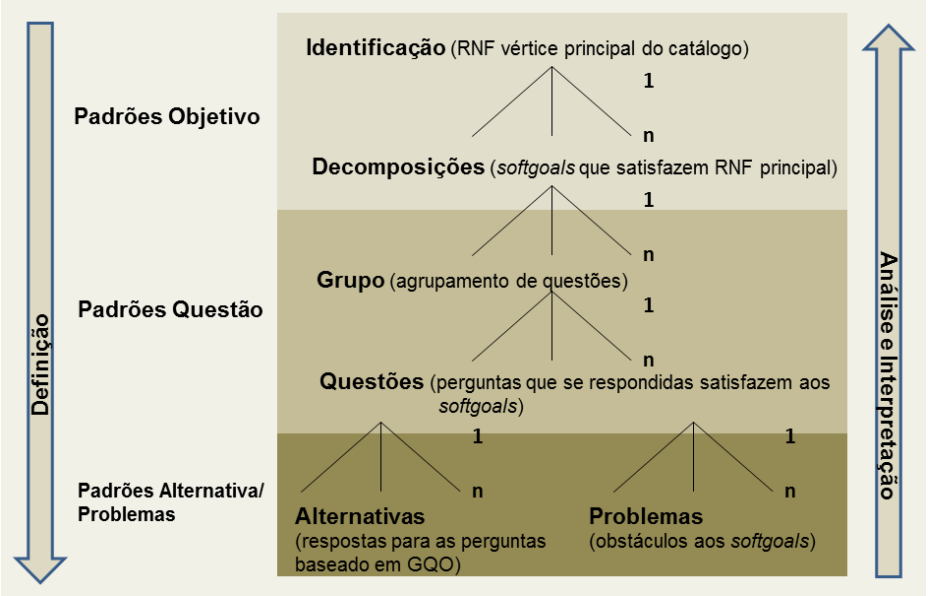


Figura 7: Camadas dos *Patterns*. *Adaptado de* [Supakkul et al. 2010] e [Serrano et al. 2010].

A inclusão da proposta de Serrano et al. (2010) propõe que na camada de Alternativas sejam utilizadas boas práticas de ES para compor operacionalizações (respostas alternativas) que possam atender às questões postas no modelo.

Nessa tese utilizaremos os padrões objetivos subdivididos em dois grupos, primeiro o que chamaremos de *pattern* Identificação (*Identification Pattern*). Esse padrão será utilizado para identificar aquele RNF que se deseja analisar, após essa identificação será utilizado o *pattern* Decomposição (*Decomposition Pattern*), que receberá as decomposições do *pattern* Identificação. Na Figura 7 estão presentes esses elementos na camada Padrões Objetivo.

2.5
Modelagem Intencional e o *Framework* i*

As motivações e interesses dos atores envolvidos em uma organização são o suporte do conceito de intencionalidade [Yu 1995], a modelagem intencional por sua vez é representada por aqueles métodos ou *frameworks* que possibilitam a criação de modelos intencionais.

O *iStar* (i*) [Yu 1995] é um *framework* convergente a esse tipo de abordagem. Seu fundamento principal está no fato de conceituar meta como: "*A goal is a condition or state of affairs in the world that an actor would like to achieve*" [Yu 1995]. Por ser um *framework* baseado em metas, i * é considerado pelos pesquisadores como GORE. Sua representação permite a modelagem a partir de 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. O *framework*

possui dois modelos que possibilitam trabalhar dois níveis de abstração: o modelo SD (*Strategic Dependency* – dependência estratégica) e o modelo SR (*Strategic Rationale* – raciocínio estratégico). O modelo SD apresenta os atores e suas dependências, enquanto que o modelo SR apresenta os detalhes internos dos atores. Adicionalmente o modelo de atores estratégicos descreve os conceitos de agente, papéis e posições.

As declarações explícitas nesse tipo de modelo permitem uma melhor avaliação das intenções internas dos atores, bem como naquelas compartilhadas com outros envolvidos. No modelo SD é possível a apresentação de agentes e suas dependências, já no SR estão representados esses detalhes internos dos agentes. Dessa forma, o SR representa tudo aquilo que o agente irá desempenhar para atingir as metas definidas, enquanto que no SD são identificadas as relações externas e sua dependência de agentes externos.

Em um modelo SD um agente pode depender de outro para satisfazer uma meta, para executar uma tarefa, fornecer um recurso ou satisfazer um *softgoal*. Os *softgoals* estão relacionados à RNFs, enquanto que *goals*, tarefas e recursos estão relacionados com funcionalidades do *software* [Yu et al. Castro 2008]. O agente que depende do outro é chamado de *Depender* enquanto que o outro agente que recebe a requisição é chamado de *Dependee* [Yu 1995]. O elo da dependência entre esses agentes é chamado de *dependum* e representa o objeto da dependência que pode ser: uma meta concreta, uma meta flexível, uma tarefa ou um recurso, e esse é sempre uma entidade física ou informacional.

No modelo SR aparecem elementos intencionais para descrever a análise de oportunidades e vulnerabilidades e o tratamento das dependências entre atores. Os elementos intencionais podem ser *goals*, tarefas, recursos, bem como outros tipos de relacionamentos como tipo *means-end (—▷)* (relação meio fim, que indica o tipo de meio (*means)* para atingir determinada meta (*goal*)), *decomposition* (*task*) (┼—) e contribuição [Yu 1995].

Um relacionamento de decomposição-tarefa existe entre uma tarefa e suas partes, mostrando como uma tarefa é executada. O relacionamento do tipo contribuição um *means* (tarefa ou *goal*) para a realização de um *end* (*softgoal*), esse tipo de relacionamento possibilita um raciocínio qualitativo representado por uma semântica a partir de rótulos do tipo: *Help, Hurt, Make*, + e – (Figura 1), além de variações de *Some+* (+ e ++) e *Some-* (- e --). Elementos que representam atores, agentes, papeis, tarefas, recursos, metas, *softgoal* e tipos de dependência são apresentados nas Figuras 8 e 9.

Figura 8: Elementos i* de modelagem intencional [Yu 1995].

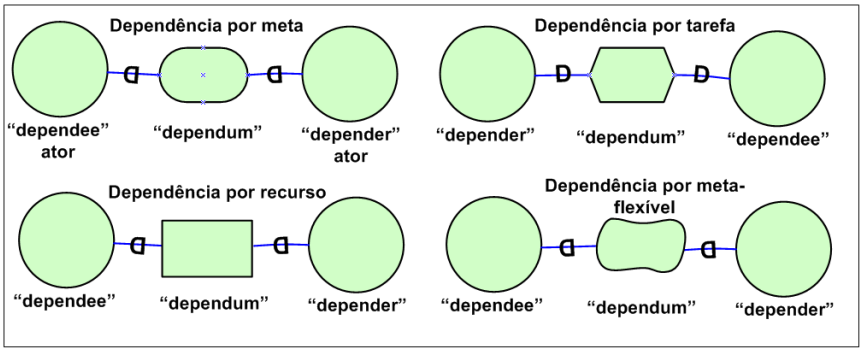


Figura 9: Tipos de relacionamentos [Yu 1995].

A escolha da abordagem i* para esse trabalho é justificada devido aos seus benefícios, tais como: a modelagem de contextos organizacionais baseado nos relacionamentos de dependências entre atores, obtenção de um melhor entendimento dos relacionamentos, a possibilita da compreensão das razões internas dos atores, uma vez que as mesmas são expressas de forma explícita, e o auxilio na escolha de alternativas durante a etapa de modelagem do *software* [Yu 1995].

2.6
Painel de Intencionalidades – Diagrama IP

A modelagem intencional é utilizada no trabalho para modelar conceitualmente a intencionalidade das interações dos agentes participantes no SMA construído para análise de RNFs.

O Diagrama IP proposto em Oliveira et al (2008) permite que inicialmente sejam modeladas metas (concretas e flexíveis), suas relações, e os atores ou agentes envolvidos em um contexto. O Diagrama IP é um diagrama de transição de estados por conter os estados (representados pelas metas) conectados pelas transições [Oliveira et al. 2008]. Seu uso é motivado pela representação da intencionalidade dos agentes em um único diagrama a partir de elos de inter-relações entre metas que representam dependência, correlação, contribuição e equivalência no mesmo diagrama.

A Figura 10 apresenta os elos de ligação a partir desses tipos de transição.

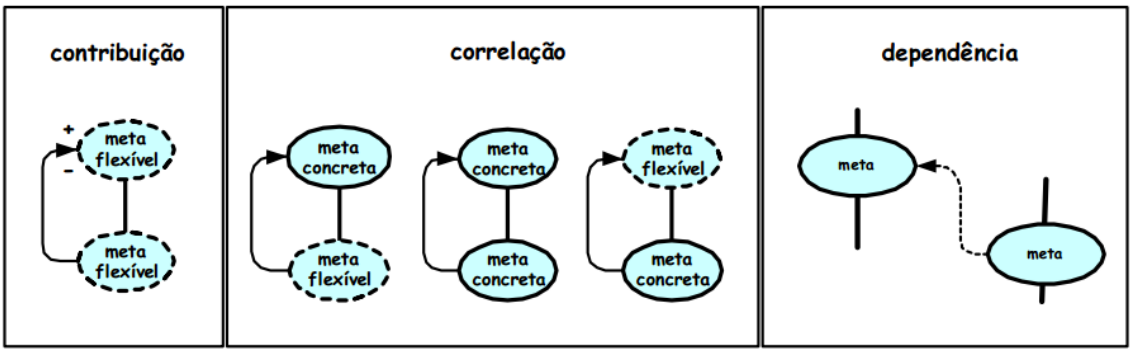


Figura 10: Ilustração das relações entre as metas em um Diagrama IP [Oliveira et al. 2008].

Segundo o autor da proposta do Diagrama IP [Oliveira et al. 2008] as conceituações ou características desses elos de ligação podem ser vistos como:

A) Correlação entre duas metas: ocorre entre duas metas de um mesmo ator. Uma meta inicial, decomposta em uma atividade, é o meio para a meta principal. A correlação indica que o sucesso da meta principal é influenciado caso a meta inicial seja atingida. Há três situações no elo de correlação que necessitam ser observadas:

a.1) Correlação entre duas metas concretas: a meta concreta principal possui a meta concreta como um subcomponente da tarefa-meio.

a.2) Correlação entre uma meta flexível e uma meta concreta: a meta concreta principal possui a meta flexível como um subcomponente da tarefa-meio.

a.3) Correlação entre uma meta concreta e uma meta flexível: a meta flexível principal possui a meta concreta como um subcomponente da tarefa-meio.

B) Contribuição de meta flexível para meta flexível: a relação de contribuição ocorre entre duas metas flexíveis de um mesmo ator. Uma das metas, chamada de meta inicial, pode contribuir de forma positiva (+), neutra (?) ou negativa (-) para a meta flexível principal. A contribuição afeta a meta flexível principal que é influenciada pela meta flexível inicial.

C) Dependência entre duas metas: a relação de dependência ocorre entre duas metas de atores diferentes. Ela representa a necessidade do suprimento (ou atendimento) de uma dependência entre dois atores. As dependências embutidas e não-representadas na relação podem ser uma tarefa, um recurso, uma meta flexível ou uma meta concreta, e são mapeadas posteriormente em outra etapa do método. A relação de dependência indica que o sucesso da meta principal (do *depender*) é influenciado pelo alcance da meta inicial (do *dependee*).

Apesar de não ser representada como transição há um quarto elo, representado por setas bidirecionais como extremos de uma linha tracejada (◄- - - -►).

Esse elo é uma ligação entre metas de atores diferentes e serve para representar a equivalência dessas metas.

D) Equivalência entre metas concretas ou entre metas flexíveis: sinaliza simplesmente que as metas flexíveis ou as metas concretas são equivalentes para atores diferentes.

Basicamente os elementos que compõem o Diagrama IP são os apresentados na Figura 11.

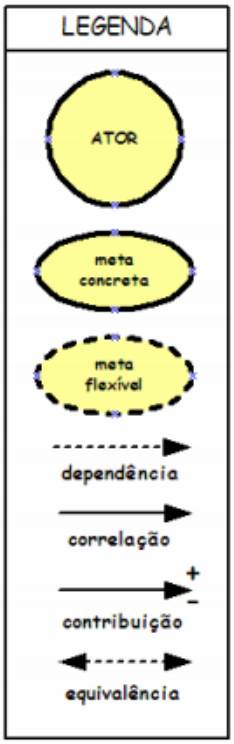


Figura 11: Elementos do Diagrama IP [Oliveira et al 2008].

A Figura 12 apresenta os elementos utilizados no Diagrama IP.

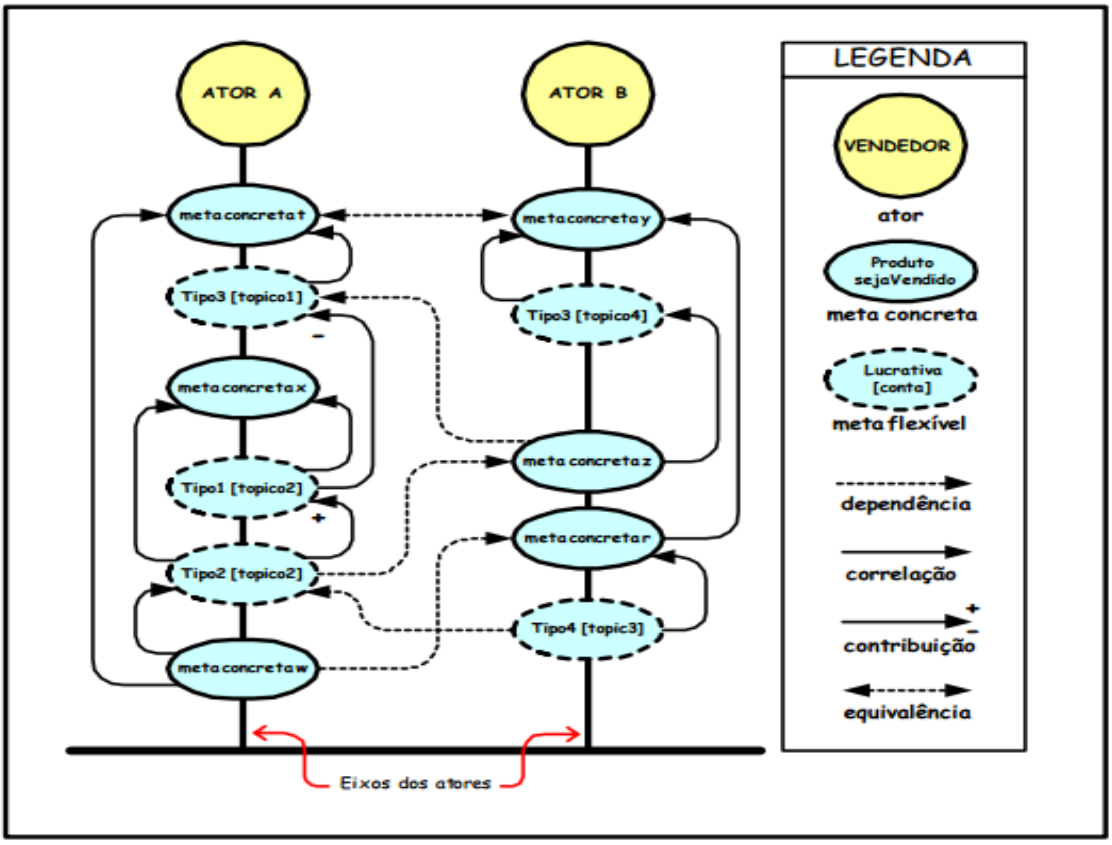


Figura 12: Tipos de elementos utilizados no Digrama IP [Oliveira et al. 2008].

A importância de criação de modelos a partir desses diagramas é, em um primeiro momento, tentar entender as metas e relações dos agentes de *software* envolvidos no método sistêmico e consequentemente tornar transparente a informação sobre as interações dos agentes no SMA que fará a análse dos RNFs. Seu uso anterior ao uso do modelo SR [Yu 1995] é justificado pela simplicidade de construção a partir de uma semântica que envolve apenas três tipos de notação de transição representados pela correlação, pela contribuição e pela dependência e assim poder dar maior ênfase à intencionalidade dos agentes antes de se construir de fato os seus detalhes internos de funcionamento.

Outra importante contribuição do uso do Diagrama IP é a possibilidade de se trabalhar sob o conceito de visões, ou seja, fazer recortes do do modelo intencional e permitir a visualização, por exemplo, apenas da interação de dois agentes [Oliveira et al. 2008].

2.7
Sistemas Multi-Agente e o Modelo BDI

Sistemas desenvolvidos com base em GORE são implementados levando-se em consideração que as metas sejam a representação do conhecimento de objetivos estabelecidas por atores organizacionais [Lamsweerde 2001]. Uma arquitetura que demonstra ser mais aderente às soluções orientadas à metas é a de SMA baseada na abstração de intencionalidade [Bratman 1999]. Braubach et al (2003) citam que um sistema multi-agentes intencional representa uma forma adequada de lidar com o raciocínio prático humano, simular a formação de metas, e interpretar os estados mentais humanos. Dentro dessa filosofia, o modelo BDI proposto por Bratman (1999) é o que proporciona melhor aderência ao propósito de metas uma vez que sua arquitetura considera que agentes inteligentes trabalhem de forma a simular o raciocínio humano.

O modelo BDI potencializa a representação das atitudes mentais a partir da criação de crença (*Belief*), desejo (*Desire*) e intenção (*Intention*). As crenças representam o que sabemos; os desejos, o que queremos; e as intenções, o que podemos fazer [Bratman 1999] [Rao 1996]. Essas características aplicadas a agentes inteligentes permitem a representação do conhecimento do agente sobre o contexto em que está inserido; as metas que necessitam atingir diante do contexto conhecido; e a tarefa ou conjunto delas que necessitam ser executadas pelo agente para que alcancem as metas estabelecidas.

Com isso, além de já atuarem de forma autonômica, comunicativa, social, com capacidade de aprendizado, com raciocínio e pró-atividade conforme proposto em [Wooldridge 2002] [Jennings e Wooldridge 2002], os agentes também passam a trabalhar a partir de uma arquitetura organizada e especializada para que possam atingir metas específicas ou comuns.

O *framework* JADEX (*JADE eXtension*) [Braubach et al. 2003] foi de desenvolvido de forma a suportar a arquitetura baseada no modelo BDI. JADEX possibilita o desenvolvimento de agentes a partir de uma arquitetura híbrida reativa/deliberativa. Nas características reativas estão as atitudes mentais informativas (crenças), motivacionais (desejos) e nas deliberativas as ações ou planos [Serrano e Leite 2011b]. Como o *framework* é produzido a partir do padrão *Foundation for Intelligent Physical Agents* (FIPA), padrão para comunicação entre agentes, ele permite com que os agentes construídos no modelo BDI sejam implementados e executados na plataforma *Java Agent Development Environment* (JADE) a partir de paradigmas de arquitetura semelhantes, por exemplo, a construção

de classes em linguagem Java para representar crenças e planos [Braubach et al 2003]. Apesar do código de execução ou acionamento dos agentes e seus métodos serem por características Java, toda sua arquitetura interna, bem como a especialização de crenças, desejos, intenções, tarefas, planos, mensagens são construídos a partir de XML em arquivos denominados *Agent Definition File* (ADF).

A arquitetura do SMA desenvolvido nesse trabalho seguiu o formalismo proposto em [Serrano e Leite 2011b] onde os autores associam o desenvolvimento do modelo intencional de agentes com o modelo BDI e em seguida para elementos da implementação em JADEX. A Figura 13 representa essa associação (mapeamento) onde elementos do modelo i* [Yu 1995] do modelo conceitual dos agentes é relacionado à proposta BDI e consequentemente em elementos físicos da camada JADEX que foram implementados em XML ou classes Java.

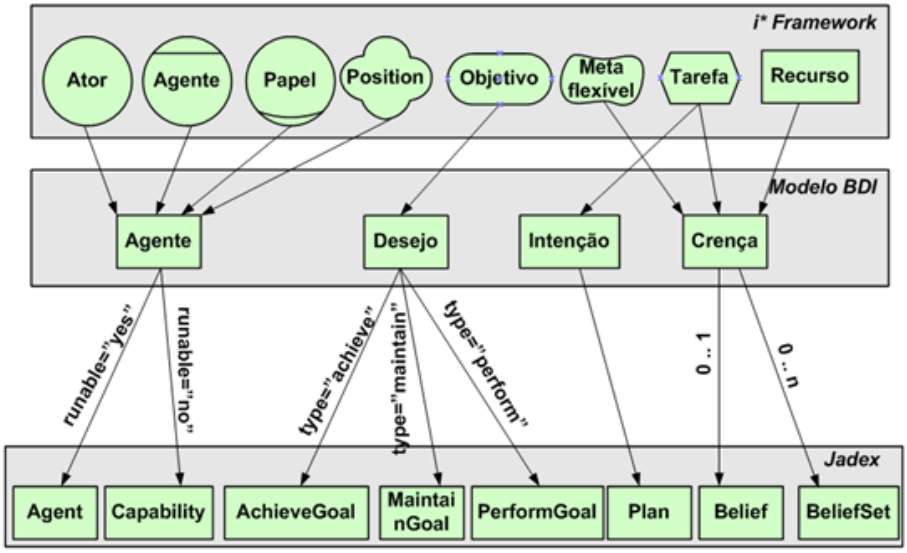


Figura 13: Mapeamento i* x BDI x JADEX [Serrano e Leite 2011b].

Os desejos foram traduzidos como objetivos e desenvolvidos ou mantidos de acordo com a *tag type* do modelo BDI, as intenções foram implementadas como planos com classes Java que estenderam os planos da plataforma JADEX [Braubach et al 2003], as crenças, com cardinalidades de 0 para 1 ou 1 para 1 foram traduzidas para *beliefs* JADEX, enquanto que crenças com cardinalidades de 0 para n e 1 para n foram traduzidas como conjuntos de *beliefset* JADEX, conforme proposto em [Serrano e Leite 2011b].

2.8
Considerações finais

É importante salientar que a fundamentação teórica teve como objetivo abranger aspectos conceituais das principais técnicas, ferramentas ou métodos utilizados no trabalho. Nesses aspectos foram consideradas teorias de GORE, RNF *framework,* GQM, GQO e padrões de RNFs que dão suporte ao que é construído enquanto método sistêmico. A construção do SMA, com agentes que atuam sobre o método para analisar RNFs baseados em catálogo pré-estabelecido, foi fundamentada sobre modelagem intencional, *framework* i*, painel de intencionalidades e BDI

3.
Método Sistêmico para Análise de Conformidade de RNF

Nesse capítulo é apresentado o método sistêmico para análise de RNF operacionalizado por SMA. Inicialmente, um contexto histórico da pesquisa é apresentado para que possa ser dada a devida atenção para a evolução da construção dos agentes e da pesquisa até à proposta final do método.

3.1
Contexto Histórico da Pesquisa

A pesquisa iniciada no início do ano de 2010 teve como objetivo estabelecer etapas de desenvolvimento para que a construção de uma política de análise de RNFs pudesse ser satisfatória. Um dos primeiros desafios foi a construção de arcabouço de um SMA que poussuísse agentes que atuam de forma independente e com tarefas distintas para que pudessem ser evoluídos conforme o desenvolvimento da pesquisa. Em um primeiro momento a pesquisa indicou que a arquitetura do SMA seria suficiente para estabelecer uma política para a análise dos RNFs, mas com o seu desenvolvimento foi necessário estabelecer um método sistêmico em que o SMA é o mecanismo automático para a captura de artefatos, sua análise e disgnóstico baseados no que está estabelecido no método.

A versões do SMA, bem como o método desenvolvido nesse trabalho passam a ser descritos nas seções posteriores.

3.1.1
Primeira Versão

A primeira versão de agentes na análise de RNFs foi um esforço de análise de viabilidade de agentes monitores que atuam de forma conjunta para captura de traços de execução de um *software*. Após a captura dos dados esses eram comparados com uma lista de regras estabelecidas no CTS. Tal versão desenvolvida de forma *ad-hoc* não contemplava uma sistematização de política de monitoração e sim uma modelagem intencional da arquitetura da política de funcionamento dos agentes, isso foi percebido após a evolução da pesquisa. Seus resultados foram publicados em [Leal et al. 2013a] e [Leal et al. 2013b].

Foi construído um SMA que efetuava coleta de dados dos traços de execução gerados a partir de agentes do *software Lattesscholar* (LS) [Lattesscholar 2013]. O LS é um SMA independente para contagem de citações científicas, que combina os serviços do sítio do *Lattes* [Lattes 2013] e do *Google Scholar* [Google Scholar 2013].

Sua execução é baseada na pesquisa inicial pelo nome do autor no sítio do *Lattes* e posteriormente uma pesquisa pela relação de artigos no *Google Scholar* com o objetivo de contabilizar citações sobre cada publicação do autor. A vantagem de seu uso está na consulta de citações a partir do título da publicação, o que difere, por exemplo do *Publish and Perish* [Publish and Perish 2013], do *Microsoft Academic Search* [MASearch 2013] e do *Google Scholar Citation* [ScholarCitation 2013], uma vez que suas pesquisas são realizadas a partir do nome do autor podendo acarretar inconsistências de contagem devido a erros causados por homônimos, abreviaturas, nomes muito extensos e sinônimos em nomes muito pequenos.

A partir da identificação do *software* a ser monitorado foi elaborado um modelo intencional baseado em i* (i-estrela) [Yu 1995] que sugere agentes de monitoração para analisar traços de execução em um *software*, no caso aplicados sobre o LS.

Os traços de execução do LS são registros gerados em *log* a cada iteração de execução do LS a partir de pesquisas por nome do autor na plataforma *Lattes* e o seu número de citações a partir do Google Scholar. Os traços de execução possuem registros de cada iteração do LS a partir do armazenamento das ações dos agentes, como *ReceberUrlPesquisador*, *SolicitarObrasCurriculo*, entre outras. Além disso, os traços contém o número de citações por obra do autor. Para reforçar o explicado anteriormente, o LS é um *software* independente do sistema de monitoração e dele são utilizados apenas os traços de sua execução.

Os traços de execução são monitorados pelos agentes do SMA e as evidências encontradas são comparadas ao catalogo de transparência para indicar conformidades e não conformidades com o mesmo. Os quatro agentes no SMA possuem ações bem definidas e especializadas. Nessa versão, cada agente trata de forma distinta suas tarefas, recursos, desejos e objetivos. Suas discriminações são apresentadas como em [Leal et al. 2013a] e [Leal et al. 2013b]:

- MONITOR: o agente monitora constantemente os traços de execução gerados pelos agentes do LS, com o objetivo de captar os seus registros. Os registros dos traços de execução do LS são gerados pela própria aplicação, inicialmente ele verifica se já há traços de execução do agente ANALISADOR com resultados de avaliações. Para isso ele acessa um arquivo com o registro de rastros (traços de execução) do ANALISADOR referente à memória de avaliações anteriores, tais rastros são registrados a partir de uma estratégia de proveniência (*provenance*) [Miles et al. 2005], [Miles et al. 2005], [Pinheiro et al. 2003]. O traço de execução do ANALISADOR, ou "memória de monitoração", contém o registro de *tags* das regras de monitoração baseados no modelo canônico e a correlação dessas *tags* com as alternativas do CTS de monitorações anteriores.

Dessa forma, o MONITOR decide por encaminhar diretamente ao ANALISADOR as respostas de conformidade de transparência sem ter a necessidade de transcorrer por todo o processo de monitoração, ou seja, passando por outros agentes como o CANONIZADOR e o CONSOLIDADOR.

Caso o agente não encontre os traços de execução dos rastros do ANALISADOR ele procede com o processo completo e envia uma mensagem ao agente CANONIZADOR informando sobre a captura e disponibilidade do arquivo.

- CANONIZADOR: tem por objetivo receber os registros dos traços de execução e transformar as informações em um padrão que possa ser consolidado. O CANONIZADOR utiliza endereços de posição nos traços de execução, como também ocorrências delimitadas por algum trecho de texto que evidencie o que está sendo canonizado. Por exemplo: o nome de um agente que está sendo monitorado pode ocorrer após a descrição Agente name=, inicia uma posição após o sinal de igual (=) e finaliza uma posição a frente do caractere @. Ou seja, o agente interpreta o arquivo dos traços de execução, padroniza sua leitura e disponibiliza os registros dos traços de execução em classes na memória, para que possam ser utilizadas por outros agentes.

- CONSOLIDADOR: verifica a partir da estrutura canonizada os registros de conformidades e não conformidades e disponibiliza ao ANALISADOR recursos de conformidades que possam ser verificados de acordo com o CTS.

- ANALISADOR: tem por finalidade verificar se os registros padronizados e extraídos pelo CANONIZADOR correspondem às exigências dos Grupos, Questões e Alternativas normatizadas para um atributo de transparência. O agente verifica quais são as exigências da regra para que uma alternativa seja positivada.

Por exemplo, para ser atendida uma alternativa para a questão:

Questão: As redes de interação são mapeadas?

A alternativa escolhida foi:

Alternativa: é explicitado o nome dos agentes envolvidos nas comunicações, nome do *software* e sistemas externos, serviços (webServices) e o registro de suas comunicações, passagem de parâmetros, troca mensagens.

Com a alternativa definida o agente ANALISADOR busca nos registros gerados pelo CANONIZADOR, o registro do nome dos agentes ou de um agente envolvido na troca de mensagens de *Send* e *Receive*. Assim o agente deve marcar a alternativa como satisfeita, ou positiva. O ANALISADOR guarda o traço de execução com o objetivo de manter uma memória das *tags* avaliadas nos *software* monitorados e sua correlação com o CTS.

3.1.2
Segunda Versão

A segunda versão dos agentes foi construída para eliminar algumas deficiências da primeira. Essas alterações perpassaram por:

a) alteração do formato de mensagens trocadas na comunicação entre os agentes: a comunicação, anteriormente feita a partir de passagem de parâmetros através de métodos das classes da linguagem Java, passaram a ser feitas por estruturas XML, escolhida por ser uma linguagem universal de troca de mensagens, por exemplo, entre *webservices*;

b) leitura dos traços de execução: na primeira versão eram lidos traços de execução gerados a partir de formatos ASCii[1] e passaram a ser lidos apenas por conteúdos gerados em estruturas XML;

c) outra importante alteração foi o uso de *tags* lidas das estruturas de comunicação dos agentes e também dos traços de execução baseados em XML: uma vez utilizadas as *tag* XML entendemos que seu uso poderia facilitar a criação de dicionários de termos que pudessem auxiliar o processo de construção de conhecimento dos agentes. *Tags* conhecidas e registradas no domínio dos agentes passaram a compor uma base de conhecimento para futuras análises de arquivos com traços de execução de *software* variados. *Tags* desconhecidas também passaram a ficar registradas para que pudessem ser analisadas por agente humano e relacionada às regras de Transparência.

Nessa segunda versão o SMA tornou-se mais dinâmico ao incorporar nos agentes a linguagem XML, tanto como insumo de leitura como de parâmetros de troca de mensagens entre os agentes.

A partir dessa dinamização foi possível estabelecer um método em que o modelo conceitual dos RNFs, bem como suas operacionalizações, fossem configuradas em XML para que fosse estabelecido um catálodo de RNF. O catálogo a partir de então poderia ser um *baseline* com estruturas de padrões de RNF, com características e relações, que é utilizado na comparação com o artefato de *software* para análise de conformidade.

[1] *American Standard Code for Information Interchange*: http://pt.wikipedia.org/wiki/ASCII

3.1.3
Terceira Versão

A terceira e última versão foi evoluída após a criação do método, uma vez que foi necessário alterar as tarefas dos agentes CANONIZADOR, CONSOLIDADOR e ANALISADOR. Os agentes tiveram suas tarefas alteradas para ficarem mais aderentes ao que é proposto no método, sua nova arquitetura será discutida na seção 4.4 Arquitetura do SMA.

O método, o catálogo, bem como as arquiteturas utilizadas como solução para análise de confirmidade de RNFs implementados em *software* passam a ser descritos nas seções subsequentes.

3.2
Método Sistêmico para Análise de RNFs

A análise de RNFs implementados em *software* foi uma problemática levantada na pesquisa desde seu início. A construção do SMA, em sua primeira versão, já tinha um intuído de direcionar a pesquisa para a elaboração de um método que pudesse explicitar de forma detalhada como um mecanismo de *software* poderia analisar a conformidade de RNFs. Mesmo que de forma *ad-hoc*, a primeira versão do SMA [Leal et al. 2013a] foi concebida baseada em características descritas de forma organizada a partir de dados orientados à metas do RNF de Transparência.

A evolução do SMA e a consolidação da necessidade de se estabelecer um método sistêmico, baseado em passos e arquiteturas que lhe dão suporte, foi o ponto principal para se consolidar a pesquisa. Tal consolidação deu origem ao método que será proposto nas seções posteriores.

3.2.1
Modelo Central

Para apresentar o método sistêmico é importante que um modelo central seja exposto para explicitar o contexto central do método, além de deixar explícitas as contribuições a partir do detalhamento das tarefas principais expostas no modelo central. A partir daí as estruturas são apresentadas em blocos para facilitar a leitura e compreensão. A Figura 14 apresenta, na linguagem SADT [Ross 1997], o modelo central do método sistêmico para análise de RNF. Sua operacionalização é dada a partir de cinco atividades:

(A1) Criar SIG (proposto a partir de Chung (2000);

(A2) Definir patterns (proposto a partir de Supakkul et al. (2010) e Serrano et al. (2010);

(A3) Configurar XML;

(A4) Configurar *software* (no caso *software* a ser analisado, cujo papel de configuração fica sob responsabilidade de seu proprietário);

(A5) Operacionalizar agentes (propriamente a execução da análise de artefato ou traços de execução do *software*. Nessa tese artefatos e traço de execução são tratados simplesmente como artefatos, exceto nos estudos de caso, uma vez que os traços de execução geralmente estão armazenados em arquivos).

O método é apresentado a partir de modelos SADT para deixar explícitas as atividades de sua operacionalização. Foram considerados grandes grupos de atividades na construção do SADT (visão macro) e atividades de menor granularidade foram tratadas em quadros como passos no detalhamento de uma atividade de maior nível. Nos SADT foram incorporadas linhas tracejadas verticais apenas para delimitar atividades e foram inseridos na parte inferior (rodapé) a origem da fundamentação teórica utilizada. Nesse caso, é indicado a referência bibliográfica do autor que propos o método ou técnica utilizada em cada passo. A construção do método proposto nessa tese é portanto uma conjunção de métodos e técnicas propostas por autores da literatura da área de ES no que diz respeito a primeira parte da modelagem conceitual dos RNFs, ou seja o que se refere às atividades A1 e A2, a partir de então tais abordagens são associadas e operacionalizadas por outras técnicas inseridas nesse trabalho de tese.

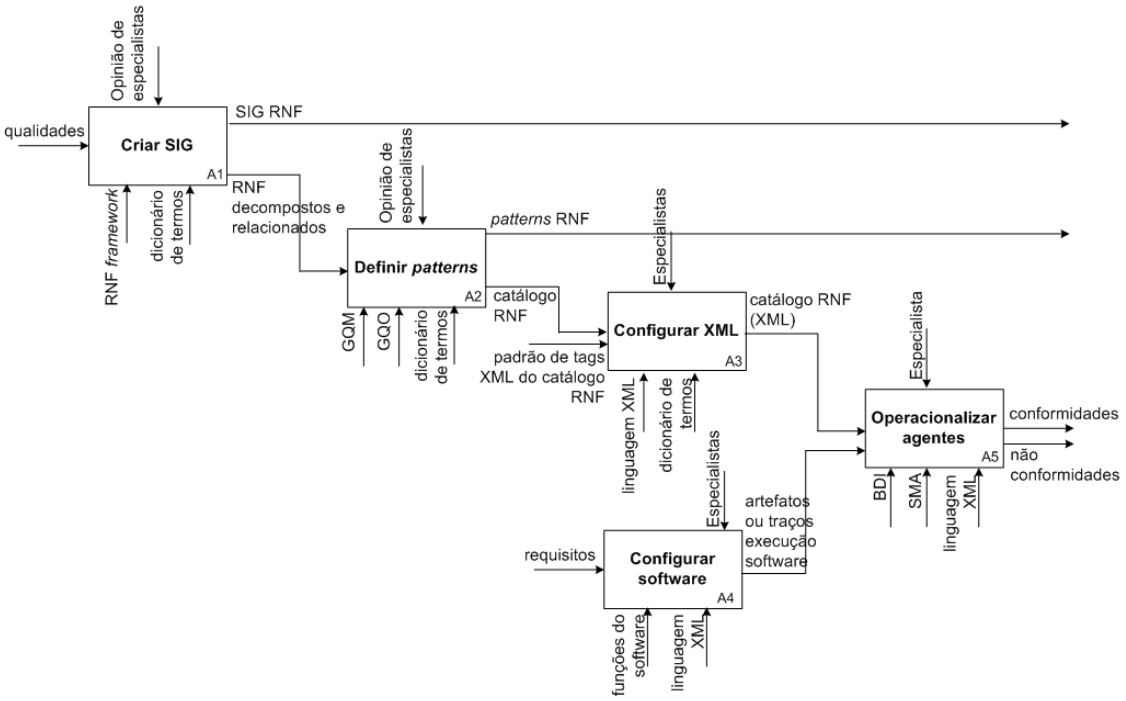


Figura 14: Visão Macro do Método.

1. *Criar SIG (A1)*: consiste em definir de forma sistemática a decomposição de uma qualidade em outras qualidades que a caracterizem, sua hierarquização (a partir de uma das qualidades colocada em evidência, pois trata-se da qualidade que é verificada a partir do modelo) e seus elos de relacionamento, sejam eles de contribuição positivos ou negativos. Tal estrutura é baseada na proposta do *NFR framework* de Chung et al. [2000]. Esses fatores auxiliam na criação de uma semântica rica para entendimento das qualidades e suas relações e o seu resultado são: RNFs decompostos e relacionados; e SIG do RNF. Chung et al. (2000) estabelece que tais qualidades passam a ser metas a serem cumpridas, ou *softgoals*. Na ER, o uso da palavra *goal* significa que o método é *goal oriented*, ou seja, trabalha na captura dos requisitos do sistema priorizando a elicitação do porquê (*why*) antes de trabalhar e definir, através dos modelos, o que (*what*) o *software* deverá fazer [Oliveira et al. 2008]. A importância da orientação à meta para o método está no fato de que as qualidades organizadas passam a ser *softgoals* a atingir em um processo de avaliação. Esses *softgoals* devem ser operacionalizados através de procedimentos funcionais que contribuam para sua satisfação, usado com a mesma ideia de *satisfice* (bom o suficiente) [Simon 1996] para solucionar um problema, e essa satisfação a contento [Cunha 2007] é propagada a partir das relações

de contribuições entre os demais *softgoals* estabelecidos no modelo SIG. Essa atividade é composta por três atividades:

a. Elicitar qualidades: propõe a identificação de um RNF principal e suas decomposições;
b. Relacionar qualidades: objetiva relacionar os RNFs identificados/decompostos ao *softgoal* principal;
c. Verificar estrutura: sugere que as atividades anteriores sejam verificadas. As ponderações ocorridas devem ser verificadas pela atividade onde foi detectada a incoerência.

2. *Definir Patterns (A2)*: a atividade tem por objetivo estabelecer maior conhecimento a respeito do *softgoal* objeto da análise a partir de outros *softgoals*, de elos de contribuição, de grupos (categorias), questões e alternativas (operacionalizações/procedimentos funcionais), conforme sugerido no trabalho de [Serrano e Leite 2011a]. Esses elementos em conjunto permitem a criação de uma estrutura baseada em uma visão *TOP-DOWN* onde o *softgoal* passa a ser o elemento principal e em uma camada mais baixa estão localizadas as operacionalizações. A importância da aplicação do método está no refinamento a partir do *softgoal* principal e suas decomposições até as operacionalizações que possam satisfazê-los a contento [Cunha 2007]. A partir desses desdobramentos há a criação de um catálogo do *softgoal* a partir de uma coleção de padrões denominado padrões de RNFs (*Objetive patterns*, *Alternative patterns, Selection patterns*, Problem Patterns [Supakkul et al. 2010] e *Question Patterns* [Serrano et al. 2010]).

   A atividade é sub-dividida em:

   a. Descrever questões: a atividade tem por objetivo a identificação, descrição e relacionamento de questões que possam responder, com intuito de avaliação*;*
   b. Operacionalizar questões: objetiva identificar e relacionar práticas da ES que possam servir de alternativas de respostas para as questões;
   c. Verificar estrutura: sugere com que as atividades anteriores sejam verificadas. As ponderações ocorridas devem ser verificadas pela atividade onde foi detectada a incoerência.

3. *Configurar XML (A3)*: a partir das duas atividades anteriores o processo de operacionalização do método necessita que os requisitos estabelecidos, bem como suas operacionalizações, sejam configurados para que possam

ser utilizados como insumos para o SMA. Essa configuração padroniza estruturas XML, devidamente sistematizadas, hierarquizadas e relacionadas, para que os agentes possam estabelecer um *baseline* estruturado com um método e avaliação.

O arcabouço do XML é apresentado no APÊNDICE A e nele estão registradas as estruturas para a definição de *patterns* do RNF decomposto e relacionado a partir das atividades A1 e A2, dessa forma é estabelecido um padrão para tratamento de conformidades e não conformidades entre um *software* a ser analisado e as regras definidas no catálogo.

O uso do arcabouço XML tem como objetivo possibilitar o reuso do conhecimento sobre RNF [Leite et al. 2005]. As estruturas em XML dos padrões são mantidas, independente do *software* analisado, e o ponto de variabilidade encontra-se na configuração do arcabouço XML com a customização dos conteúdos das *tags* dos padrões dependendo do domínio de aplicação do catálogo.

Por exemplo, para o RNF de Transparência [Leite e Cappelli 2010] há uma customização que envolveria o Universo de Informação (UdI) [Leite et al. 1997] desse domínio, enquanto que para o RNF de Segurança seria necessária a análise de metas e operacionalizações de seu UdI. A atividade é composta por atividades que tem por objetivo principal estruturar o catálogo do RNF em um padrão XML:

a. Configurar *objetive patterns*: objetiva configurar o RNF principal, suas decomposições e relacionamentos (padrões seleção);
b. Configurar *question patterns*: objetiva configurar os padrões questões já relacionados às decomposições dos *objetive patterns*;
c. Configurar *alternative patterns*: objetiva configurar as operacionalizações (alternativas/respostas) para as questões identificadas para cada decomposição;
d. Configurar variáveis e sinônimos: nessa atividade há como foco a identificação de variáveis e seus sinônimos que servirão de marcadores para configuração do artefato, ou quando necessário, também dos artefatos do *software* que não são gerados sobre um padrão pré-determinado. Quando há padrão, a configuração pode ser feita apenas no catálogo XML. A atividade consiste, primeiramente na identificação de variáveis e seus sinônimos para cada operacionalização e em um segundo momento na sua configuração em XML;

e. Verificar estrutura: consiste em revisar o catálogo configurado para identificar sua consistência, as observações de avaliação devem ser revistas pelos passos anteriores.

4. *Configurar software (A4)*: a importância da atividade consiste em configurar o artefato, ou quando necessário (traços sem padrão pré-determinado), os traços de execução do *software* que são analisados. Os artefatos devem estar com marcações compatíveis com o padrão estabelecido no XML de configuração do catálogo dos *patterns* de RNF. A atividade é responsabilidade do produtor do *software* e está representada na regra do catálogo de forma a deixar explícita a necessidade de sua execução enquanto atividade.
5. *Operacionalizar agentes (A5)*: o objetivo da atividade é prover um mecanismo automático de operacionalização da politica de análise a partir de agentes baseados em uma abordagem de SMA. Os agentes são importantes uma vez que trabalham fundamentalmente como sistemas de *software* que representam os atores em um determinado contexto, são autônomos, pró-ativos e sociais (colaboram entre si, se comunicam e interagem), possuem capacidade de aprendizado e de adaptação [Wooldridge 2002]. Essas características são importantes em um contexto de análise uma vez que a complexidade dos elementos envolvidos é peculiar. Os agentes trabalham com o contexto do catálogo do RNF parametrizado, conforme sugerido em A3, desde a caracterização e o estabelecimento das relações iniciais dos RNF até os procedimentos funcionais para satisfazê-los a contendo [Cunha 2007]. A atividade é sub-dividida em:
    a. Capturar artefato: objetiva a captura do artefato do *software* a ser analisado;
    b. Analisar: tem por finalidade analisar os artefatos a fim de disponibilizá-lo para avaliação;
    c. Apresentar Resultados: consiste em apresentar os resultados de conformidades, a partir da comparação do artefato, comparando-o a uma estrutura pré-definida no catálogo XML de RNF.

As sub-seções subsequentes tratam do detalhamento de cada atividade do modelo central (A0). As derivações do modelo central serão compostas a partir de:

- detalhamento de cada atividade (*rationale* de A1, A2, A3, A4 e A5);

- de um quadro com o detalhamento de cada sub-atividade, seus passos e significados. O quadro foi desenhado para facilitar a apresentação do conhecimento

sobre o detalhamento, e é subdividido em partes: 1º O título da atividade, o mesmo representado dentro do SADT; 2º Um resumo sobre o *rationale* da atividade; 3º A subdivisão da atividade em passos de execução, enumerados sequenciamente a partir da numeração da atividade de origem, bem como um resumo do *rationale* de cada passo; 4º Uma observação sobre o contexto do quadro. A observação pode trazer alguma complementação teórica com relação a algum item citado no texto, o detalhamento de uma caracterização técnica utilizada no texto ou simplesmente não conter nenhuma descrição, por ser entendido que os passos ou o *rationale* contém por si só descrições suficientes. Em 5º está um uma coluna a indicação numérica da atividade, numerada sequenciamente a partir da Atividade Central no SADT.

3.2.2
Detalhamento de Atividade - Criar SIG (A1)

A Figura 15 apresenta o detalhamento da atividade *Criar SIG* (A1) presente no modelo central. Nela é possível perceber as três sub-atividades apresentadas como: A1.1 Elicitar qualidades, A 1.2 Relacionar qualidades e A 1.3 Analisar estrutura.

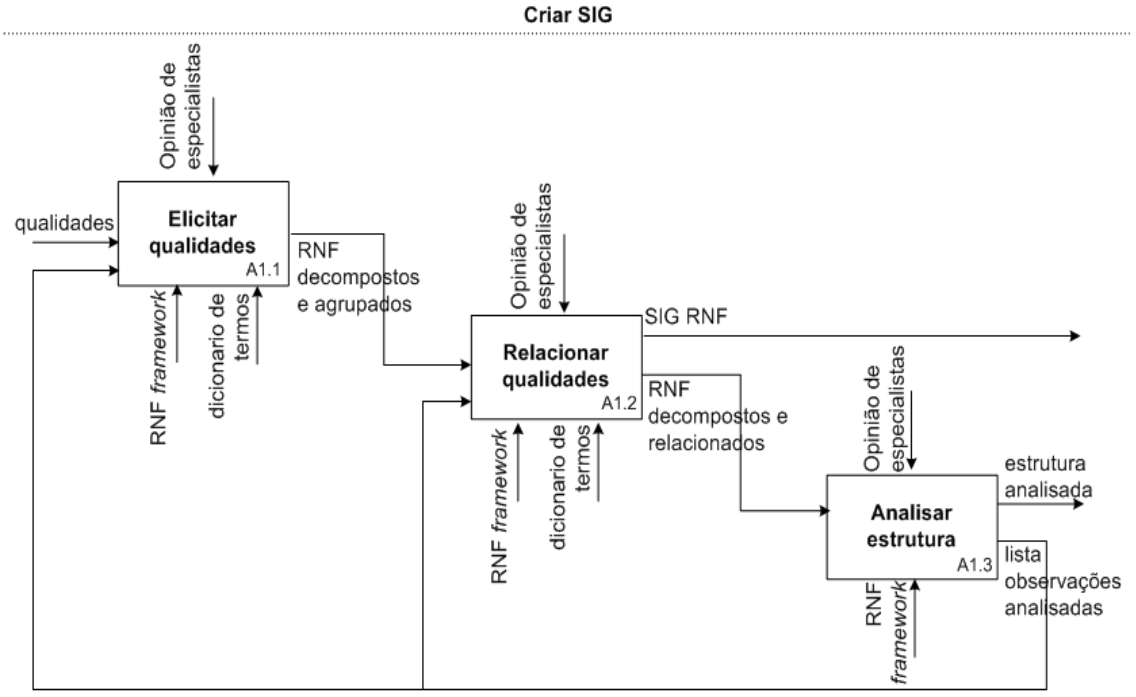


Figura 15: Decomposição da A1: Criar SIG.

Os quadros apresentam as informações e passos de execução de cada sub-atividade, bem como seus significados. Há também espaço para a descrição de observações sobre a sub-atividade.

A1.1	**Elicitar qualidades** (identificação das qualidades)
	Resumo: A atividade faz-se necessária para a escolha de uma qualidade principal (*softgoal* principal), que é objeto da análise, para que o mesmo possa ser desmembrado em outros *softgoals* que o qualifiquem. A definição do *softgoal* principal deve ser estabelecida como aquele que se pretende avaliar, sua decomposição feita a partir de outras qualidades que melhor o definam em termos semânticos. O RNF *framework de* Chung et al. (2000) é uma importante ferramenta para tal atividade uma vez que propõe a decomposição de RNFs, as priorizam e organizam.

Passos	Resumo
1.1.1. Identificar o RNF principal;	A partir das diversas qualidades de entrada elencar aquela que seja objeto da análise a fim de transformá-la em RNF principal;
1.1.2. Decompor RNF;	Decompor o RNF principal em diversas outras qualidades com o objetivo de caracterizá-lo em menores porções para um maior nível de detalhamento e semântica;
1.1.3. Estabelecer *softgoals*;	Estabelecer os RNFs listados como objetivos a serem atingidos. Como são requisitos de qualidade, aqueles com maior grau de subjetividade de julgamento sob o ponto de vista do avaliador, esses devem ser considerados como *softgoals*.
1.1.4. Qualificar *softgoals*.	Qualificar cada um dos *softgoals* a partir de taxonomias, ou seja, a identificação dos significados de cada elemento.
Observação:	

A1.2	**Relacionar qualidades** (traçar elos de contribuição)
	Resumo: Os relacionamentos traçados nessa atividade são utilizados para criar elos de contribuição entre os *softgoals* encontrados e organizados. Com

<table>
<tr><td></td><td colspan="2">isso pretende-se dar definição ao softgoal principal a partir de uma semântica rica em termos de suas decomposições (elaboradas a partir da A.1.1) e suas relações. O softgoal principal é colocado em uma posição de destaque para que se perceba as contribuições, positivas ou negativas, das qualidades posicionadas abaixo. É importante agrupar softgoals a fim de se ter maior organização de qualidades de menor porção que possam dar significado ao elemento do grupo e esse por sua vez ao softgoal principal.</td></tr>
<tr><td colspan="2">Passos</td><td>Resumo</td></tr>
<tr><td colspan="2">1.2.1. Evidenciar agrupadores;</td><td>Colocar em evidência sofgoals que sejam agrupadores de qualidades que influenciam o softgoal principal e que por ventura tenham mais de uma qualidade como detalhe de menor nível;</td></tr>
<tr><td colspan="2">1.2.2. Identificar detalhes;</td><td>Identificar qualidades (softgoals folha) que sejam detalhamentos de softgoals agrupadores;</td></tr>
<tr><td colspan="2">1.2.3. Relacionar detalhes a seus agrupadores;</td><td>Relacionar aos agrupadores, a partir de elos de contribuição positiva do tipo Help ou a partir de elos de contribuição negativa do tipo Hurt, os softgoals de mais baixo nível;</td></tr>
<tr><td colspan="2">1.2.4. Relacionar agrupadores;</td><td>Relacionar aos agrupadores ao softgoal principal a partir de elos de contribuição positiva do tipo Help ou elos de contribuição negativa do tipo Hurt;</td></tr>
<tr><td colspan="3">Observação: o framework também possui representações para os tipos de elo de contribuição, tais como: a) Break: provê contribuição negativa suficiente para que a característica superior não seja atendida; b) Hurt: provê contribuição negativa parcial para não atendimento da característica superior; c) Unknown: provê contribuição porém não se sabe se negativa ou positiva; d) Help: provê contribuição positiva parcial para atendimento da característica superior; e) Make: provê contribuição positiva suficiente para que a característica superior seja atendida [Chung et al. 2000]. Por medidas de representação semântica o Grupo ER PUC-Rio trabalha com as ligações entre softgoals de diferentes grupos a partir de elos Some+ ou Some- o qual significam, respectivamente, contribuição positiva e negativa.</td></tr>
</table>

A1.3	**Analisar estrutura** (Analisar modelo)
	Resumo: Os RNFs, suas decomposições e relacionamentos necessitam ser

	analisados. Nessa atividade há o objetivo de se rever o modelo criado, revisar cada detalhe e deixá-lo revisto para as atividades seguintes.
Passos	Resumo
1.3.1. Revisar qualidades;	Colocar em evidência *sofgoals* que sejam agrupadores de qualidades que influenciam o *softgoal* principal e que por ventura tenham mais de uma qualidade como detalhe de menor nível;
1.3.2. Revisar relações;	Identificar qualidades (*softgoals* folha) que sejam detalhamentos de *softgoals* agrupadores;
1.3.3. Analisar relações.	Verificar relações a partir da análise de especialistas e *stakeholders* do domínio.
Observação:	

Ao final do processo o resultado produzido é uma combinação de qualidades relacionadas em um modelo visual, RNF *framework* Chung et al (2000), conforme discutido na seção 2.2 do Capítulo 2 Fundamentação Teórica.

3.2.3
Detalhamento de Atividade - Definir patterns (A2)

O refinamento dos *softgoals* contribui para fortalecer a semântica da representação dessas qualidades. Nessa atividade, um conjunto de outras características é incorporado ao modelo. O refinamento dos *softgoals* implementado na atividade anterior fez com que o *softgoal* principal fosse decomposto e elos de contribuição foram estabelecidos. As características nessa etapa sugerem a construção de categorias de questões (perguntas), questões que possam guiar as operacionalizações dos *softgoals* para que os mesmos possam ser satisfeitos a contento, bem como as relações entre esses elementos. Dessa forma, completa-se um ciclo de criação de padrões de RNFs.

Na Atividade A1 são representadas as qualidades, metas a serem atingidas para satisfazer ao RNF principal. Nessa atividade A2 são representadas as operacionalizações que devem ser selecionadas e construídas para necessidades específicas.

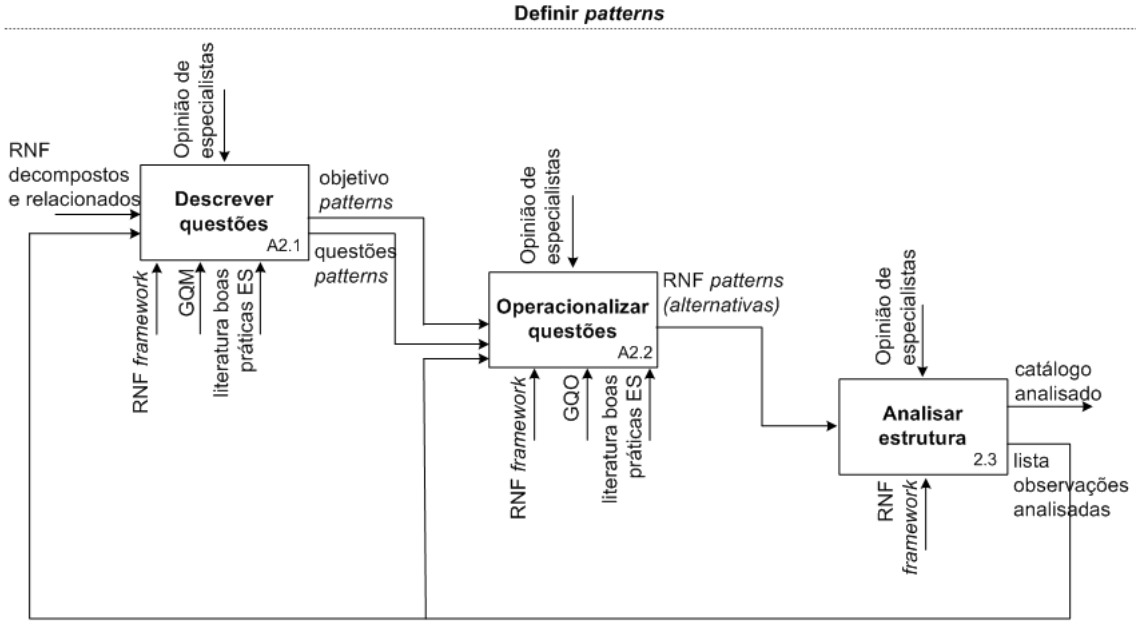


Figura 16: Decomposição da A2: Definir patterns.

A2.1	**Descrever questões** (descrever questões para *softgoals* folha)
	Resumo: Baseado na ideia de GQM [Basili 1992], a atividade objetiva a criação de questões que possam servir de meios para avaliação dos objetivos traçados, no caso *sofgoals.* As questões devem ser categorizadas e agrupadas de acordo com características intrínsecas dos *softgoals* de que fazem parte.

Passos	Resumo
2.1.1. Elicitar questões;	Elicitar questões que possam estabelecer parâmetros para avaliação da satisfação dos *sofgoals*;
2.1.2. Estabelecer categorias;	De acordo com a lista de questões, verificar aquelas que possuem características similares, basicamente encontradas a partir de características intrínsecas do *softgoal*, para que as questões possam ser agrupadas;
2.1.3. Agrupar questões;	Agrupar questões por categoria;
2.1.4. Vincular categorias;	Vincular as categorias aos *sofgoals* folha, aquelas de nível mais baixo no grafo do SIG;
2.1.5. Definir *patterns* Objetivos;	Descrever *patterns* identificação e decomposição como *patterns* Objetivos. Primeiramente com a identificação de um *softgoal* vértice, o qual será objeto de análise, e em segundo momento sua decomposição em outros *softgoals*.

2.1.6. Definir *patterns* Questões.	Descrever as Questões como *patterns* sem deixar de relacioná-las às suas categorias e aos *patterns* Objetivos.
Observação:	

A2.2	**Operacionalizar questões** (descrever operacionalizações para *questões*)
	Resumo: As operacionalizações são feitas a partir da proposta de Serrano e Leite (2011a) que adaptam o GQM [Basili 1992] para o método chamado GQO [Serrano e Leite 2011a]. O método propõe responder às questões dos *softgoals* a partir de boas práticas (métodos, técnicas, ações, procedimentos e ferramentas) da ES. O GQO permite a criação de uma maior granularidade a fim de proporcionar maior significado para as questões e lista possíveis operacionalizações para os *softgoals*.

Passos	Resumo
2.2.1. Elicitar operacionalizações;	Elicitar operacionalizações para uma dada questão;
2.2.2. Relacionar operacionalizações;	Relacionar as operacionalizações com a questão;
2.2.3. Avaliar operacionalização;	Avaliar se uma operacionalização pode ser alternativa para responder as questões de outras categorias;
2.2.4. Relacionar operacionalizações a outras questões;	Relacionar as operacionalizações com as questões de outras categorias;
2.2.5. Definir *patterns* Alternativas.	Descrever as Alternativas como *patterns* sem deixar de relacioná-las aos *patterns* Questões.
Observação: Uma alternativa listada e relacionada a uma ou mais questões de um *softgoal* pode ser respostas para questões de outro *softgoal*. Dessa forma, é necessária uma avaliação sobre alternativas incluídas e seu potencial de impacto nesses outros *softgoals*. Caso isso ocorra, é importante estabelecer elos de contribuição entre essa alternativa e as questões desses outros *softgoals*.	

A2.3	**Analisar estrutura** (Analisar modelo)
	Resumo: Os RNFs, suas decomposições e relacionamentos, necessitam ser

	verificados. Nessa atividade há o objetivo de se rever o modelo criado, revisar cada detalhe e deixa-lo verificado para as atividades seguintes.
Passos	Resumo
2.3.1. Revisar qualidades;	Colocar em evidência *sofgoals* que sejam agrupadores de qualidades que influenciam o *softgoal* principal e que por ventura tenham mais de uma qualidade como detalhe de menor nível;
2.3.2. Revisar relações;	Identificar qualidades (*softgoals* folha) que sejam detalhamentos de *softgoals* agrupadores;
2.3.3. Analisar relações.	Analisar relações a partir da análise de especialistas e *stakeholders* do domínio.
Observação:	

3.2.4
Detalhamento de Atividade - Configurar estrutura XML (A3)

As regras do método são baseadas em um catálogo do RNF configurado a partir de uma estrutura XML, conforme apresentada no APÊNDICE A. Tal estrutura é explicada em detalhes na seção Arquitetura, mas para sua configuração o método propõe as atividades conforme apresentada na Figura 17.

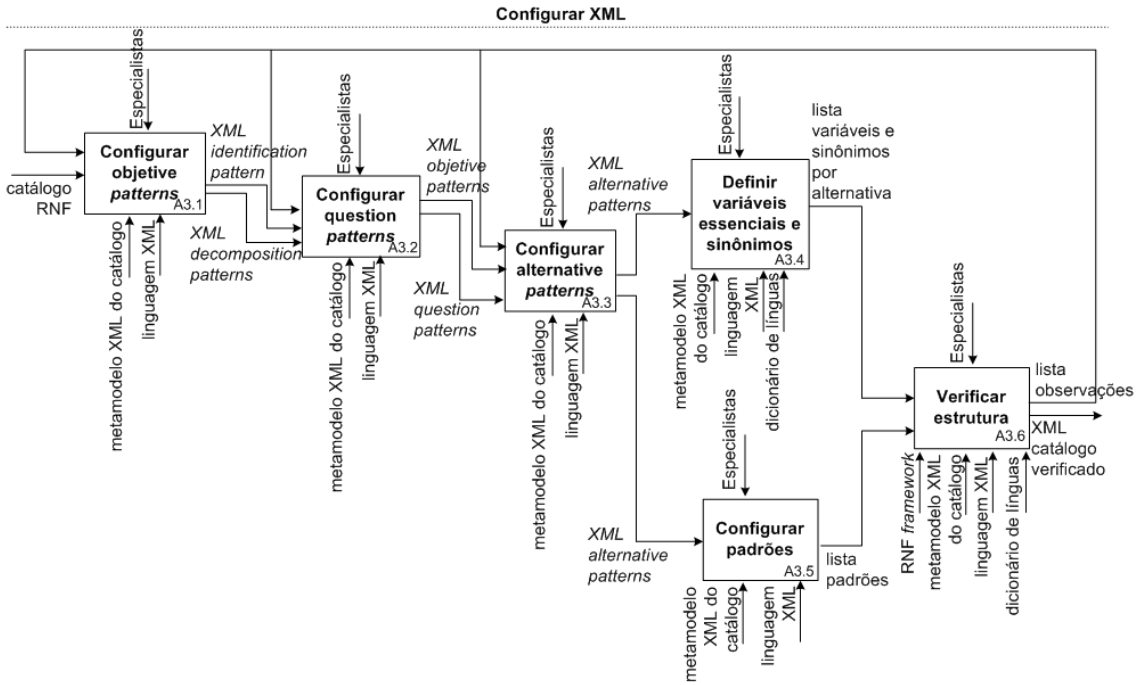


Figura 17: Decomposição da A3: Configurar XML.

A3.1	**Configurar *Objetive patterns*** (estruturas XML)
	Resumo: A atividade consiste em criar uma estrutura XML com decomposições e relações entre as qualidades estabelecidas no SIG do RNF proposta na atividade A1. A estrutura é criada nesse momento e evoluída a partir das outras atividades do modelo central do método. Ao final ela se torna uma estrutura para configuração do SMA, definida como catálogo XML RNF, e servirá de regra que deverá ser seguira pelos agentes na avaliação dos artefatos do *software*.

Passos	Resumo
3.1.1. Analisar SIG RNF;	Fazer leitura do SIG RNF a partir do formalismo proposto por Chung et al. (2000) com as qualidades identificadas e relacionadas.
3.1.2. Escrever XML;	Escrever descrição das qualidades e seus relacionamentos a partir da linguagem XML com base na estrutura pré-definida no APÊNDICE A.
3.1.3. Verificar.	Verificar se os formatos inseridos estão condizentes com o padrão XML utilizado.
Observação: A estrutura pré-definida para a escrita do XML deve respeitar as hierarquias propostas no APÊNDICE A.	

A3.2	**Configurar *Questions patterns*** (estruturas XML)
	Resumo: Configuração de grupos, questões e operacionalizações relacionadas aos *objetive patterns*. A configuração continua sendo feita a partir do preenchimento dos *tags* XML do arquivo do APÊNDICE A.

Passos	Resumo
3.2.1. Configurar grupo;	Configurar grupos em formato XML que devem ser preenchidos na seção <*patternGrupo*> a partir de dois *tags* principais: <*idGrupo*>, que deve conter um identificar numérico sequencial para cada grupo; <*tituloGrupo*>, que propõe o preenchimento do título do grupo;
3.2.2. Configurar *Question patterns*;	Configurar *Question patterns* em formato XML que deverão ser preenchidos na seção <*patternQuestoes*> a partir de dois *tags* principais: <*idQuestao*>, que deve

	conter um identificar numérico sequencial para cada questão a partir da numeração definida no grupo; <*tituloQuestao*>*,* que propõe o preenchimento do título da questão;
3.2.3. Configurar *Alternative patterns*;	Configurar *Alternative patterns* em formato XML que deverão ser preenchidos na seção <*patternAlternativas*> a partir de dois *tags* principais: <*idAlternativa*>*,* que deve conter um identificar numérico sequencial para cada alternativa a partir da numeração definida na questão; <*tituloAlternativa*>, que propõe o preenchimento do título da questão; <*patternSelecao*>, que propõe o preenchimento da alternativa com o indicador RESPONDE (*ANSWER*), tal indicação formaliza que a alternativa está vinculada à questão como opção de resposta;
3.2.4. Verificar.	Verificar se os formatos inseridos estão condizentes com o padrão XML utilizado.
Observação:	

A3.3	**Configurar *Alternative patterns*** (estruturas XML)
	Resumo: Configuração de alternativas que servem como opção de respostas às questões estabelecidas no catálogo. A configuração continua sendo feita a partir do preenchimento dos *tags* XML do arquivo do APÊNDICE A.
Passos	Resumo
3.3.1. Configurar *Alternative patterns*;	Configurar *Alternative patterns* em formato XML que devem ser preenchidos na seção <*patternAlternativas*> a partir das *tags*: <*idAlternativa*>*,* que deve conter um identificar numérico sequencial para cada alternativa a partir da numeração definida na questão; <*tituloAlternativa*>, que propõe o preenchimento do título da questão; <*patternSelecao*>, que propõe o preenchimento da alternativa com o indicador RESPONDE (*ANSWER*), tal indicação formaliza que a alternativa está vinculada à questão como opção de

	resposta;
3.3.2. Verificar.	Verificar se os formatos inseridos estão condizentes com o padrão XML utilizado.
Observação:	

A3.4	**Configurar *Variáveis essenciais e seus sinônimos*** (estruturas XML)	
	Resumo: Define-se por variável essencial nessa tese, àqueles termos que são associados às alternativas no catálogo. Esses termos são utilizados para inserção nos artefatos de *software* para que possam ser analisados e confrontados com os termos disponíveis no catálogo com o objetivo de analisar conformidades da implementação de RNFs em *software*. Opcionalmente, pode-se utilizar, além das variáveis essenciais, os seus sinônimos. A atividade tem por finalidade a identificação de variáveis essenciais e seus sinônimos e sua posterior configuração no arquivo XML do catálogo do RNF. São o elo de ligação entre o SMA e os artefatos do *software*, pois permitem a identificação de conteúdos a serem analisados.	
Passos		Resumo
3.4.1. Identificar variáveis essenciais;		Mapear termos que sirvam de variáveis essenciais que possam servir de suporte para identificação de conteúdos nos artefato do *software*. Um insumo para o mapeamento das variáveis deve ser feito por um especialista a partir de dicionários de línguas;
3.4.2. Relacionar variáveis;		Relacionar as variáveis essenciais às alternativas identificadas;
3.4.3. Identificar sinônimos;		Identificar termos que possuem escrita distinta, mas o mesmo ou aproximadamente o mesmo significado da variável essencial identificada;
3.4.4. Relacionar sinônimos às variáveis essenciais.		Estabelecer vínculo entre os sinônimos e a variável essencial;
3.4.5. Editar arquivo XML;		Editar arcabouço XML disponível no APÊNDICE A;
3.4.6. Registrar variáveis essenciais;		Configurar Variaveis essenciais em formato XML que deverão ser preenchidos na seção <*variaveisEssenciais*> a partir das *tags*: <*idVariaveisEssenciais*>, que sugere o preenchimento do termo que identifica a variável essencial;

	<*nocaoVariaveisEssenciais*>, que objetivam o preenchimento da noção (significado) da variável essencial, tal noção é baseado na proposta de Léxicos [Leite et al. 1997];
3.4.7. Registrar sinônimos;	Configurar Sinônimos em formato XML que deverão ser preenchidos na seção <*sinonimosVariaveisEssenciais*> a partir da *tag*: <*idSinonimosVariaveisEssenciais*>, que propõe o preenchimento de sinônimos para a variável essencial. A abordagem de sinônimos foi proposta a partir do trabalho sobre Léxicos [Leite et al. 1997];
3.4.8. Verificar.	Verificar se os formatos inseridos estão condizentes com o padrão XML utilizado.
Observação:	

A3.5	**Configurar Padrão** (configurar conteúdos para avaliação automática)
	Resumo: Os padrões propostos são conteúdos baseados em heurística para que possam ser analisados automaticamente pelo SMA. Tais conteúdos são lidos no catálogo e comparados em textos dos artefatos analisados pelo SMA com posterior divulgação do resultado sem necessidade de interação humana. São alternativas às variáveis essenciais, uma vez que os artefatos analisados possuem algum padrão conhecido.

Passos	Resumo
3.5.1. Identificar padrões;	Estabelecer padrões que possam servir como operacionalização das alternativas. Os padrões devem ser conteúdos isolados ou em conjunto que permitam uma avaliação automática por parte dos SMA;
3.5.2. Editar arquivo XML;	Editar arcabouço XML disponível no APÊNDICE A;
3.5.3. Registrar padrões;	Configurar Padrões em formato XML que devem ser preenchidos na seção <*Padrao*> a partir das *tags*: <*idPadrão*>, que indicam o preenchimento do termo que identifica o padrão do conteúdo que é pesquisado no artefato de *software* para comparação;
3.5.4. Verificar estrutura.	Verificar relações a partir da análise de especialistas e *stakeholders* do domínio.
Observação: Nesse trabalho, o reconhecimento de padrões é feito a partir do	

reconhecimento do conteúdo indicado no marcador e comparado com conteúdos de artefatos de *software*. Não consiste em um sistema complexo de reconhecimento de padrões a partir de sensor que obtém observações a serem classificadas ou descritas; ou um mecanismo de extração de características que computa informações numéricas ou simbólicas das observações; e um esquema de classificação das observações, que depende das características extraídas. O SMA desenvolvido até o momento nessa pesquisa reconhece padrões configurados no catálogo e que não necessitam ser configurados no *software* a ser analisado. Estruturas mais elaboradas de padrões poderão ser criadas no futuro e implementadas no agente do SMA responsável pela analise de conteúdos. Os padrões são utilizados no SMA sem a necessidade de estruturação de variáveis essenciais e sinônimos, portanto, fazem a avaliação sem a necessidade de configuração do *software* analisado.

A3.6	**Verificar estrutura** (Verificar estrutura XML)	
	Resumo: Atividade que tem como objetivo a avaliação da sintaxe do catálogo XML após o preenchimento das estruturas.	
Passos		Resumo
3.6.1. Revisar estruturas de *tags*;		Analisar sintaticamente se estruturas estão condizentes com o modelo (APÊNDICE A);
3.6.2. Verificar relações.		Verificar se os formatos inseridos estão condizentes com o padrão XML utilizado (APÊNDICE A).
Observação:		

3.2.5
Detalhamento de Atividade - Configurar software (A4)

A atividade A4 não é decomposta com as atividades apresentadas até esse momento, portanto sua visualização deve ser feita a partir do modelo central (A0). Seu detalhamento é feito a partir do quadro com os passos necessários para a execução da atividade. Tal atividade só deve ser realizada para aquelas analises dependentes de variáveis essenciais, ou seja, o *software* que necessita ser analisado para confrontar sua estrutura com conformidades de um padrão pré-estabelecido em catálogo, necessitará indicar em qual artefato e em quais pontos do artefato há pontos que indicam que o *software* possui conformidade com o que está normatizado. Outras analises feitas pelo SMA a partir de padrões são feitas diretamente nos artefatos do software sem a necessidade de indicação dos pontos passíveis de analise, pois essas

são feitas automaticamente, bastando para isso que o artefato esteja presente na *path* onde o SMA coleta os dados para análise.

A4	**Configurar *software*** (configurar saída dos artefato do *software*)	
	Resumo: A atividade tem como objetivo deixar claro a necessidade de se configurar os artefato do *software* a ser analisado para que o mesmo gere traços com *tags* XML condizentes com um padrão de estabelecido em catálogo.	
Passos		Resumo
4.1. Escolher funcionalidades;		Escolher funcionalidades do *software* que possuem relação com o RNF objeto da análise ou artefatos do *software* que necessitam ser analisados para a análise da conformidade com o catálogo de RNF;
4.2. Configurar funcionalidades;		Configurar *software* para que gere artefatos com *tags* XML padronizados conforme APÊNDICE A. Devem ser configurados nos artefafos os conteúdos presentes nos marcadores *<idVariaveisEssenciais> ou <idsinonimosVariaveisEssenciais>.*
Observação:		

3.2.6
Detalhamento de Atividade - Analisar com agentes (A5)

A Figura 18 apresenta sucintamente as tarefas principais envolvidas na atividade Analisar com agentes (A5) do modelo central do método. Elas são descritas aqui de forma mais genérica uma vez que a arquitetura dos agentes, bem como modelos de seu funcionamento são apresentados em seções posteriores. Tal atividade demonstra as principais atividades dos agentes envolvidos no processo de análise. Eles atuam de forma geral basicamente na captura, análise e avaliação dos artefato quando comparados a um padrão pré-estabelecido.

É importante a percepção dos mecanismos descritos no SADT que servem de suporte tecnológico para a operacionalização dos agentes. Tais mecanismos são:

a) Catálogo XML: estrutura do catálogo do RNF formatado em estrutura XML, conforme apresentado no APÊNDICE A;
b) *tags* XML: *tags* em XML utilizados na comunicação dos agentes, nas estruturas internas da arquitetura do SMA, nos artefatos do *software* analisado e no catálogo de RNF;

c) plataforma Jadex: plataforma utilizada na implementação dos agentes;
d) arquitetura BDI: arquitetura ou modelo BDI utilizada na implementação da arquitetura funcional dos agentes.

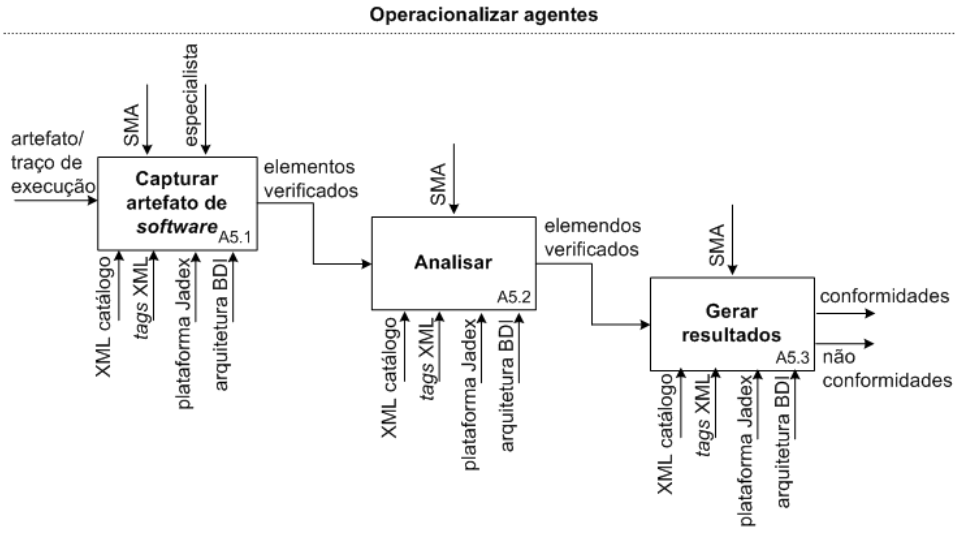


Figura 18: Decomposição da A5: Operacionalização de agentes.

A5.1	**Capturar artefato** (captura artefato de *software*)
	Resumo: O agente efetua coleta de artefatos do *software* que por ventura possam ser gerados. Para isso é necessário que o especialista configure a *path* onde estão os arquivos com os artefatos que são analisados. Uma vez o *path* configurado, o próprio SMA passa a verificar se há a existência de registros desses artefatos.

Passos	Resumo
5.1.1. Configurar *path*;	O especialista no SMA configura o endereço das pastas onde estão localizados os arquivos com os artefatos de *software* a serem analisados;
5.1.2. Executar SMA;	O especialista executa o SMA;
5.1.3. Captura dos artefatos.	Após a iniciação do SMA o agente responsável pela captura dos artefatos efetua a coleta dos dados.
Observação:	

A5.	**Analisar** (analisar artefato para análise)
	Resumo: Atividade tem como objetivo a análise dos artefaos a partir da comparação com padrão pré-estabelecido do catálogo de RNF (catálogo XML APÊNDICE A).

Passos	Resumo
5.2.1. Análise de artefatos;	Passo em que os agentes trabalham na analise dos artefatos do *software* em comparação com o catálogo de RNF;
Observação: os detalhes do *rationale* e das operacionalizações dos agentes serão apresentados posteriormente na seção de arquitetura dos agentes. Nessa seção, além de serem detalhados os agentes envolvidos no método sistêmico, serão também detalhadas suas ações e intencionalidades.	

	Gerar resultados
A5.3	Resumo: Atividade em que os agentes geram o relatório com informações de conformidade ou não conformidade do artefato com o que está estabelecido no catálogo de RNF.
Passos	Resumo
5.3.1. Gerar resultados;	Consiste em gerar resultados de conformidades do objeto analisado a partir de sua comparação com o catálogo do RNF.
Observação:	

3.3
Considerações Finais

O método descrito nas seções anteriores, composto pelas atividades de Criar SIG, Definir patterns, Configurar XML, Configurar *software* e Operacionalizar agentes é apoiado em na arquitetura inicial. Essa arquitetura que utiliza agentes para atuar em ambiente orientado à metas, distribuídos com objetivos distintos, relacionados por meio de troca de mensagens para atuarem de forma colaborativa a partir de um conjunto de operações, possibilita a interação do SMA com o catálogo XML e com os artefatos de *software* para analisar as conformidades em *software.* As arquiteturas, tanto do SMA como do catálogo, que dão suporte ao método passam a ser detalhadas na seção posterior.

4.
Arquiteturas da Infraestrutura que dão Suporte o Método

Nessa seção são apresentadas as partes da infraestrutura utilizadas para suporte do método apresentado no Capítulo 3. Primeiramente é apresentado o modelo conceitual dos patterns para RNF. Em seguida o detalhamento da criação e estrutura do catálogo XML de RNFs. Para a criação e uso das variáveis essenciais e seus sinônimos é apresentada a técnica de Léxicos e são mostrados esses elementos de acordo com a estrutura XML proposta para o catálogo de RNFs. Essas partes dão apoio à arquitetura do SMA, que é apresentada a partir da intencionalidade dos agentes sob a perspectiva de Painéis de Intencionalidade e Modelos i, seguida da arquitetura de implementação baseada no Mapeamento i* x BDI x Jadex.*

4.1
Modelo conceitual dos *patterns*

O modelo conceitual dos *patterns* de RNF elaborado nessa seção surge a partir das seguintes propostas:

- Supakkul et al. (2010) no que se refere a camadas de "Padrões Objetivo" (Objective Patterns), subdividido em *patterns* Identificação e Decomposição. O primeiro para identificar o RNF objeto da análise e o segundo suas decomposições. Além dos Padrões Seleção (*Selection Patterns*).

- o proposto em CTS (2013) e Serrano e Leite (2011a), no que se refere à criação de grupos (categorias) de questões, questões, alternativas (respostas às questões) baseado na proposta do GQO;

- variáveis essenciais e seus sinônimos [Leite et al. 2000], propostas nessa tese, para que possam servir de marcadores para operações dos agentes do SMA na comparação de artefatos do *software* analisado com o padrão estabelecido em catálogo.

O modelo conceitual dos *patterns* pode ser visto na Figura 19. A figura apresenta na sua estrutura os padrões, relações e as suas cardinalidades. São detalhados no modelos os *patterns* Objetivos, subdividido em padrão Identificação e Decomposição, Grupo de Questões, Questões, Alternativas, Variáveis Essenciais com seus sinônimos e padrões, esses últimos onde são inseridas marcações com conteúdos padronizados vinculados às alternativas para serem pesquisados em artefatos de *software* que por *default* já possuem esse tipo de marcação.

O *pattern* Identificação é o elemento principal de onde todos os outros passam a ser estruturados. Ele deverá conter *softgoal* principal a ter sua implementação analisada no *software*. Esse *pattern* possui decomposições para obter-se maior detalhamento sobre si e esse detalhamento é representado pelo *patterns* Decomposições. Uma decomposição pode ainda ser desmembrada em outras decomposições, quando for necessário seu detalhamento para explicitar suas características.

Grupos de questões e questões são definidas conforme proposto por Serrano e Leite (2010) para que minimize a dificuldade de se elicitar operacionalizações, que no NFR *framework* [Chung et al. 2000] são associadas diretamente aos *softgoals*. O grupo de questões e questões são um facilitador no processo de elicitação de operacionalizações, uma vez que elas possibilitam uma reflexão sobre o *softgoal* a partir de perguntas agrupadas em categorias de mesma natureza. As respostas às perguntas são as alternativas, ou operacionalizações, transformadas em metas concretas a serem satisfeitas, para minimizar a subjetividade de julgamento dos *softgoals*.

As alternativas de respostas deverão ser pensadas com o objetivo de reuso entre as diversas questões do catálogo do RNF, uma vez que os padrões são estabelecidos a partir de uma estrutura que permite seu reuso, onde o ponto de variabilidade encontra-se nos conteúdos preenchidos em cada *pattern* que dependerâ do domínio da aplicação.

As variáveis essenciais devem representar marcadores passíveis de comparação com os artefatos do *software*. Dessa forma, a comparação entre esses marcadores e o artefato, bem como sua relação com aos *patterns* alternativas possibilitam a avaliação do artefato a partir das regras de *patterns* e suas associações definidas no método. Alternativamente, pode-se configurar o elemento de nome Padrões, que como já mencionado, são marcações padronizados vinculados às alternativas e que são encontradas em artefatos de *software* geradas pelo próprio *software*, mas que devem ser configuradas para se permitir uma comparação com o catálogo e consequentemente sua análise de conformidade.

O SMA irá confrontar os *patterns* do modelo apresentado na Figura 19, configurados em catálogo por *tags* XML, com marcações nos artefatos analisados.

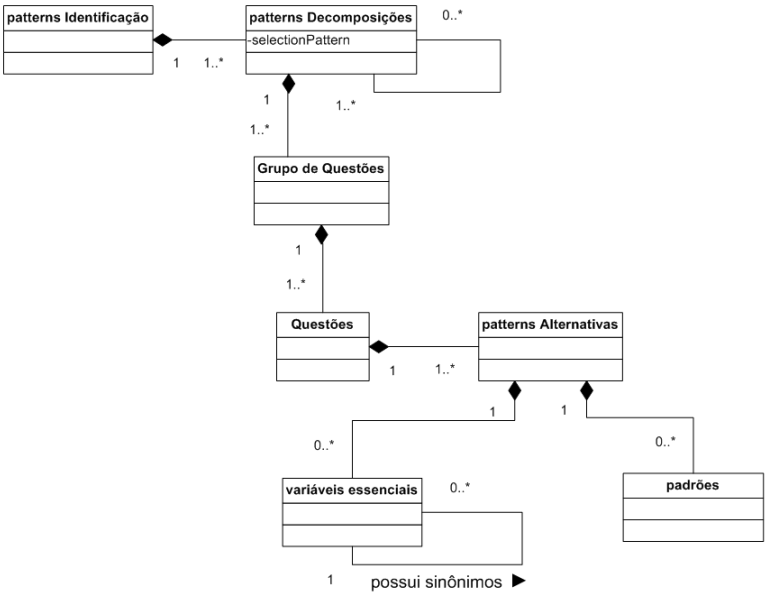


Figura 19: Modelo conceitual dos *patterns* de RNF.

4.2
Catálogo XML

Após a criação do modelo conceitual dos *patterns* foi possível iniciar a criação de uma estrutura XML para o catálogo de *patterns* de RNF, uma contribuição dessa tese. Sua estrutura é baseada em *tags* XML criados a partir da estrutura principal do modelo conceitual e serve como regra, uma vez que toda sua arquitetura de decomposição de elementos, relacionamentos, elos de contribuição, conteúdos estão definidas na estrutura.

A estrutura XML criada pode ser vista no APÊNDICE A com suas descrições apresentadas na Tabela 2. Na Tabela 2 as *tags tags* finais são utilizadas conforme padrão XML iniciadas por "/" como por exemplo: </catalogo>, </versão>, </patternObjetivos>, mas não foram apresentadas na tabela.

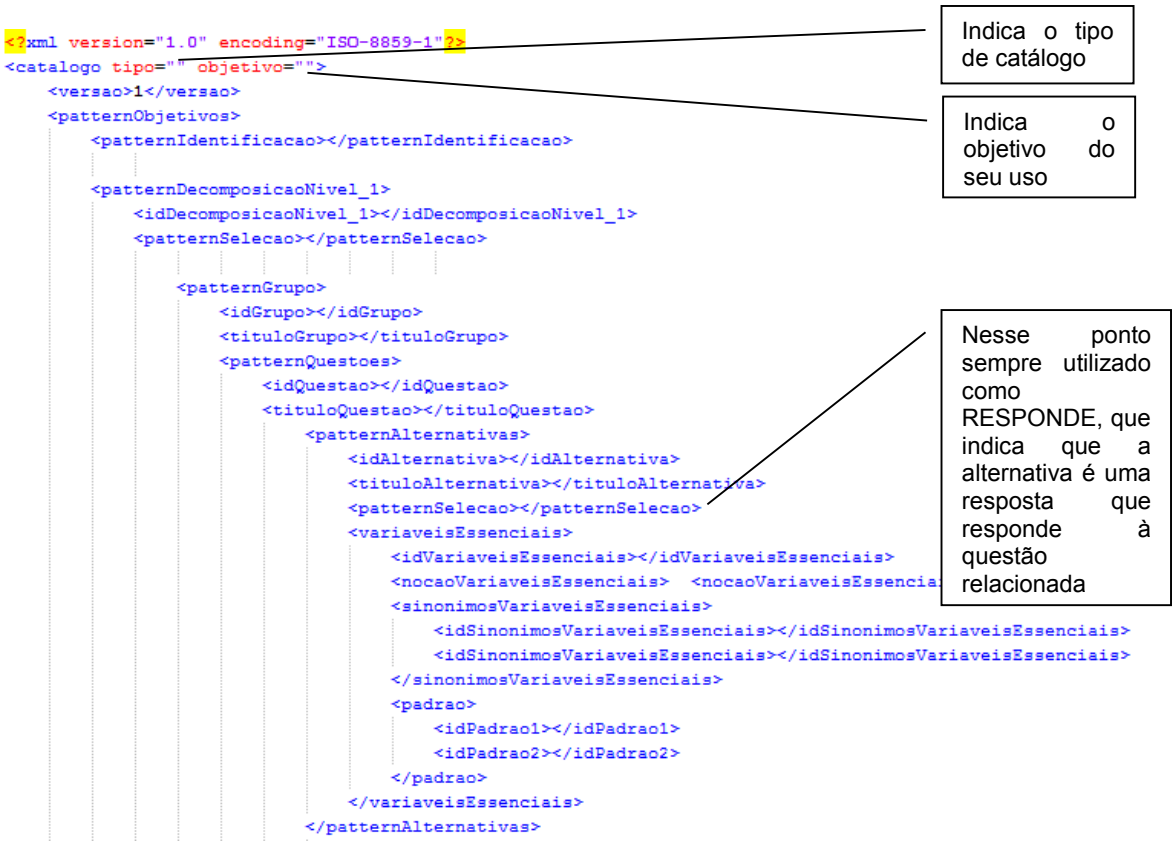
Tabela 2: *Tags* presentes no catálogo de RNF.

Identificador	Descrição
`<catalogo tipo="" objetivo="">`	Identificador inicial da estrutura do catálogo de RNF. Utiliza dois parâmetros TIPO e OBJETIVO. O primeiro indica o tipo de catálogo, caso um catálogo de RNF e o segundo seu objetivo, utilizado nesse trabalho como análise.
`<versao>`	Versão do catálogo. Deve ser indicada, pois, a análise pode ser baseada em versões diferentes do catálogo.
`<patternObjetivos>`	Identificador inicial dos *patterns* objetivos.
`<patternIdentificacao>`	Identificador inicial do RNF principal objeto de análise.
`<patternDecomposicaoNivel_1>`	Identificador inicial da seção quer irá apresentar as decomposições do

	patternIdentificacao em outros RNFs.
<idDecomposicaoNivel_1>	Identificador inicial do nome do RNF da decomposição de primeiro nível.
<patternSelecao>	Identificador inicial do *patternSelecao* que irá conter elos de ligação do tipo *Some (+-), Help, Make, Hurt, Break* que irá indicar o elo do tipo de contribuição da decomposição para o *patternIdentificacao.* Pode estar presente em diferentes seções do catálogo XML. Para o *patternAlternativas* a *tag* deve ser apenas preenchida com RESPONDE, que indica que a alternativa responde à questão na qual está relacionada.
<patternDecomposicaoNivel_2>	Identificador inicial da seção quer irá apresentar as decomposições do *patternDecomposicaoNivel_1* em outros RNFs
<idDecomposicaoNivel_2>	Identificador inicial do nome do RNF da decomposição de segundo nível.
<patternGrupo>	Identificador inicial das representações das questões e alternativas detalhadas a partir da proposta de GQO.
<idGrupo>	Identificador inicial do nome do grupo (agrupador) da questão.
<tituloGrupo>	Identificador inicial da descrição do grupo.
<patternQuestoes>	Identificador inicial da seção que irá detalhar as questões e suas alternativas.
<idQuestao>	Identificador inicial da descrição da questão.
<tituloQuestao>	Identificador inicial do título da questão ou descrição da questão.
<patternAlternativas>	Identificador inicial da seção que irá detalhar alternativas de respostas para às questões.
<idAlternativa>	Identificador inicial do identificador (código ou sigla) da alternativa.
<tituloAlternativa>	Identificador inicial do detalhe do título da alternativa.
<variaveisEssenciais>	Identificador inicial da seção que irá detalhar as variáveis essenciais e seus sinônimos.
<idVariaveisEssenciais>	Identificador inicial da descrição do símbolo (palavra, termo ou frase sucinta) que irá representar uma variável essencial vinculada a uma alternativa.
<nocaoVariaveisEssenciais>	Identificador inicial que descreve o significado da variável essencial.
<sinonimosVariaveisEssenciais>	Identificador inicial da seção quer irá detalhar os sinônimos das variáveis essenciais.
<idSinonimosVariaveisEssenciais>	Identificador inicial da descrição do símbolo (palavra, termo ou frase sucinta) que irá representar um sinônimo vinculado a uma variável essencial.
<padrão>	Identificador inicial da descrição de padrões.
<idpadrao>	Indicador dos padrões utilizados para representar as alternativas. Os padrões podem ser utilizados com *tags* simples do tipo: <idpadrao> </idpadrao>, nesse caso o SMA irá efetuar a pesquisa pelo conteúdo indicado entre as *tags* de forma isolada; e pode utilizar padrões compostos do tipo: <idpadrao1> </idpadrão1>, <idpadrao2> </idpadrão2>, ... <idpadraoN> </idpadrãoN>, o que indica que o

	SMA irá fazer uma pesquisa e comparação de toda a estrutura do padrão, que deve ser informado de forma seguida, um embaixo do outro, com os conteúdos dispostos entre *tags*. Para efeito de estudo, está implementado no SMA o uso de % para os conteúdos dos padrões, o que significa que ao encontrar um conteúdo do tipo %, o SMA irá interpretá-lo como qualquer cadeia de *string*. O uso de mais de um carácter especial % é permitido.

A organização hierárquica do catálogo de RNF pode ser vista conforme apresentado no código XML. O trecho de código é referente à estrutura do metadado das *tags* explicadas na tabela acima.




```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalogo tipo="" objetivo="">
    <versao>1</versao>
    <patternObjetivos>
        <patternIdentificacao></patternIdentificacao>

        <patternDecomposicaoNivel_1>
            <idDecomposicaoNivel_1></idDecomposicaoNivel_1>
            <patternSelecao></patternSelecao>

                <patternGrupo>
                    <idGrupo></idGrupo>
                    <tituloGrupo></tituloGrupo>
                    <patternQuestoes>
                        <idQuestao></idQuestao>
                        <tituloQuestao></tituloQuestao>
                            <patternAlternativas>
                                <idAlternativa></idAlternativa>
                                <tituloAlternativa></tituloAlternativa>
                                <patternSelecao></patternSelecao>
                                <variaveisEssenciais>
                                    <idVariaveisEssenciais></idVariaveisEssenciais>
                                    <nocaoVariaveisEssenciais>  <nocaoVariaveisEssencia
                                    <sinonimosVariaveisEssenciais>
                                        <idSinonimosVariaveisEssenciais></idSinonimosVariaveisEssenciais>
                                        <idSinonimosVariaveisEssenciais></idSinonimosVariaveisEssenciais>
                                    </sinonimosVariaveisEssenciais>
                                    <padrao>
                                        <idPadrao1></idPadrao1>
                                        <idPadrao2></idPadrao2>
                                    </padrao>
                                </variaveisEssenciais>
                            </patternAlternativas>
```

4.3
Uso do Léxico na Definição de Variáveis Essenciais

A estrutura XML nessa seção apresenta o metadado e o conteúdo das *tags* referentes às variáveis essenciais e seus sinônimos definidos no catálogo XML (APÊNDICE A). As três *tags* <idVariaveisEssenciais>, <nocaoVariaveisEssenciais> e <sinonimosVariaveisEssenciais> tiveram como fundamento o que é utilizado em Léxico Ampliado da Linguagem (LAL) [Leite et al. 1997] para definição de símbolos, seu

significado (noção) e sinônimos. O trecho de código é um exemplo dos conteúdos da variável essencial “Data da Coleta” com sua noção e os sinônimos.

```
<variaveisEssenciais>
    <idVariaveisEssenciais>Data da Coleta</idVariaveisEssenciais>
    <nocaoVariaveisEssenciais>Data em que o traço de execução foi coletado
            <nocaoVariaveisEssenciais>
        <sinonimosVariaveisEssenciais>
            <idSinonimosVariaveisEssenciais>Coletado em</idSinonimosVariaveisEssenciais>
          <idSinonimosVariaveisEssenciais>Gravado em</idSinonimosVariaveisEssenciais>
        </sinonimosVariaveisEssenciais>
</variaveisEssenciais>
```

O processo de elicitação do LAL, ou simplesmente, Léxico, permite a identificação das frases e palavras amparadas pelas estratégias de identificação de símbolos da linguagem do problema, ou Universo de Informações (UdI), e da identificação da semântica de cada símbolo [Leite et al. 1997]. O Léxico do UdI é composto por entradas, onde cada entrada está associada a um símbolo (palavra ou frase) da linguagem do UdI. Esses símbolos podem possuir sinônimos e são descritos por noções e impactos.

O Léxico permite a captura do significado (símbolos) dos termos a partir da Noção, bem como seus sinônimos, como a maioria dos glossários, e também inclui a conotação dos mesmos. Além disso, o Impacto permite descrever os efeitos do uso/ocorrência do símbolo na aplicação, ou do efeito de algo na aplicação sobre o símbolo [Leite et al 1997]. Desta forma podemos compreender o significado dos termos de modo independente e em relação a outros termos. Essa definição de noções e impactos devem ser estruturadas com base nos princípios de vocabulário mínimo e circularidade que prescrevem que as noções e impactos devem ser descritos com a utilização dos símbolos da própria linguagem do problema sem o uso demasiado de símbolos externos ao Léxico.

Para identificar a qual classe o símbolo pertence, em [Leite et al. 2000] são sugeridas as seguintes heurísticas: se o símbolo pratica uma ação, ele é classificado como sujeito; se é quem sofre a ação, é classificado como objeto; se o símbolo é uma situação em um dado momento, ele é classificado como estado; se representa uma ação, é classificado como verbo. A Tabela 3 apresenta os detalhes das classificações de noção e impacto dos símbolos de acordo com as classes.

Tabela 3: Noção e Impacto em Léxico [Leite et al. 1997].

	Noção	**Impacto**
Sujeito	Quem é o sujeito	Quais ações executa.
Verbo	Quem realiza, quando acontece e quais os procedimentos envolvidos.	Quais os reflexos da ação no ambiente (outras ações que devem ocorrer) e quais os novos estados

		decorrentes.
Objeto	Definir o objeto e identificar outros objetos com os quais se relaciona.	Ações que podem ser aplicadas ao objeto.
Estado	O que significa e quais ações levaram a esse estado	Identificar outros estados e ações que pode ocorrer a partir do estado que se descreve.

A abordagem da especificação da linguagem do domínio utilizada nesse trabalho tem por objetivo utilizar a sua estrutura na definição das variáveis essenciais para o método sistêmico. As variáveis essenciais são utilizadas para permitir que o SMA possa fazer a leitura dos artefatos do *software*, com o objetivo de encontrá-las como marcadores válidos de conteúdos dos artefatos de *software* e a posterior comparação com as regras definidas. As buscas nos artefatos são feitas a partir das variáveis essenciais e também por seus sinônimos definidos no catálogo XML.

Diante das definições do UdI, o Léxico se torna um potencial elemento para a geração das variáveis essenciais e seus sinônimos, uma vez que esse levantamento e especificação são feitos em tempo de definição de requisitos. Dessa forma, é sugerido que seja feita a associação dos elementos do Léxico ao catálogo XML de RNF.

4.4
A Arquitetura do SMA

O método proposto nessa tese tem o suporte da análise de conformidades de RNF implementado em *software* a partir da execução de um SMA. Agentes são executados na leitura dos artefatos de *software* e a partir de tarefas distintas, comparam marcações nos artefatos com o que é configurado no catálogo em XML, conforme descrito no método.

Nas seções posteriores são apresentadas as arquiteturas que envolvem o SMA. Primeiro, com o Painel de Intencionalidades, ou Diagrama IP, para demonstrar a intencionalidade das interações dos agentes. Em uma outra seção é apresentado o modelo i* do SMA e por fim são feitas considerações a respeito de sua camada de implementação.

4.4.1
Painel de Intencionalidades dos Agentes do SMA

O detalhamento da arquitetura do SMA a partir da representação por Diagramas IP [Oliveira et al. 2007] potencializa o entendimento sobre à intencionalidade das interações entre os agentes.

A Figura 20 apresenta na sua parte superior a representação dos agentes e em seu eixo vertical, as metas de cada ator. O agente Monitor possui as metas concretas “Informações sejam monitoradas” e “Informações sejam organizadas”. No caso, monitorada indica que uma pasta onde artefatos devem estar disponíveis para serem executados com o SMA passam a ser lidos e capturados, caso existam. No intervalo entre essas duas metas concretas há as metas flexíveis do agente, representadas pelas figuras tracejadas. É possível perceber os elos de contribuição positivo entre essas metas indicando que ao atingir um nível de qualidade do rastro, há um consequente benefício da meta flexível “Compatibilidade [regras]”, ou seja, a compatibilidade dos rastros com as regras exigidas pelo método. As dependências entre as metas dos agentes ficam evidenciadas a partir das setas tracejadas que indicam a dependência entre metas dos diferentes agentes. Dessa forma, o modelo propõe que a meta concreta “Informações sejam padronizadas”, apresentada no eixo do agente Canonizador, sofre uma dependência de metas do agente Monitor a apresenta de forma explícita essas metas.

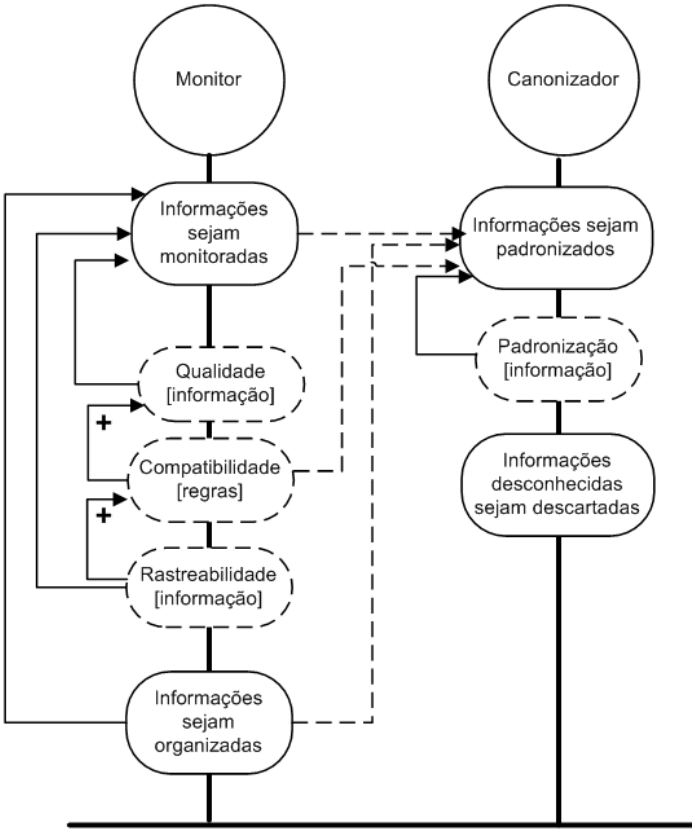


Figura 20: Diagrama IP das intencionalidades dos agentes Monitor e Canonizador.

A Figura 21 apresenta a relação de dependência entre os agentes Canonizador e o Consolidador, e também a relação de dependência entre Canonizador e Analisador.

No eixo do Consolidador é possível identificar as suas metas que incluem a meta concreta “Informações sejam consolidados” que quer dizer o objetivo principal do agente a partir de todas as suas ações para verificar os artefatos do *software* compatível com regras definidas no método. Essa compatibilidade não se dá apenas pela estrutura do arquivo, mas sim pela identificação de elementos nas informações que sejam compatíveis com um catálogo pré-definido para RNFs. É possível perceber nas metas flexíveis do agente a identificação de regras compatíveis irá contribuir de forma positiva para a qualidade da informação e consequentemente para a meta concreta principal, como também que sua meta de encontrar informações incompatíveis é utilizada para representar a contribuição negativa desse tipo de elemento na cadeia de metas que procuram contribuir positivamente para a meta concreta.

O Canonizador possui a tarefa de encontrar assinaturas conhecidas dentro do artefato analisado e comparado com o catálogo XML para que as ocorrências possam ser passadas diretamente para o Analisador. Essa padronização é um meio de tornar as informações estruturadas a partir de um padrão que o sistema de análise possa

reconhecer ou em outra situação de verificar se *tags* presentes nos artefatos são conhecidas ou desconhecidas. Essa necessidade de padronização é representada pela meta concreta “Informações sejam padronizados”. Os artefatos com assinaturas desconhecidas são armazenadas para que uma análise futura possa ser feita a partir da atualização do catálogo XML. A atualização do catálogo deve ser feita por especialista pela análise das assinaturas desconhecidas. Os artefatos com assinaturas conhecidas são passadas diretamente para o Analisador.

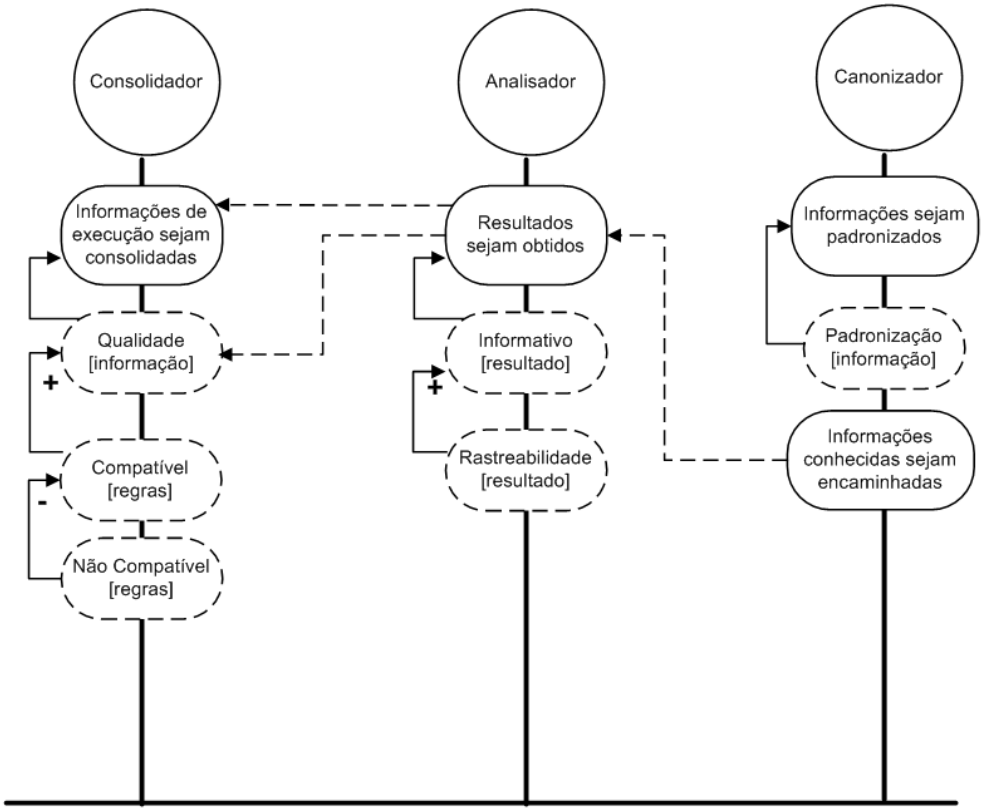


Figura 21: Diagrama IP das intencionalidades dos agentes Canonizador/Consolidador e Canonizador/Analisador.

Já a Figura 22 apresenta a relação de dependência entre os agentes Consolidador e o Analisador e procura demonstrar as metas de cada agente, conforme mesma lógica de apresentação dos dois casos anteriores, como também a relação de dependência dos agentes. Essa dependência se dá basicamente pela relação das metas onde a identificação de compatibilidade das informações lidas em artefatos do *software*, representado pelas metas do agente Consolidador, irão contribuir para a sugestão de níveis do RNF representado pela meta concreta do agente Analisador. Particularmente no eixo desses agentes, há metas flexíveis voltadas ao grupo que contribuem de forma positiva para a meta concreta do agente. O Consolidador permite a leitura e identificação de padrões estabelecidos no catálogo de RNF pelo objetivo de consolidação de informações.

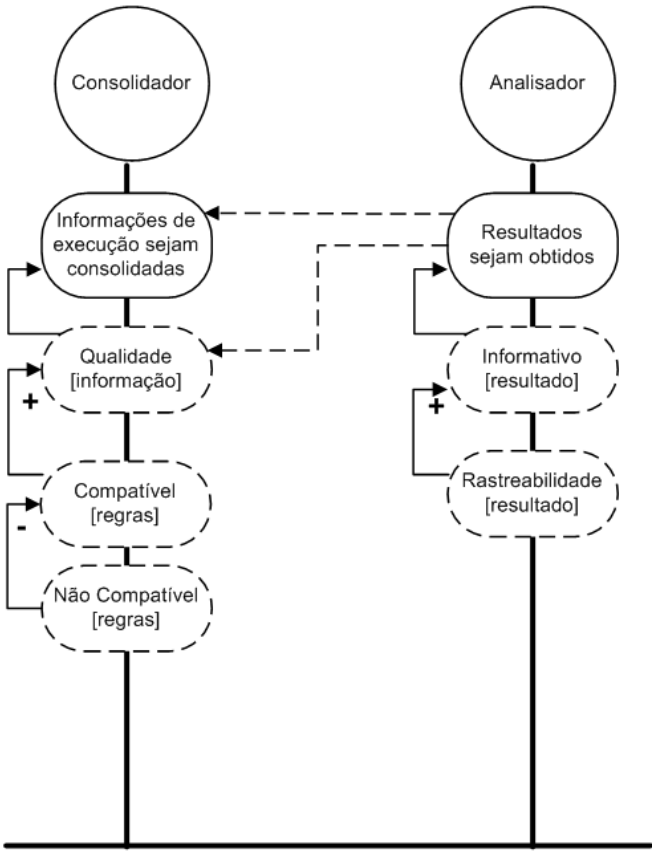


Figura 22: Diagrama IP das intencionalidades dos agentes Consolidador e Analisador.

Por fim a relação de dependência entre os agentes Monitor e Analisador é apresentada na Figura 23. A apresentação das metas posicionadas no eixo de cada agente já foi discutida anteriormente e não sofre alterações no próprio eixo, mas é importante deixar destacado que a cada nova relação de agentes confrontada gera novas situações de dependência entre metas. Dessa forma, agentes Monitor e Analisador, quando colocados juntos, destacam suas relações de dependência e o que é possível notar é uma equivalência entre as metas flexíveis posicionadas nos eixos de cada ator, no caso a meta “Rastreabilidade [resultado]”. Além disso, o modelo deixa explícitas outras tantas relações de dependência, onde ao atingir a meta em um eixo de um ator irá consequentemente beneficiar metas do eixo do outro ator. Nesse caso a rastreabilidade um produto gerado pelas execuções dos atores e que facilitam em determinado momento a otimização do processo de análise dos RNFs.

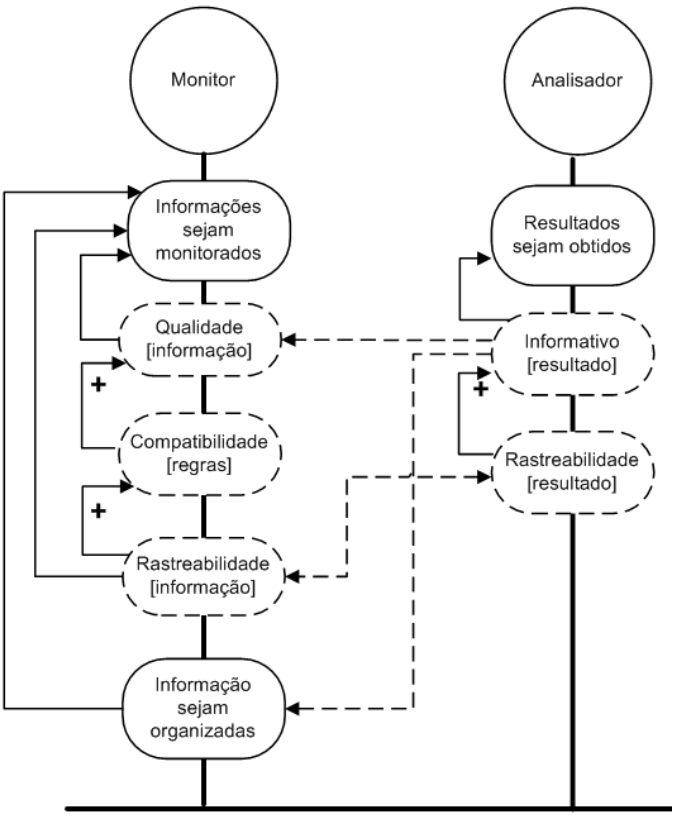


Figura 23: Diagrama IP das intencionalidades dos agentes Monitor e Analisador.

4.4.2
Modelo Intencional dos Agentes

Nessa última versão o SMA (as versões anteriores foram apresentadas na seção 3.1 Contexto Histórico da Pesquisa) é composto por quatro agentes com objetivos e funções distintas, conforme apresentado nos Diagramas IPs. Embora haja essa especialização (cada agente com funções distintas e bem definidas) há um objetivo central que é a avaliação das informações dos artefatos de *software* em relação a um catálogo pré-definido do RNF. A seguir apresentamos o modelo central, em i* e detalhes sobre a estrutura de assinaturas e estrutura de proveniência.

4.4.2.1
*O Modelo i**

Para atingir ao objetivo principal, cada agente trata de forma distinta suas tarefas, recursos, desejos e objetivos. Suas discriminações são apresentadas como:

- MONITOR: o agente monitora constantemente uma pasta onde artefatos devem estar disponíveis para que o SMA consiga efetuar a análise. Após coleta do artefato o Monitor verifica se já há artefatos do agente ANALISADOR referente à memória de avaliações anteriores (base de conhecimento), tais rastros são registrados

a partir de uma estratégia de proveniência [Miles et al. 2005 e 2007]. A base de conhecimento do ANALISADOR contém o registro de *tags* das regras de análise baseados na execução do agente Canonizador e a correlação dessas *tags* com as alternativas do catálogo do RNF de monitorações anteriores.

Dessa forma, caso encontre correlações das informações lidas recentemente com a base de conhecimento, o MONITOR encaminha diretamente ao ANALISADOR as respostas de conformidade sem ter a necessidade de percorrer todo o processo de análise, ou seja, passando por outros agentes como o CANONIZADOR e o CONSOLIDADOR. Dessa forma, tem-se um processo otimizado apoiado pela base de conhecimento formada a partir do histórico de análises. Caso o agente não encontre indicativos de marcações conhecidas e já analisadas anterioemente na base de conhecimento alimentada pelo ANALISADOR ele procede com o processo completo e envia uma mensagem ao agente CANONIZADOR informando sobre a captura e disponibilidade do arquivo. Está definido para o processo otimizado que cem por cento das marcações encontradas no artefato devem ser compatíveis e já analisadas anterioremente.

- CANONIZADOR: tem por objetivo receber os artefatos do MONITOR para proceder com a leitura e comparação estruturas de *tags* conhecidas, ou seja, aquelas disponíveis em catálogo. Dessa forma, o CANONIZADOR separa artefatos que possuem marcações compatíveis com o catálogo, sem fazer a análise de satisfação do *softgoal* e também artefatos que não possuem marcações ou marcações desconhecidas. Artefatos que possuam apenas marcações compatíveis com variáveis essenciais ou sinônimos são encaminhados diretamente para o ANALISADOR, apóes pesquisa em base de estruturas conhecidas, enquanto que artefatos com marcações compatíveis com padrões são encaminhadas para o CONSOLIDADOR para que efetue a análise de padrões, ou seja, se o conteúdo disposto indica compatibilidade com o catálogo.

- CONSOLIDADOR: verifica a partir do catálogo se o conteúdo disponível no artefato, seja delimitado por marcações ou conteúdos avulsos, condiz com estruturas conhecidas e disponibiliza ao ANALISADOR para que possa ser analisado e os resultados obtidos.

- ANALISADOR: efetua a comparação de marcações dos artefatos disponibilizados com o catálogo de RNF. Ao comparar as marcações conhecidas com variáveis essenciais, sinônimos e padrões, o agente inicia o processo de propagação de acordo com *patterns seleção* indicados no catálogo. Com a propagação, o agente sinaliza se o *softgoal* foi satisfeito a contendo e os resultados de conformidade e não conformidades (não atende) são apresentados através de uma trilha que tem origem

na alternativa, passa pelas questões e culmina no(s) *softgoal(s)*, de acordo com o SIG definido.

A cada análise efetuada o ANALISADOR gera:

- base de estruturas conhecidas: armazena em uma estrutura específica as *tags* que analisou e que estiveram em conformidade com o catálogo. Dessa forma, compõe uma "memória de *tags*" (assinaturas) já analisadas.

- base de conhecimento de proveniência: registra em estrutura própria o rastro de analises dos diversos artefatos. Nesse rastro estão as *tags* de variáveis essenciais, sinônimos e padrões que satisfizeram alternativas nas análises efetuadas.

- resultado a partir da análise da propagação dos elos de relação entre operacionalizações e *softgoals* e entre *softgoals*: registra em estrutura própria o resultado das análises por artefato, conforme poderão ser vistos nos estudos de caso. Esses resultados são apresentados em forma de relatórios, *report*, em padrão XML com artefatos de saída que apresentam as análises efetuadas pelo SMA.

A interação dos agentes é determinada pelo modelo intencional conforme Figura 24:

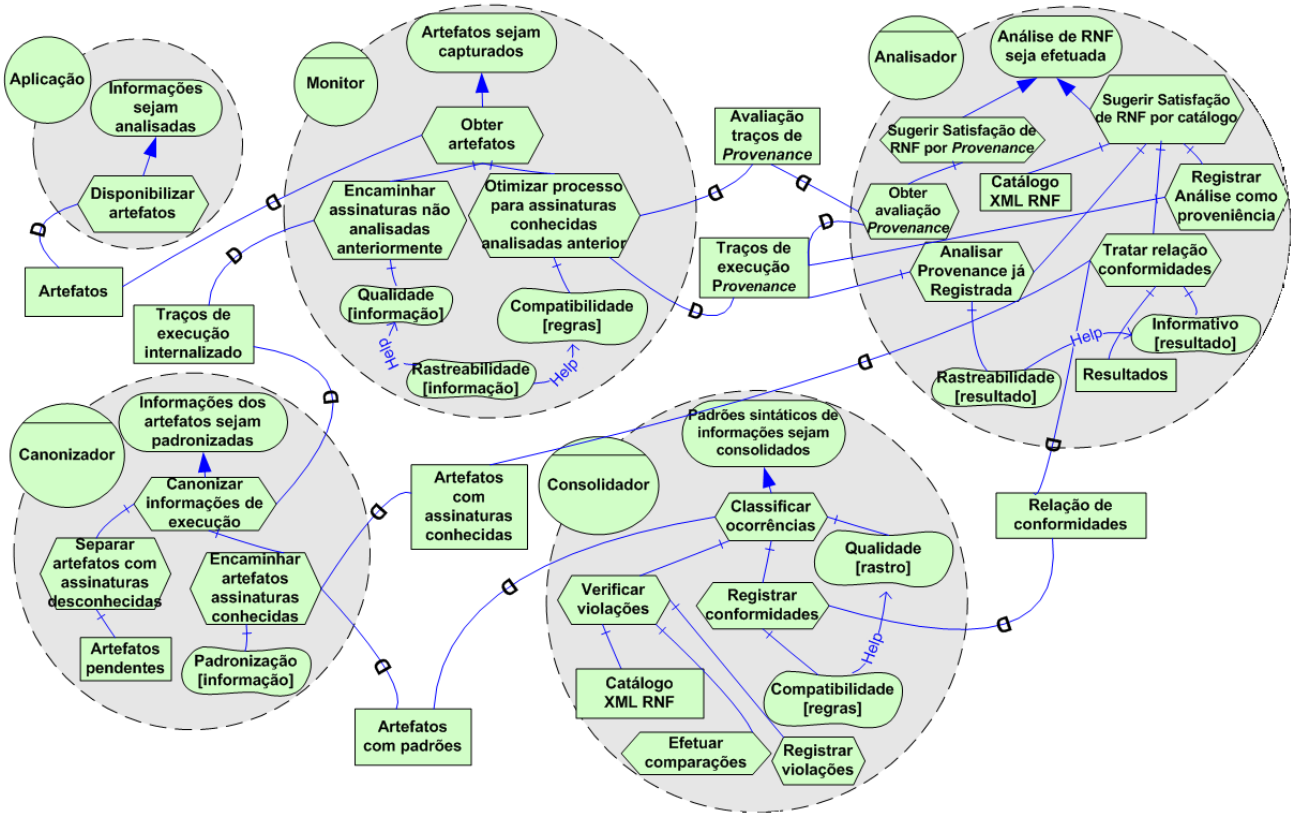


Figura 24: Modelo intencional da interação dos agentes – *framework* i*.

4.4.2.2
Estrutura de Assinaturas Conhecidas

As assinaturas conhecidas correspondem àquelas *tags* que foram analisadas pelo analisador e cujo resultado da análise foi positiva, ou seja, que houve a compatibilidade da *tag* do artefato, com a *tag* do catálogo, portanto dada como em conformidade. A estrutura de *tags* (assinaturas) conhecidas é criada a partir da primeira analise e à medida que outros artefatos são analisados, a estrutura é reestruturada e acrescida de novas *tags.* Dessa forma, ela só é criada a partir da primeira vez que o SMA é executado.

A estrutura segue o padrão XML e nela são armazenados um identificador, a *tag* e assinaturas equivalentes, ou sinônimos, registradas no próprio catálogo. Sua composição segue como pode ser apresentado a seguir:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<kbase tipo="RNF" objetivo="estruturas conhecidas">
    <KnownStructureBase>
        <KnownStructure>
            <index>0</index>
            <name>agente</name>
            <equivalency>ator<equivalency/>
        </KnownStructure>

        <KnownStructure>
            <index>1</index>
            <name>data da coleta</name>
            <equivalency>data da gravação<equivalency/>
          <equivalency>data do registro<equivalency/>
        </KnownStructure>

        <KnownStructure>
            <index>2</index>
            <name>error</name>
            <equivalency>erro<equivalency/>
           <equivalency>fault<equivalency/>
            <equivalency>falha<equivalency/>
        </KnownStructure>
    </KnownStructureBase>
</kbase?
```

Tal estrutura auxilia no processo de análise, uma vez que o CANONIZADOR só permite que sejam processados artefatos que possuem *tags* conhecidas. Uma vez que isso é identificado, o artefato pode ser analisado pelo ANALISADOR para as devidas comparações com o catálogo para que os resultados sejam obtidos e apresentados.

4.4.2.3
Estrutura de Dados de Proveniência

Proveniência (*provenance*) tem sido utilizado por pesquisadores para manter rastros de dados para a obtenção da forma com que esses dados foram produzidos e para a análise da qualidade do processo que o produziu [Miles et al. 2005] [Miles et al. 2007]. O dicionário Inglês Oxford (2013) define *provenance* como: '*(i) The fact of coming from some particular source or quarter; origin, derivation; (ii) The history of the ownership of a work of art or an antique, used as a guide to authenticity or quality; a documented record of this*'. Pinheiro [Pinheiro et al. 2003] and Vasquez [Vasques et al. 2005] propõem que proveniência tem um foco baseado na história, no processo e na transformação da origem dos dados, mas também deveria prover métricas qualitativas e quantitativas.

O objetivo do uso de proveniência nesse trabalho foi o de dar uma solução heurística, que consiste em efetuar uma análise customizada a partir de uma base de conhecimentos de outras análises efetuadas pelo agente ANALISADOR para dar maior agilidade no processo de análise e geração dos resultados. Além disso, a estratégia de proveniência permite o acumulo de conhecimento a partir de diversas análises de *software* que formam uma base de conhecimento que registraram conformidades com catálogo. A partir dessa abordagem, a proposta de análise pretende estar aderente a um conceito mais amplo de proveniência, que está relacionado não somente na origem da informação, mas também de como ela é derivada, o seu contexto e como é utilizada.

Para prover artefatos de proveniência o agente ANALISADOR mantém um rastro de sua própria execução no momento da comparação das *tags* dos *software* analisados com as alternativas, questões e grupos do catálogo de RNF e esse rastro passa a servir como heurística para novas análises.

O recorte superior da Figura 25 aprepresenta uma otimização do processo de analise com o uso da base de conhecimento formada a partir dos traços de proveniência. A interação entre os agentes MONITOR e ANALISADOR requer apenas os artefatos do software e os de proveniência para que seja avaliada e sugerida a conformidade com o RNF, uma vez que o agente MONITOR utiliza os artefatos de proveniência como heurística de monitoração e transfere o processamento para o ANALISADOR ao invés de caminhar com o processo para uma varredura mais completa a partir dos agentes CANONIZADOR E CONSOLIDADOR.

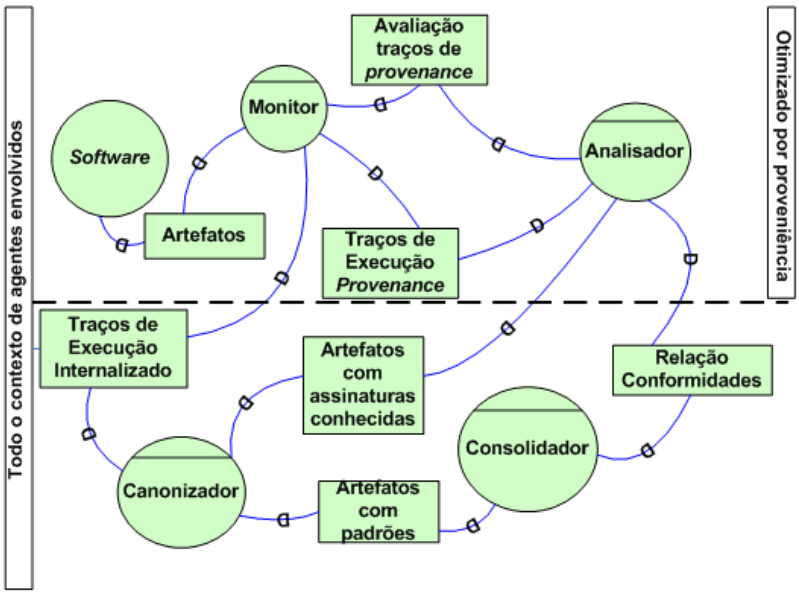


Figura 25: Visão da interação dos agentes com recorte da análise com suporte de proveniência.

O fluxo baseado em registros de proveniência evita o uso de todos os agentes no processo de análise. Essa estratégia é baseada na disponibilidade de uma base de conhecimento histórica para dar suporte às decisões do agente MONITOR. Essa base de conhecimento, armazenada a partir de uma abordagem de proveniência, permite ao MONITOR ter acesso às informações de analises anteriores e decidir pela sugestão direta ao agente ANALISADOR que o artefato analisado no momento já possui uma correlação de semelhança com artefatos já analisados.

Os registros de proveniência são armazenados em formato XML a partir de cada avaliação do agente ANALISADOR. Os conteúdos são registrados entre as *tags*, ou seja, à medida que o ANALISADOR encontra uma evidência baseada no catálogo, ele grava o conteúdo encontrado entre as *tags* indicadoras da conformidade com o catálogo. A estrutura XML criada para armazenar os dados de proveniência segue conforme o código:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<baseproveniencia tipo="RNF" objetivo="proveniência">
    <patternIdentificacao>Transparencia</patternIdentificacao>
      <patternDecomposicaoNivel_1>
        <idDecomposicaoNivel_1>Auditabilidade</idDecomposicaoNivel_1>
            <patternDecomposicaoNivel_2>
                <idDecomposicaoNivel_2>Explicacao</idDecomposicaoNivel_2>
                  <idGrupo>1</idGrupo>
                  <tituloGrupo>Descrever módulos, funções, termos e variáveis utilizadas</tituloGrupo>
                  <patternQuestoes>
                        <idQuestao>1.1</idQuestao>
                        <tituloQuestao>Os módulos, funções, termos e variáveis têm descrições
                                    suficientes?</tituloQuestao>
                        <patternAlternativas>
                              <idAlternativa>1.1.2</idAlternativa>
                              <tituloAlternativa> Utilizar descrições textuais explicativas, baseadas em critério
                                          técnico, ao longo do código fonte, para demonstrar sua relação
```

```
                                        com a especificação do software.</tituloAlternativa>
                        <variáveis>
                            <tagsVariável>passodeExecução</tagsVariável>
                        </variáveis>
                    </patternAlternativas>

                <idGrupo>2</idGrupo>
                <tituloGrupo>Fornecer Ajuda</tituloGrupo>
                <patternQuestoes>
                    <idQuestao>2.1</idQuestao>
                    <tituloQuestao>Existem fontes de informação que descrevem o software?
                        </tituloQuestao>
                    <patternAlternativas>
                        <idAlternativa>2.1.1</idAlternativa>
                        <tituloAlternativa> Possuir descrições comentadas com regras condicionais ou de
                                loop, em linguagem natural estruturada, que corresponda
                                à implementação de estruturas condicionais ou de
                                repetição.</tituloAlternativa>
                        <variáveis>
                            <tagsVariável>estruturaAlgoritmo</tagsVariável>
                        </variáveis>
                    </patternAlternativas>
            <patternDecomposicaoNivel_2>
            ...
            <idDecomposicaoNivel_2>Rastreabilidade</idDecomposicaoNivel_2>
                    <id/Grupo>4</idGrupo>
                    <tituloGrupo>Fazer Pré-Rastreabilidade</tituloGrupo>
                    <patternQuestoes>
                        <idQuestao>4.1</idQuestao>
                        <tituloQuestao> As fontes de informação utilizadas são rastreadas?</tituloQuestao>
                        <patternAlternativas>
                            <idAlternativa>4.1.2</idAlternativa>
                            <tituloAlternativa> Data em que a mensagem é gravada, registrada ou coletada.
                                    </tituloAlternativa>
                        <variáveis>
                            <tagsVariável>data do registro</tagsVariável>
                        </variáveis>
                    </patternAlternativas>
        </patternDecomposicaoNivel_1>

            ...
</baseproveniencia>
```

Nessa estrutura de base de conhecimento, formada a partir das heurísticas de análise de artefatos ao longo da execução, o SMA está interessado em armazenar quais *tags* responderam quais questões a partir do seu vínculo com as alternativas propostas no catálogo. Portanto, os grupos, questões e alternativas se mantém e *tags* de variáveis essenciais e sinônimos são adicionados para cada estrutura de grupo, questão e alternativa. A partir do momento em que são analisados diferentes tipos de *software* comparados ao catálogo, são identificadas quais *tags,* que historicamente, foram dadas como em conformidade.

4.4.3
Arquitetura de Implementação

A implementação dos agentes foi baseada em um mapeamento entre o modelo intencional, a arquitetura BDI e a codificação em Jadex [Braubach et al. 2003], conforme proposto em [Serrano e Leite 20011b]. A Figura 26 apresenta a proposta do mapeamento entre as diferentes camadas.

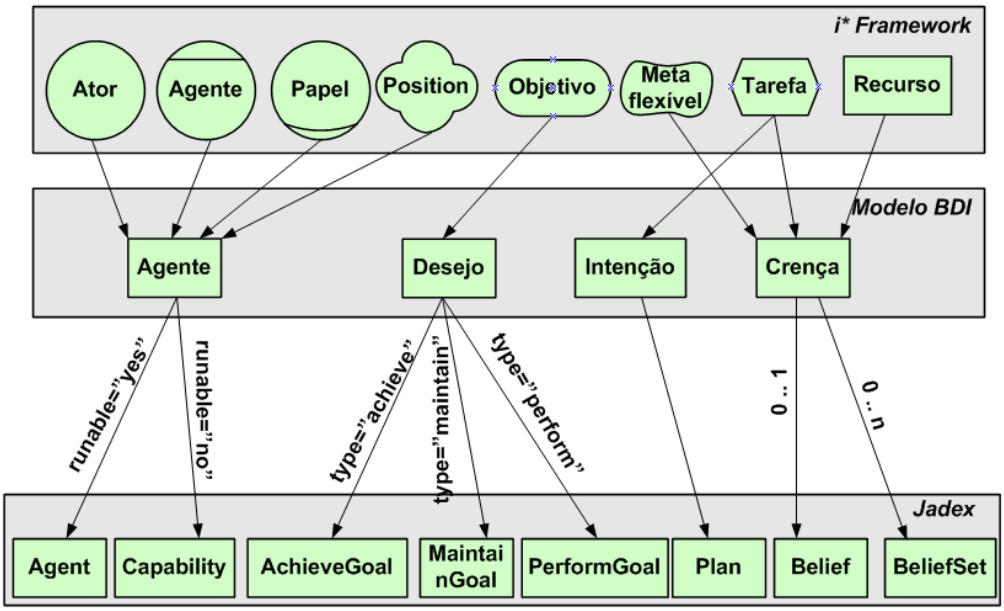


Figura 26: Mapeamento i* x BDI x Jadex [Serrano e Leite 2011b]

Em nossa proposta de implementação o CTS foi descrito nas crenças do agente ANALISADOR a partir dos *beliefs*, que passaram a indicar as questões dos patterns de transparência e os beliefset que tiveram o objetivo para indicar as alternativas. Portanto, o SMA desenvolvido não só utiliza das abordagens BDI, como também aplica um mapeamento desde a camada de modelo, abstração em i*, como segue em sua arquitetura de forma organizada ao levar em consideração tal mapeamento.

A especificação BDI dos agentes foi feita em uma arquitetura baseada em XML. O trecho de código da página subsequente apresenta um arcabouço da estrutura do agente ANALISADOR [Leal et al 2013a] baseado na proposta de [Serrano e Leite 20011b] do mapeamento i* para BDI e Jadex. Alguns trechos de código foram suprimidos devido à extensão do código fonte, mas a estrutura procura refletir a arquitetura utilizada na implementação.

A implementação visa a refletir as características do método sistêmico apresentadado no modelo intencional da arquitetura dos agentes do SMA. Apesar de

funcional, ela ainda é um esforço inicial de codificação para operacionalizar as questões de análise.

A implementação dos gentes é feita a partir de duas abordagens: a primeira a partir da implementação de estruturas XML para caracterizar os agentes, seus papeis, posições, relações, crenças, desejos e planos; a segunda, a partir de classes java que implementam o comportamento dos agentes.

Trechos de código das características da arquitetura dos agentes ANALISADOR e CONSOLIDADOR podem ser vistos como:

- Trecho de código da arquitetura do agente ANALISADOR.

```
<!--
        <H3>The Analyzer agent.</H3>
        This agent analyze the log for evaluate the RNF
-->

<agent xmlns="http://jadex.sourceforge.net/jadex"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                    http://jadex.sourceforge.net/jadex-0.96.xsd"
        name="AnalyzerAgent"
        package="jadex.examples.TMMAS.agent.analyzerAgent">

        <imports>
                <import>jadex.adapter.fipa.SFipa</import>
                <import>java.io.File</import>
                <import>handleClasses.CanonicalDefinition</import>
                <import>org.w3c.dom.Document</import>
        </imports>

        <beliefs>
                <belief name="stdPath" class="String">
                        <fact>"C:/Policy.xml"</fact>
                </belief>
                <belief name="policy" class="Document">
                        <fact>null</fact>
                </belief>
        </beliefs>

        <plans>
                <!-- Plan for analyze the log. -->
                <plan name="receiveLog">
                        <body class="ReceiveLogPlan"/>
                </plan>

                <!-- Plan for evaluate the RNF according to the operacionalizations. -->
                <plan name="evaluateRNFLog">
                        <body class="EvaluateRNFPlan"/>
                        <trigger>
                                <internalevent ref="call_EvaluateRNFPlan"/>
                        </trigger>
                </plan>
                  <!-- Plan for provenance registration. -->
```

```
                <plan name="provenanceRegister">
                        <body class="provenanceRegisterPlan"/>
                        <trigger>
                                <internalevent ref="call_provenanceRegisterPlan"/>
                        </trigger>
                </plan>

                <!-- Plan for canonize the different log patterns. -->
                <plan name="SendRNFEvaluation">
                        <body class="SendRNFEvaluationPlan"/>
                        <trigger>
                                <internalevent ref="call_SendRNFEvaluationPlan"/>
                        </trigger>
                </plan>

                <!-- Plan to save the Unknown Elements. -->
                <plan name="receiveExecTraces">
                        <parameter name="execTraces" class="Document">
                        <messageeventmapping ref="receiveExecTracesMSG.content"/>
                        </parameter>
                        <body class="EvaluateRNFPlan"/>
                        <trigger>
                                <messageevent ref="receiveExecTracesMSG"/>
                        </trigger>
                </plan>
        </plans>
```

- Trecho de código da arquitetura do agente CONSOLIDADOR.

```
<!--
        <H3>The Consolidate agent.</H3>

        This agent consolidates all the information handled in the process of RNF analyze
-->

<agent xmlns="http://jadex.sourceforge.net/jadex"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                http://jadex.sourceforge.net/jadex-0.96.xsd"
        name="ConsolidatorAgent"
        package="jadex.examples.TMMAS.agent.consolidatorAgent">

        <imports>
                <import>jadex.adapter.fipa.SFipa</import>
                <import>handleClasses.CanonicalDefinition</import>
                <import>handleClasses.OpsOfRNF</import>
        </imports>

        <plans>
                <!-- Plan for receive the log and register it in the belief base. -->
                <plan name="ReceiveRaEvCaMo">
                        <body class="ReceiveRaEvCaMoPlan"/>
                </plan>

                <plan name="ConsolidateInformation">
                        <body class="ConsolidateInformationPlan"/>
                        <trigger>
```

```
                    <internalevent ref="call_ConsolidateInformationPlan"/>
                </trigger>
            </plan>
        </plans>

        <events>
            <messageevent name="receiveCanonicModelMSG" type="fipa" direction="receive">
                <parameter name="performative" class="String" direction="fixed">
                    <value>SFipa.INFORM</value>
                </parameter>
                <parameter name="content-class" class="Class" direction="fixed">
                    <value>CanonicalDefinition.class</value>
                </parameter>
            </messageevent>

            <messageevent name="receiveRastEvaluationMSG" type="fipa"
                direction="receive">
                <parameter name="performative" class="String" direction="fixed">
                    <value>SFipa.INFORM</value>
                </parameter>
                <parameter name="content-class" class="Class" direction="fixed">
                    <value>OpsOfRNF.class</value>
                </parameter>
            </messageevent>

            <!-- This internal event means the call for the Send RNF Evaluation Plan.-->
            <internalevent name="call_ConsolidateInformationPlan"/>
        </events>
```

A implementação do comportamento dos agentes segue a codificação tradicional em linguagem Java com definição de classes, métodos. O trecho de código a seguir, representa uma parte de classes criadas para o agente ANALISADOR. Essas classes são derivadas dos planos definidos em XML da arquitetura do agente.

```
package jadex.examples.TMMAS.agent.analyzerAgent;


import jadex.examples.TMMAS.util.handleClasses.CanonicalDefinition;
import jadex.model.IMBelief;
import jadex.model.IMBeliefbase;
import jadex.runtime.IInternalEvent;
import jadex.runtime.IMessageEvent;
import jadex.runtime.Plan;


public class ReceiveLogPlan extends Plan {
    private static final long serialVersionUID = 1L;
    public void body() {
        System.out.println("Analyser: The Analyser agent is ready.");
        System.out.println("Analyser: Waiting for a Canonic log.");
```

```
                while (true) {
                        CanonicalDefinition CanonicModel = new CanonicalDefinition();
                        // This will be a new plan. It will receive a message from another
                        // agent and then make something
                        IMessageEvent msg =
                          waitForMessageEvent("receiveCanonicLogMSG");
                        System.out.println("Analyser: Log recebido.");
                        CanonicModel = (CanonicalDefinition) msg.getContent();// Get the
                            content of the message
                        // Create a new belief in run time to store the canonic log
                        IMBeliefbase model = (IMBeliefbase) getBeliefbase()
                                        .getModelElement();
                        IMBelief belief = model.createBelief("CanonicLog",
                              CanonicalDefinition.class, 0, "false");// Create the belief
                        System.out.println("Analyser: Belief Created.");
                        getBeliefbase().registerBelief(belief);// Register belief at runtime
                        System.out.println("Analyser: Belief Registreated.");
                        getBeliefbase().getBelief("CanonicLog").setFact(CanonicModel);// Set
                              the log in the new belief
                        // Internal event that represents the reception of the canonic log
                        IInternalEvent event =
                              createInternalEvent("call_EvaluateRNFPlan");
                        dispatchInternalEvent(event);
                }
        }
}
```


4.5
Considerações finais

A seção descrita apresenta a arquitetura sobre dois aspectos.

O primeiro, a partir da arquitetura utilizada para a concepção do catálogo de RNF. Nesse ponto foi apresentado o modelo conceitual com diagrama de classes dos diversos padrões utilizados no método. Esse serve de base para a concepção das estruturas do catálogo XML.

O segundo, teve o foco no detalhamento da arquitetura do SMA, ou seja, da intencionalidades dos agentes em suas interações, nesse caso apresentado pelos

Paineis de Intencionalidade; como também o modelo SR do i* para representar os agentes e suas decomposições em objetivos, tafefas, recursos utilizados, além da representação explícita de sua dependência de outros agentes. Após o modelo i* apresentado houve a preocupação nessa tese de a arquitetura da camada de implementação a partir do Mapeamento i* x BDI x Jadex.

A implementação da arquitetura foi feita a partir da aplicação do método em estudos de caso que envolveram cinco (explicação, disponibilidade, rastreabilidade, detalhamento, consistência) *softgoals* folha do SIG de Transparência. A aplicação do método com base na arquitetura proposta indicam que *softgoals* que sigam a mesma abordagem de modelagem (*goal-oriented*) dos cinco *softgoals* analisados serão passíveis de análise de conformidade a partir do SMA desenvolvido. Os estudos de caso serão apresentados na seção posterior.

5.
Estudo de Casos

Esse capítulo relata características operacionais da execução da aplicação do método sistêmico para análise de RNF. O capítulo apresenta a transição dos modelos conceituais propostos no processo até a sua operacionalização pelo SMA. O estudo de casos é feito a partir da aplicação do método sobre em quatro softwares reais que operam em ambiente de produção. O estudo de casos foi realizado sobre a Aplicação do Método no Código Fonte do C&L, nos Traços de Execução de Servidor Apache, nos Traços de Execução do LattesScholar e nos Artefatos da Arquitetura do Lattes Scholar.

5.1
Aplicação do Método no Código Fonte do *Software* C&L

5.1.1
Definição do Software C&L

O C&L (Cenários e Léxico) [Leite et al. 2000] é um sistema de *software* que utiliza a representação de cenários para facilitar a compreensão de domínios da aplicação. O sistema tem como objetivo criar um ambiente colaborativo para criação, edição, manutenção e evolução de léxicos e cenários, a partir da disponibilização de informações em linguagem natural semi-estruturada e organizadas de forma simples, de rápida acesso e inter-relacionadas a partir de hipertextos.

A estrutura do C&L é composta por elementos descritivos que expressam: o objetivo, o contexto, os recursos, os atores, os episódios, i.e., as ações, as exceções que ocorrem nestas atividades. O conjunto destas características representa uma situação ou cenário, além de reservar um atributos para descrição de restrições, recursos e episódios para refletir aspectos não-funcionais [Felicissimo et al. 2004].

O C&L segue o princípio de *software* livre com seu código fonte disponibilizado na internet[2]. A versão atual está implementada em Lua[3], com códigos comentados a partir de episódios criados na própria ferramenta.

[2] pes.inf.puc-rio.br/cel
[3] http://www.lua.org/portugues.html

5.1.2
Aplicação do Método para Análise de Código Fonte

A aplicação do método no *software* C&L foi feita a partir da análise do RNF de Auditabilidade, definido e presente no CTS proposto em [Grupo ER PUC-Rio 2013] e formalizado em [Cappelli 2009]. A definição de Auditabilidade de acordo com o CTS é *"Capacidade de ser identificada pela aferição de práticas que implementem características de validade, controlabilidade, verificabilidade, rastreabilidade e explicação"* [CTS 2013]. O estudo de caso pretende avaliar se há no *software* características que satisfazem tal RNF. As atividades do método foram aplicadas conforme abaixo:

A1 - Criar SIG

Uma série de qualidades é entrada para a A1, nesse caso, Auditabilidade, Validade, Controlabilidade, Verificabilidade, Rastreabilidade e Explicação. Ao final das atividades essas características passam a ser *softgoals* que necessitam ser satisfeitos para atender à Auditabilidade. As relações de Help indicam como são os elos de contribuição entre os *softgoals* folha em relação à Auditabilidade. O resultado final da A1, além da relação dos *softgoals*, é o SIG apresentado na Figura 27.

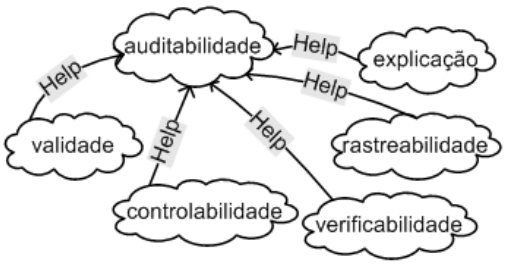


Figura 27: SIG Auditabilidade [Cappelli 2009].

Para efeito desse estudo de caso for escolhido o *sofgoal*: Explicação.

A2 – Definir *patterns*

Para o *Softgoal* Explicação

A definição de Explicação no CTS corresponde à "*Capacidade de informar a razão e o propósito do software e suas características*". A aplicação da A2 para o *softgoal* resulta em um detalhamento de grupos, questões e alternativas de respostas para as questões.

O *softgoal* é decomposto em questões que se respondidas satisfazem à meta definida. Relacionadas às questões as mesmas são agrupadas em grupos que melhor a representam. No caso, Explicação possui três grupos: 1) Descrever módulos, funções, termos e variáveis utilizadas; 2) Oferecer fontes alternativas de informação; e 3) Fornecer ajudas. Os grupos e as questões podem ser vistos conforme apresenta a Figura 28 no quadro Resultado. Tais detalhamentos são propostos no CTS [CTS 2013] e são definidos a partir da aplicação da **A2.1 Descrever questões** (descrever questões para *softgoals* folha, no caso Explicação).

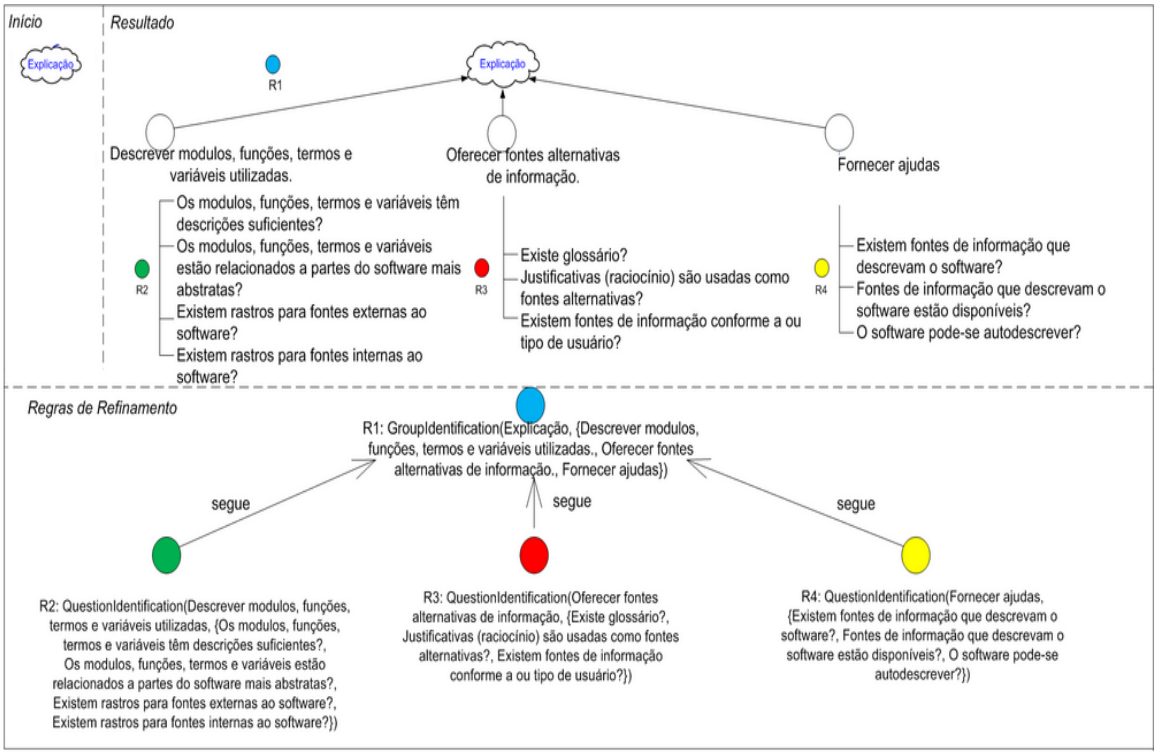


Figura 28: Patterns do *softgoal* Explicação [CTS 2013].

A partir da definição das questões foi aplicada a atividade **A2.2 Operacionalizar questões** da seção 3.4 Detalhamento de Atividades (Definir *patterns* A2) para a escolha de alternativas que possam satisfazer às questões. Nessa seção do estudo de caso, foram escolhidas algumas questões para que operacionalizações sejam identificadas.

As operacionalizações relatadas a seguir definidas a partir da aplicação da A2.2 foram elaboradas considerando código fonte do *software*. O grupo de questões e operacionalizações pode ser visto na Tabela 4. As operacionalizações foram retratadas de forma *ad-hoc* para efeito de estudo da aplicação do método, não necessariamente são estruturas que devem ser consideradas para reuso.

Tabela 4: Relação de grupos, questões e operacionalizações para o *softgoal* Explicação.

Grupo:	**1 Descrever módulos, funções, termos e variáveis utilizadas**
Questão 1:	1.1 Os módulos, funções, termos e variáveis têm descrições suficientes?
Operacionalizações:	1.1.1 Usar cabeçalhos de códigos fonte comentados a partir de critérios técnicos.
	1.1.2 Usar comentários nas funções para demonstrar sua relação com a especificação do *software*.
Grupo:	**2 Fornecer ajudas**
Questão 1:	2.1 Existem fontes de informação que descrevem o *software*?
Operacionalizações:	2.1.1 Possuir descrições comentadas com regras condicionais ou de *loop*, em linguagem natural estruturada, que corresponda à implementação de estruturas condicionais ou de repetição.

A3 – Configurar XML

→ Configuração de grupos, questões e operacionalizações:

Definidas as estruturas anteriormente descritas, faz-se necessária sua transposição para uma arquitetura padronizada em XML definida como catálogo XML RNF (APÊNDICE A) que servem como regras para os agentes do SMA. Sua transcrição é feita a partir das atividades propostas do detalhamento da A3 Configurar XML. Suas sub-atividades descritas como A3.1 Configurar *Objetive patterns* (estruturas XML), A3.2 Configurar *Questions patterns* (estruturas XML), A3.3 Configurar *Alternative patterns* (estruturas XML) e A3.4 Configurar Variáveis essenciais e seus sinônimos (estruturas XML) serão utilizadas para a composição da estrutura XML do catálogo de RNF que servirá como regra para as ações dos agentes do SMA.

O catálogo XML para as estruturas definidas até o momento segue conforme apresentado a seguir:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<catalogo tipo="RNF" objetivo="análise">
  <versao>1</versao>
      <patternObjetivos>
          <patternIdentificacao>Transparencia</patternIdentificacao>

          <patternDecomposicaoNivel_1>
                <idDecomposicaoNivel_1>Auditabilidade</idDecomposicaoNivel_1>
                <patternSelecao>HELP</patternSelecao>

                  ... (trecho suprimido propositalmente)
```

```
                    <patternDecomposicaoNivel_2>
                        <idDecomposicaoNivel_2>Explicacao</idDecomposicaoNivel_2>
                        <patternSelecao>HELP</patternSelecao>

                        <patternGrupo>
                            <idGrupo>1</idGrupo>
                            <tituloGrupo>Descrever módulos, funções, termos e variáveis utilizadas</tituloGrupo>

                            <patternQuestoes>
                                <idQuestao>1.1</idQuestao>
                                <tituloQuestao>Os módulos, funções, termos e variáveis têm descrições suficientes?
                                </tituloQuestao>

                                    <patternAlternativas>
                                        <idAlternativa>1.1.1</idAlternativa>
                                        <tituloAlternativa>Usar cabeçalhos de códigos fonte comentados a partir
                                            de critérios técnicos.</tituloAlternativa>
                                        <patternSelecao>RESPONDE</patternSelecao>
                                        <variaveisEssenciais>
                                            <idVariaveisEssenciais></idVariaveisEssenciais>
                                            <nocaoVariaveisEssenciais>                </nocaoVariaveisEssenciais>
                                            <sinonimosVariaveisEssenciais>
                                                <idSinonimosVariaveisEssenciais>
                                                </idSinonimosVariaveisEssenciais>
                                            </sinonimosVariaveisEssenciais>
                                        </variaveisEssenciais>

                                        <idAlternativa>1.1.2</idAlternativa>
                                        <tituloAlternativa>Usar comentários nas funções para demonstrar
                                            sua relação com a especificação do software.</tituloAlternativa>
                                        <patternSelecao>RESPONDE</patternSelecao>

                                        ... (trecho suprimido propositalmente)

                                    </patternAlternativas>

                            </patternQuestoes>
                        </patternGrupo>

                        <patternGrupo>
                            <idGrupo>2</idGrupo>
                            <tituloGrupo>Fornecer ajudas</tituloGrupo>

                            <patternQuestoes>
                                <idQuestao>2.1</idQuestao>
                                <tituloQuestao>Existem fontes de informação que descrevem o software
                                    </tituloQuestao>

                                    <patternAlternativas>
                                        <idAlternativa>2.1.1</idAlternativa>
                                        <tituloAlternativa> Possuir descrições comentadas com regras condicionais
                                        ou de loop, em linguagem natural estruturada, que corresponda à
                                        implementação de estruturas condicionais ou de repetição.</tituloAlternativa>
                                        <patternSelecao>RESPONDE</patternSelecao>

                                        ... (trecho suprimido propositalmente)

                                    </patternAlternativas>
                            </patternQuestoes>
                        </patternGrupo>
                    </patternDecomposicaoNivel_2>

                    ... (trecho suprimido propositalmente)

                </patternDecomposicaoNivel_1>
            </patternObjetivos>
        </catalogo>
```

→ Configuração de variáveis essenciais, sinônimos e padrões:

Após as execuções das atividades 3.1 a 3.3 é necessário executar A3.4 Configurar Variáveis para que se estabeleçam marcadores vinculados às operacionalizações para que as mesmas possam ser identificadas no artefato monitorado. O uso de marcadores é importante uma vez que a análise se dá em dois contextos:

A) Contexto de análise automática de sintaxe nos artefatos análisados - o SMA tem bases de conhecimento estabelecidas em catálogo que proporcionam a um algoritmo interno o reconhecimento de conformidades a partir de comparação das sintaxe do artefato analisado e do catálogo;

B) Contexto de análise a partir de interação humana - uma vez que não existam bases de conhecimento estabelecidas para as análises sintáticas, o SMA identifica no artefato analisado os marcadores e indica os pontos que podem ter análise humana para avaliação de conformidade.

A aplicação da atividade **A3.4 Configurar Variáveis essenciais e seus sinônimos** é feita inicialmente para o Grupo 1, Questão 1.1 e Operacinalização 1.1.1, bem como a atividade **A3.5 Configurar Padrões** deram origem às descritas conforme o resultado apresentado a seguir.

Tabela 5: Inserção de Variáveis Essenciais, Sinônimos e Padrões para o *softgoal* Explicação.

Grupo:	**1 Descrever módulos, funções, termos e variáveis utilizadas**
Questão 1:	1.1 Os módulos, funções, termos e variáveis têm descrições suficientes?
Operacionalização:	1.1.1 Utilizar cabeçalhos de códigos fonte comentados a partir de critérios técnicos.
Variável Essencial:	comentáriodeCabeçalho
Noção:	Descrição textual que consiste na parte que contém informações suplementares e explicativas colocadas no começo de um artefato de código fonte, no início de declaração de funções (métodos, procedimentos).
Sinônimos:	cabeçalho, notadeCabeçalho
Padrão:	@Titulo: % @Objetivo: % @Contexto: % @Localização: % @Pré-condição: % @Atores: % @Recursos: %

Para o caso acima, o padrão foi definido com um metadado de acordo com a estrutura do *software* C&L para descrição de cenários. O padrão descrito uma linha após a outra é pesquisado no texto com tal formatação. O símbolo de percentual indica que pode haver qualquer outro texto após o identificador. A variável essencial e sinônimos foram criados do UdI do contexto de grupo, questão e operacionalização a partir da sua arquitetura baseada em Léxico. Sua estruturação em XML segue como apresentado abaixo:

```
... (trecho suprimido propositalmente)
<patternDecomposicaoNivel_2>
    <idDecomposicaoNivel_2>Explicacao</idDecomposicaoNivel_2>
    <patternSelecao>HELP</patternSelecao>

    <patternGrupo>
        <idGrupo>1</idGrupo>
        <tituloGrupo>Descrever módulos, funções, termos e variáveis utilizadas</tituloGrupo>

        <patternQuestoes>
            <idQuestao>1.1</idQuestao>
            <tituloQuestao>Os módulos, funções, termos e variáveis têm descrições suficientes?</tituloQuestao>

                <patternAlternativas>
                    <idAlternativa>1.1.1</idAlternativa>
                    <tituloAlternativa>Usar cabeçalhos de códigos fonte comentados a partir de critérios técnicos.
                        </tituloAlternativa>
                    <patternSelecao>RESPONDE</patternSelecao>
                    <variaveisEssenciais>
                        <idVariaveisEssenciais>comentáriodeCabeçalho</idVariaveisEssenciais>
                        <nocaoVariaveisEssenciais> Descrição textual que consiste na parte que contém
                        informações suplementares e explicativas colocadas no começo de um bloco de dados.
                        </nocaoVariaveisEssenciais>
                        <sinonimosVariaveisEssenciais>
                            <idSinonimosVariaveisEssenciais>cabeçalho</idSinonimosVariaveisEssenciais>
                            <idSinonimosVariaveisEssenciais>notadeCabeçalho </idSinonimosVariaveisEssenciais>
                        </sinonimosVariaveisEssenciais>

                        <padrao>
                            <idPadrao1>@Titulo:%</ idPadrao1>
                            <idPadrao2>@Objetivo:%</ idPadrao2>
                            <idPadrao3>@Contexto:%</ idPadrao2>
                            <idPadrao4>@Localização:% </ idPadrao2>
                            <idPadrao5>@Pré-condição:%</ idPadrao2>
                            <idPadrao6>@Atores:%</ idPadrao2>
                            <idPadrao7>@Recursos:%</ idPadrao2>
                        </padrao>
                    </variaveisEssenciais>
... (trecho suprimido propositalmente)
```

Nesse ponto é configurada a estrutura XML para o Grupo 1, Questão 1.1 e Operacionalização 1.1.2, a lógica de criação segue a mesma descrita anteriormente e são descritas conforme o resultado apresentado a seguir.

Tabela 6: Inserção de Variáveis Essenciais, Sinônimos e Padrões para o *softgoal* Explicação-Passo de Execução.

Grupo:	**1 Descrever módulos, funções, termos e variáveis utilizadas**
Questão 1:	1.1 Os módulos, funções, termos e variáveis têm descrições suficientes?
Operacionalização:	1.1.2 Utilizar descrições textuais explicativas, baseadas

	em critério técnico, ao longo do código fonte, para demonstrar sua relação com a especificação do *software*.
Variável Essencial:	fluxodeExecução
Noção:	Descrição textual que consiste na parte que contém as informações suplementares e explicativas colocadas ao longo do código fonte para descrever o passo a passo de execução.
Sinônimos:	passodeExecução, episódiodeExecução
Padrão:	Não se aplica.

Configuração do XML no catálogo:

```
... (trecho suprimido propositalmente)
<patternDecomposicaoNivel_2>
    <idDecomposicaoNivel_2>Explicacao</idDecomposicaoNivel_2>
    <patternSelecao>HELP</patternSelecao>

    <patternGrupo>
        <idGrupo>1</idGrupo>
        <tituloGrupo>Descrever módulos, funções, termos e variáveis utilizadas</tituloGrupo>

        <patternQuestoes>
            <idQuestao>1.1</idQuestao>
            <tituloQuestao>Os módulos, funções, termos e variáveis têm descrições suficientes?</tituloQuestao>

                <patternAlternativas>
                    <idAlternativa>1.1.2</idAlternativa>
                    <tituloAlternativa> Utilizar descrições textuais explicativas, baseadas em critério técnico, ao
                                        longo do código fonte, para demonstrar sua relação com a especificação
                                        do software.
                     </tituloAlternativa>
                    <patternSelecao>RESPONDE</patternSelecao>
                    <variaveisEssenciais>
                        <idVariaveisEssenciais> fluxodeExecução </idVariaveisEssenciais>
                        <nocaoVariaveisEssenciais> Descrição textual que consiste na parte que contém as
                            informações suplementares e explicativas colocadas ao longo do código
                            fonte para descrever o passo a passo de execução.
                        </nocaoVariaveisEssenciais>
                        <sinonimosVariaveisEssenciais>
                            <idSinonimosVariaveisEssenciais>passodeExecução</idSinonimosVariaveisEssenciais>
                            <idSinonimosVariaveisEssenciais>episódiodeExecução</idSinonimosVariaveisEssenciais>
                        </sinonimosVariaveisEssenciais>

                        <padrao>
                            <idPadrao1>não se aplica</ idPadrao1>
                        </padrao>
                    </variaveisEssenciais>
... (trecho suprimido propositalmente)
```

A configuração a seguir corresponde a estrutura XML para o Grupo 2, Questão 2.1 e Operacionalização 2.1.1, a lógica de criação segue a mesma descrita anteriormente e são descritas conforme o resultado apresentado a seguir.

Tabela 7: Inserção de Variáveis Essenciais, Sinônimos e Padrões para o *softgoal* Explicação-Fornecer Ajudas.

Grupo:	**2 Fornecer ajudas**
Questão 1:	2.1 Existem fontes de informação que descrevem o *software*?
Operacionalização:	2.1.1 Possuir descrições comentadas com regras condicionais ou de *loop*, em linguagem natural estruturada, que corresponda à implementação de estruturas condicionais ou de repetição.
Variável Essencial:	estruturaAlgoritmo
Noção:	Descrição em linguagem natural estruturada explicativa que corresponda às estruturas condicionais ou de repetição que serão implementadas nos algoritmos do código fonte.
Sinônimos:	Não se aplica
Padrão:	if%then, if%else, case%

Para os casos de padrões separados por vírgula, estabelece-se que o SMA fará uma pesquisa por um dos elementos da lista. O símbolo de percentual indica que poderá haver qualquer outro texto inserido entre os elementos do padrão. Configuração do XML no catálogo:

```
... (trecho suprimido propositalmente)
<patternDecomposicaoNivel_2>
      <idDecomposicaoNivel_2>Explicacao</idDecomposicaoNivel_2>
      <patternSelecao>HELP</patternSelecao>

      <patternGrupo>
            <idGrupo>2</idGrupo>
            <tituloGrupo>Fornecer ajudas</tituloGrupo>

            <patternQuestoes>
                  <idQuestao>2.1</idQuestao>
                  <tituloQuestao> Existem fontes de informação que descrevem o software?</tituloQuestao>

                        <patternAlternativas>
                              <idAlternativa>2.1.1</idAlternativa>
                              <tituloAlternativa> Possuir descrições comentadas com regras condicionais ou de loop,
                                    em linguagem natural estruturada, que corresponda à
                                    implementação de estruturas condicionais ou de repetição.
                               </tituloAlternativa>
                              <patternSelecao>RESPONDE</patternSelecao>
                              <variaveisEssenciais>
                                    <idVariaveisEssenciais> estruturaAlgoritmo </idVariaveisEssenciais>
                                    <nocaoVariaveisEssenciais> Descrição em linguagem natural estruturada
                                          explicativa que corresponda à estruturas
                                          condicionais ou de repetição que serão implementadas
                                          nos algoritmos do código fonte.
                                    </nocaoVariaveisEssenciais>
                                    <sinonimosVariaveisEssenciais>
                                          <idSinonimosVariaveisEssenciais> não se aplica </idSinonimosVariaveisEssenciais>
                                    </sinonimosVariaveisEssenciais>

                                    <padrao>
                                          <idPadrao1>if%then, if%else, case% </ idPadrao1>
                                    </padrao>
                              </variaveisEssenciais>
... (trecho suprimido propositalmente)
```

A4 – Configurar *Software*

Os trechos de código exibidos nessa seção foram extraídos do arquivo *modelo_projeto.lua* presente na camada de modelo (cel_B\modelo\) do *software* C&L.

→ Configuração do *software* levando-se em consideração a operacionalização descrita no catálogo de RNF no item 1.1.1 Utilizar cabeçalhos de códigos fonte comentados a partir de critérios técnicos. Tal configuração é feita pela inserção de *tags* do catálogo XML no código fonte do C&L.

Para o trecho de código a seguir foi incluída a variável essencial *[&comentáriodeCabeçalho]*. O @, utilizado nas marcações no código fonte a seguir, é utilizado para identificação automatica pelo SMA. Apesar da variável essencial inserida, para esse código, há uma análise automática do conteúdo a partir do agente do SMA uma vez que foi determinado conteúdos para a *tag* Padrão no catálogo XML de RNF e a posterior análise automática de conformidade.

O código representa uma função para seleção de um determinado projeto no *software* C&L. É utilizado para pesquisa do projeto e retorna como parâmetro para outra função os dados do projeto para posterior exibição.

```
--[[ ================================================================================
<comentáriodeCabeçalho>
@Titulo: SELECIONAR PROJETO
@Objetivo: Obter as informações sobre um projeto específico
@Contexto:
        @ Localização: camada de modelo.
        @Pré-condição: CONTROLE CONSULTA DADOS DO PROJETO
@Atores: sistema.
@Recursos: id do projeto, consulta_bd.lua.
</comentáriodeCabeçalho>
]]--
function selecionar_projeto(id_projeto)

--@Episódio 1: Montar consulta para seleção com o o id do projeto.
        local consulta_projeto_id = ("SELECT PR.NOME AS NOME_PROJETO, PR.DESCRICAO, PR.DATA_INCLUSAO,
PR.CASE_SENSITIVE, USR.NOME, USR.SOBRENOME"
                ..." FROM projeto PR, usuario USR"
                ..." WHERE PR.ID_USUARIO = USR.ID_USUARIO AND PR.ID_PROJETO=\""..id_projeto.."\"")

--@Episódio 2: EXECUTAR CONSULTA DE SELEÇÃO NO BANCO DE DADOS para selecionar projeto por id no banco de
dados.
        local resultado_consulta, dados_projeto = consulta_bd.executar_consulta_selecao(consulta_projeto_id)

--@Episódio 3: Retorna informações do projeto
        return dados_projeto
end
```

→ Configuração do *software* levando-se em consideração a operacionalização descrita no catálogo de RNF no item 1.1.2 Utilizar descrições textuais explicativas, baseadas em critério técnico, ao longo do código fonte, para demonstrar sua relação com a especificação do *software*.

No trecho de código a seguir foram utilizados os sinônimos presente no catálogo. É utilizado o sinônimo *[&passodeExecução]* para reconhecimento pelo SMA e a partir daí apresentada a tela com o código reconhecido para avaliação de conformidade por interação humana.

```
--[[ ==========================================================================
@Titulo: CADASTRAR COLABORADOR PROJETO
@Objetivo: Incluir um colaborador em um projeto cadastrado no banco de dados.
@Contexto:
        - Localização: camada de modelo
        - Pré-condição: CONTROLE CADASTRA DADOS DO PROJETO.
@Atores: sistema.
@Recursos: id do usuário, id do projeto, id_autorizacao e tipo acesso.
]]--
function cadastrar_colaborador_projeto(id_usuario, id_projeto, tipo_acesso, id_usuario_autorizou)

--<passodeExecução>
--@Episódio 1: Obtém do sistema a data em que esta sendo feito o cadastro.
--</passodeExecução>

        local data_inclusao_colaborador = os.date("%Y-%m-%d %H:%M:%S");

--<passodeExecução>
--@Episódio 2: Montar consulta para inserção com os dados do colaborador.
--</passodeExecução>

        local consulta_insercao_colaborador = ("INSERT INTO usuario_projeto (ID_USUARIO, ID_PROJETO,
TIPO_ACESSO, ID_USUARIO_AUTORIZOU, DATA_INCLUSAO, FLAG_EXCLUSAO)"
                .."VALUES ("..id_usuario..","..id_projeto..","..tipo_acesso..",\""..id_usuario_autorizou
                .."\",\""..data_inclusao_colaborador.."\",1)")

--<passodeExecução>
--@Episódio 3: EXECUTAR CONSULTA DE INSERÇÃO NO BANCO DE DADOS para cadastrar o colaborador no projeto.
--</passodeExecução>

        local resultado_insercao_colaborador, id_colaborador =
consulta_bd.executar_consulta_insercao(consulta_insercao_colaborador)

--[&passodeExecução]
--@Episódio 4: Retorna o resultado da inserção de colaborador.
        return resultado_insercao_colaborador
end
```

→ Configuração do *software* levando-se em consideração a operacionalização descrita no catálogo de RNF no item 2.1.1 Possuir descrições comentadas com regras condicionais ou de loop, em linguagem natural estruturada, que corresponda à implementação de estruturas condicionais ou de repetição.

```
--[[ ==========================================================================================
@Titulo: ALTERAR SÍMBOLO DO LÉXICO
@Objetivo: Alterar os dados de um símbolo do léxico cadastrado no sistema,
@Contexto:
        - Localização: camada de modelo.
        - Pré-condição: CONTROLE ALTERA DADOS DO LÉXICO
```

```
@Atores: sistema.
@Recursos: dados do léxico e módulo consulta_bd.
]]--
function alterar_lexico(dados_lexico)

--@Episódio 1: Montar consulta para alteração com os dados do símbolo do léxico.
        local consulta_alteracao_lexico = "UPDATE lexico SET NOME = \""..dados_lexico["nome"].."\", NOCAO = \""..dados_lexico["nocao"] .."\", IMPACTO = \""..dados_lexico["impacto"].."\","..."CLASSIFICACAO = \""..dados_lexico["classificacao"] .."\" WHERE ID_LEXICO = \""..dados_lexico["id_lexico"].."\""

--@Episódio 2: EXECUTAR CONSULTA DE ALTERAÇÃO NO BANCO DE DADOS para atualizar símbolo do léxico.
        local result_alt_lex = consulta_bd.executar_consulta_atualizacao(consulta_alteracao_lexico)

        if (result_alt_lex) then

--@Episódio 3: Montar consulta para alteração com os dados do sinônimo.
        local consulta_alteracao_sin = "UPDATE sinonimo SET FLAG_EXCLUSAO = 0 WHERE ID_LEXICO = \""..dados_lexico["id_lexico"].."\""

<estruturaAlgoritmo>
--@Episódio 4: Se o símbolo do léxico foi alterado com sucesso, então EXECUTAR CONSULTA DE ALTERAÇÃO NO BANCO DE DADOS para remover os sinônimos do símbolo do banco de dados.
</estruturaAlgoritmo>

        local result_rmv_sin_lex = consulta_bd.executar_consulta_atualizacao(consulta_alteracao_sin)

        if (result_rmv_sin_lex) then

<estruturaAlgoritmo>
--@Episódio 5: Se o cenário foi removidocom sucesso, então obtém do sistema a data em que está sendo feito o cadastro.
</estruturaAlgoritmo>

          local data_inclusao_sinonimo = os.date("%Y-%m-%d %H:%M:%S")

--@Episódio 6: Verifica o tipo da variavel sinonimos. Se o tipo for nil é porque não foi inserido nenhum sinônimo, se o tipo for string é porque apenas um sinônimo foi inserido,
        --neste caso inserimos este sinonimo em uma tabela.
          local sinonimos = dados_lexico["sinonimos"]

            if (sinonimos ~= nil) then
                local tabela_sinonimos = {}
                if (type(sinonimos) == "string") then
                        table.insert(tabela_sinonimos, sinonimos)
                else
                        tabela_sinonimos = sinonimos
                end
<estruturaAlgoritmo>
--@Episódio 7: Para cada sinônimo da lista de sinônimo, faça converter o nome do sinônimo de UTF-8 para ISO-8859-1.
</estruturaAlgoritmo>

                local cd = iconv.new("ISO-8859-1","UTF-8")
                for index, nome_sinonimo in pairs(tabela_sinonimos) do
                        local nome_iso88591 = cd:iconv(nome_sinonimo)

--@Episódio 8: Montar consulta para inserção com os dados do sinônimo.
                        local consulta_insercao_sin = "INSERT INTO sinonimo (NOME,ID_LEXICO, DATA_INCLUSAO, FLAG_EXCLUSAO) VALUES (\""..nome_iso88591
        .."\",\""..dados_lexico["id_lexico"].."\",\""..data_inclusao_sinonimo.."\", 1 )"

--@Episódio 9: EXECUTAR CONSULTA DE INSERÇÃO NO BANCO DE DADOS para cadastrar sinônimo no banco de dados.
                        local result_cad_sin_lex, id_sin = consulta_bd.executar_consulta_insercao(consulta_insercao_sin)
        ... (trecho suprimido propositalmente)
```

A5 – Operacionalizar Agentes

A execução dos agentes se dá pelas atividades A5.1, A5.2 e A5.3, basicamente as atividades representam o funcionamento desde a captura do artefato até sua análise a partir da comparação de marcações ou conteúdos com as regras definidas no catálogo. Ao ser executado, o Jadex apresenta uma tela básica que informa que os agentes estão em operação. A Figura 29 apresenta tal tela.

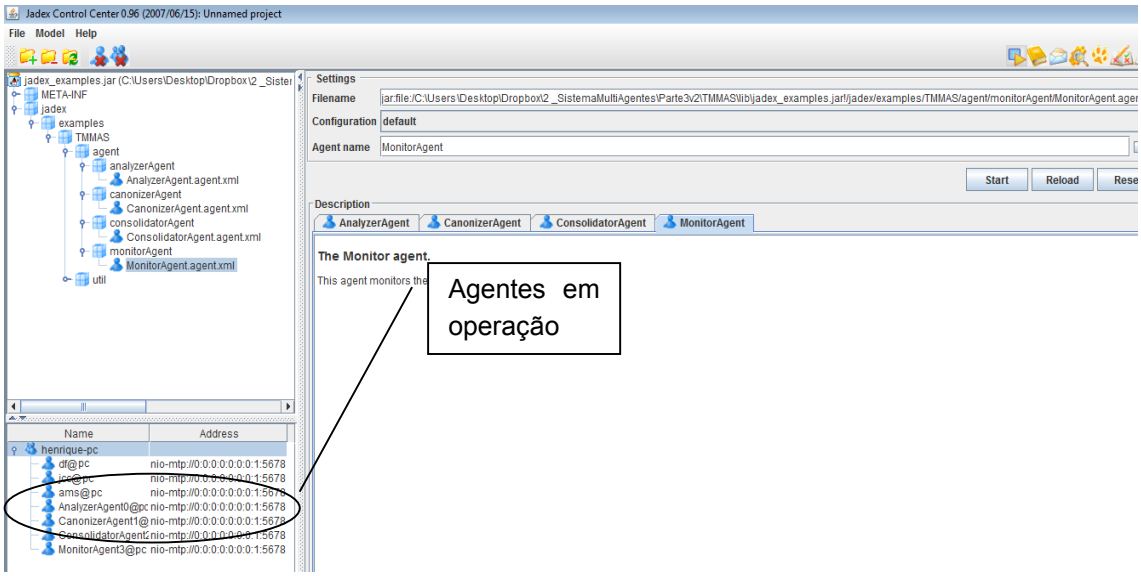


Figura 29: Tela básica do Jadex com os agentes em operação.

A importância da apresentação da tela básica de funcionamento do Jadex está relacionada a não exibição dos resultados da execução à medida que os agentes operam. Como foi desenvolvido, o SMA só apresenta os resultados da execução em um arquivo XML que contém estruturas que indicam quais os *softgoals*, as questões e alternativas foram satisfeitas pela aplicação do método. A segunda linha do arquivo <report tipo=”RNF” objetivo=”ANÁLISE”> indica que o arquivo é um artefato de saída que contem a análise de RNF.

São apresentados três *reports.* Dois para a análise da alternativa “1.1.2 Utilizar descrições textuais explicativas, baseadas em critério técnico, ao longo do código fonte, para demonstrar sua relação com a especificação do *software*” e um para a análise da alternativa “2.1.1 Possuir descrições comentadas com regras condicionais ou de loop, emlinguagem natural estruturada, que corresponda à implementação de estruturas condicionais ou de repetição.” No primeiro *report* é apresentada a análise para a *tagPadrão* e nos dois outros

reports são apresentados os resultados da análise para *tags* presentes nas variáveis essenciais que foram configuradas no catálogo XML.

Alternativa: 1.1.1 Usar cabeçalhos de códigos fonte comentados a partir de critérios técnicos

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<report tipo="RNF" objetivo="ANÁLISE">
    <artefato>modulo_projeto.lua</artefato>
    <local>cel_B\modelo</local>
     <patternIdentificacao>Transparencia</patternIdentificacao>
        <patternDecomposicaoNivel_1>
            <idDecomposicaoNivel_1>Auditabilidade</idDecomposicaoNivel_1>
                <patternDecomposicaoNivel_2>
                    <idDecomposicaoNivel_2>Explicacao</idDecomposicaoNivel_2>
                       <idGrupo>1</idGrupo>
                       <tituloGrupo>Descrever módulos, funções, termos e variáveis utilizadas</tituloGrupo>
                       <patternQuestoes>
                            <idQuestao>1.1</idQuestao>
                            <tituloQuestao>Os módulos, funções, termos e variáveis têm descrições
                                        suficientes?</tituloQuestao>
                         <patternAlternativas>
                              <idAlternativa>1.1.1</idAlternativa>
                              <tituloAlternativa>Usar cabeçalhos de códigos fonte comentados a partir de critérios
                                            técnicos.</tituloAlternativa>
                             <situação>EM CONFORMIDADE</situação>
                             <variáveis>
                                <tagsVariável>comentáriodeCabeçalho</tagsVariável>
                             </variáveis>
                             <padrões>
                                  <tagPadrao>@Titulo:% </tagPadrao>
                                  <tagPadrao>@Objetivo:% </tagPadrao>
                                  <tagPadrao>@Contexto:% </tagPadrao>
                                  <tagPadrao>@Localização:% </tagPadrao>
                                  <tagPadrao>@Pré-condição: %</tagPadrao>
                                  <tagPadrao>@Atores:% </tagPadrao>
                                  <tagPadrao>@Recursos:% </tagPadrao>
                            </padrões>
                          </patternAlternativas>
                <patternDecomposicaoNivel_2>
            </patternDecomposicaoNivel_1>
</report>
```

Alternativa: 1.1.2 Utilizar descrições textuais explicativas, baseadas em critério técnico, ao longo do código fonte, para demonstrar sua relação com a especificação do software.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<report tipo="RNF" objetivo="ANÁLISE">
    <artefato>modelo_projeto.lua</artefato>
    <local>cel_B\modelo</local>
     <patternIdentificacao>Transparencia</patternIdentificacao>
        <patternDecomposicaoNivel_1>
            <idDecomposicaoNivel_1>Auditabilidade</idDecomposicaoNivel_1>
                <patternDecomposicaoNivel_2>
                    <idDecomposicaoNivel_2>Explicacao</idDecomposicaoNivel_2>
                       <idGrupo>1</idGrupo>
                       <tituloGrupo>Descrever módulos, funções, termos e variáveis utilizadas</tituloGrupo>
                       <patternQuestoes>
                            <idQuestao>1.1</idQuestao>
                            <tituloQuestao>Os módulos, funções, termos e variáveis têm descrições
                                        suficientes?</tituloQuestao>
                         <patternAlternativas>
                              <idAlternativa>1.1.2</idAlternativa>
                              <tituloAlternativa> Utilizar descrições textuais explicativas, baseadas em critério técnico, ao
                                            longo do código fonte, para demonstrar sua relação com a especificação
```

```
                                        do software.</tituloAlternativa>
                                    <situação>EM CONFORMIDADE</situação>
                                    <variáveis>
                                        <tagsVariável>passodeExecução</tagsVariável>
                                    </variáveis>
                                    <padrões>
                                        <tagPadrao> </tagPadrao>
                                    </padrões>
                                </patternAlternativas>
            <patternDecomposicaoNivel_2>
        </patternDecomposicaoNivel_1>
</report>
```

Alternativa: 2.1.1 Possuir descrições comentadas com regras condicionais ou de loop, emlinguagem natural estruturada, que corresponda à implementação de estruturas condicionais ou de repetição.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<report tipo="RNF" objetivo="ANÁLISE">
    <artefato>modelo_lexico.lua</artefato>
    <local>cel_B\modelo</local>
    <patternIdentificacao>Transparencia</patternIdentificacao>
        <patternDecomposicaoNivel_1>
            <idDecomposicaoNivel_1>Auditabilidade</idDecomposicaoNivel_1>
                <patternDecomposicaoNivel_2>
                    <idDecomposicaoNivel_2>Explicacao</idDecomposicaoNivel_2>
                        <idGrupo>2</idGrupo>
                        <tituloGrupo>Fornecer Ajuda</tituloGrupo>
                        <patternQuestoes>
                            <idQuestao>2.1</idQuestao>
                            <tituloQuestao>Existem fontes de informação que descrevem o software?</tituloQuestao>
                            <patternAlternativas>
                                <idAlternativa>2.1.1</idAlternativa>
                                <tituloAlternativa> Possuir descrições comentadas com regras condicionais ou de loop, em
                                                    linguagem natural estruturada, que corresponda à implementação de
                                                    estruturas condicionais ou de repetição.</tituloAlternativa>
                                <situação> EM CONFORMIDADE </situação>
                                <variáveis>
                                    <tagsVariável>estruturaAlgoritmo</tagsVariável>
                                </variáveis>
                                <padrões>
                                    <tagPadrao>if%then</tagPadrao>
                                </padrões>
                            </patternAlternativas>
                <patternDecomposicaoNivel_2>
            </patternDecomposicaoNivel_1>
</report>
```

No caso apresentado, um artefato contém características que atendem às duas alternativas do catálogo representados pelas alternativas 1.1.2 e 2.1.1. Na primeira situação, o SMA fez a pesquisa pela *tag* <*comentáriodeCabeçalho*> e como existiam padrões a serem avaliados, o agentes analisaram um a um sendo que são dispostos no catálogo em linhas de estruturas de XML distintas, portanto, avaliados em conjunto. Já na segunda, os agentes fizeram a avaliação apenas pela variável essencial indicando que o artefato está em conformidade com as regras estabelecidas no catálogo para a *tag* indicada em <*idVariaveisEssenciais*>. O terceiro caso feita a análise do artefato *modelo_lexico.lua*, presente na pasta cel_B\modelo\, e o SMA indica que tal artefato está em conformidade com as regras estabelecidas

para a alternativa 2.1.1 pela evidência encontrada de *tags* conforme apresentadas nas relações de <*variáveis*> e <*padrões*>.

O arquivo XML a seguir apresenta a relação de arquivos analisados pelo SMA e que não possuem estruturas conhecidas registradas e catálogo. Os arquivos analisados estão presentes conforme configurado na marcação <local>. Os arquivos não apresentaram confirmidades, uma vez que nos mesmos não foram inseridas as marcações correspondentes ao catálogo.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<report tipo="RNF" objetivo="ANÁLISE">
    <local>cel_B\modelo</local>
        <situação>NÃO ATENDE
            <artefato>modelo_cenario.lua</artefato>
            <artefato>modelo_usuario.lua</artefato>
            <artefato>_teste_modelo.lua</artefato>
        </situação>
</report>
```

Outro tipo de análise feita pelo SMA está relacionada às assinaturas identificadas no catálogo em *Variável Essencial*, *Sinônimos* e *Padrão*. Primeiramente com relação à *Variável Essencial* e *Sinônimos*. As variáveis essenciais e sinônimos identificadas no catálogo devem vir precedidas e finalizadas por caracteres do tipo “<” e “>”, respectivamente. O SMA ao encontrar ocorrências do que estiver definido no catálogo em *Variável Essencial* e *Sinônimos*, mas que não contiverem os delimitadores “<” e “>” são registrados em arquivo XML, de nome USBase.xml (*UnkownStructureBase)* com a seguinte estrutura. Portanto nesse *report* são registrados conteúdos semelhantes aos encontrados no catálogo, mas sem a devida marcação padronizada conforme estruturas XML. O arquivo compõe uma base com estruturas desconhecidas ou fora do padrão do catálogo XML. As estruturas não conhecidas são registradas nas *tags* <index>, que contém um identificador numérico sequencial para a estrutura, e <name> com a estrutura encontrada fora de padrão.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<report tipo="unkown base" objetivo="assinaturas não conhecidas">
    <local>cel_B\modelo</local>
        <situação>NÃO ATENDE
            <artefato>modelo_cenario.lua
                    <UnknownStructureBase>
                            <UnknownStructure>
                                <index>0</index>
                                <name>cabeçalho</name>
                            </UnknownStructure>
                    </UnknownStructureBase>
            </artefato>
            <artefato>modelo_usuario.lua</artefato>
            <artefato>_teste_modelo.lua</artefato>
        </situação>
</report>
```

A não conformidade encontrada pelo SMA está relacionada ao registrado em *Padrão* no catálogo. Um padrão só é identificado no artefato se vier precedido e finalizado pelas assinaturas registradas no catálogo em *Variável Essencial* e *Sinônimos.*

Portanto, mesmo que haja a ocorrência dos textos registrados no catálogo em *Padrão*, mas que não vierem com os delimitadores, esses serão considerados em não conformidade (não atende).

Essas não conformidades também são registradas no arquivo USBase.xml. Além dos delimitadores, os padrões só são considerados em conformidade se estiverem completos conforme registrado em catálogo, caso falte algum elemento na comparação do artefato com o que esta estabelecido no catálogo, os elementos presentes no artefato são considerados como não conformidade.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<report tipo="unkown base" objetivo="assinaturas não conhecidas">
    <local>cel_B\modelo</local>
        <situação>NÃO ATENDE
            <artefato>modelo_cenario.lua
                    <UnknownStructureBase>
                            <UnknownStructure>
                               <index>1</index>
                               <name>@Titulo:</name>
                            </UnknownStructure>
                            <UnknownStructure>
                               <index>2</index>
                               <name>@Objetivo:</name>
                            </UnknownStructure>
                            <UnknownStructure>
                               <index>3</index>
                               <name>@Contexto:</name>
                            </UnknownStructure>
                            <UnknownStructure>
                               <index>4</index>
                               <name>@Atores:</name>
                            </UnknownStructure>
                            <UnknownStructure>
                               <index>5</index>
                               <name>@Recursos:</name>
                            </UnknownStructure>
                    </UnknownStructureBase>
            </artefato>
             <artefato>modelo_usuario.lua</artefato>
             <artefato>_teste_modelo.lua</artefato>
        </situação>
</report>
```

5.1.3
Considerações Finais

O estudo de caso estaria intimamente relacionado às funções de controle e auditoria em que, resumidamente, procuram verificar a corretude dos artefatos gerados, examinando dados, resultados de testes, completude em relação a requisitos, como também convergência às políticas de padronização da organização com relação aos processos, arquitetura e padrões empregados.

O *software* C&L possui 130 arquivos distribuídos em 20 pastas. Apesar de uma estrutura pequena, se estivesse inserida em um ambiente de desenvolvimento onde um número de desenvolvedores efetua manutenção em um mesmo código fonte, haveria a necessidade de se estabelecer processos que contemplem a Gestão de Configuração de *Software* (GCS) [ISO 12207]. A GCS procura agregar boas práticas de engenharia para projetos de *software*, para qualquer fase do desenvolvimento, prototipação, ou manutenção em andamento [IEEE Std-828 1998]. Ela procura aumentar a confiabilidade e qualidade do *software* através de:

- Providenciar estrutura para a identificação e controle da documentação, código, interfaces e bancos de dados para atender a todas as fases do ciclo de vida;

- Dar suporte ao método de desenvolvimento e manutenção escolhida para atender os requisitos, padrões, políticas, organização e filosofia gerencial;

- Produzir informações gerenciais e do produto com relação ao estado das linhas base, padronização, controle de mudanças, testes, liberações de software e auditorias.

Segundo a IEEE Std. 828 [IEEE Std-828 1998] e ISO 10007 [ISO 10007], a GCS é composta por quatro funções principais: (1) identificação, (2) controle, (3) acompanhamento e (4) auditoria.

A inserção do método em um processo de GCS poderia auxiliar funções de controle e auditoria em que especialistas utilizem o método para auxiliar suas atividades de chegagem de conformidade do artefato com padrões estabelecidos pela empresa. Executar o SMA em uma base de 130 arquivos poderá minizar a necessidade de interação humana para a análise de conformidades antes de se fazer o *check-in*[4] do artefato para o repositório de GCS.

[4] O termo *check-in* representa o processo de revisão, aprovação e cópia de ICs do espaço de trabalho do engenheiro de software para o repositório [Leon 2000].

O método como foi estabelecido para esse estudo de caso contempla a análise dos padrões, mas não faz uma comparação com os documentos da especificação de requisitos. Por exemplo, os padrões:

```
<padrões>
    <tagPadrao>@Titulo:% </tagPadrao>
    <tagPadrao>@Objetivo:% </tagPadrao>
    <tagPadrao>@Contexto:% </tagPadrao>
    <tagPadrao>@Localização:% </tagPadrao>
    <tagPadrao>@Pré-condição: %</tagPadrao>
    <tagPadrao>@Atores:% </tagPadrao>
    <tagPadrao>@Recursos:% </tagPadrao>
</padrões>
```

contemplam apenas as estruturas propostas na definição de cenários [Leite et al. 2000], mas não foram descritos seus conteúdos, ou seja, as especificações dos requisitos conforme apresentado para o cenário *Selecionar Projeto* do código utilizado no estudo da alternativa 1.1.1 Utilizar cabeçalhos de códigos fonte comentados a partir de critérios técnicos. Veja descrição a seguir:

> **Título:** SELECIONAR PROJETO.
> **Objetivo:** Permitir que o usuário trabalhe com um projeto específico, previamente cadastrado no sistema. O Sistema irá carregar os dados referentes ao projeto e exibirá no menu lateral todos os símbolos do léxico e cenários já cadastrados no projeto.
> **Contexto:** Usuário deve possuir um projeto cadastrado ou ser colaborador de um.
> **Recursos:** projetos criados pelo usuário e projetos em que o usuário é colaborador.
> **Atores:** usuário, sistema.
> **Episódios:**
> 1- O sistema exibe uma lista dos projetos acessíveis para o usuário, destacando o seu nível de acesso em cada um deles.
> 2- Usuário seleciona um dos projetos da lista.
> 3- Os cenários e símbolos do léxico previamente cadastrados são exibidos em um menu lateral.
> 4- As seguintes opções de tornam disponíveis para o usuário: cadastrar léxico, cadastrar cenário e gerar grafo do projeto.
> 5- Caso o usuário possua nível de acesso de administrador ou gerente as seguintes opções surgiriam, além das anteriores: GERAR XML DO PROJETO, gerenciar colaboradores do projeto e verificar alterações propostas.

Nesse estudo de caso utilizamos a análise da conformidade da estrutura e nos resultados (relatório do SMA) é indicado o nome do arquivo (código fonte) referente ao objeto analisado.

Para esse estudo de caso, haveria a necessidade de uma interação humana caso seja desejado verificar se o avaliado pelo SMA, ou seja, a conformidade entre artefato de *software* e catálogo, condizem com a os artefatos da especificação do *software*, conforme descrição do cenário anterior.

A análise foi realizada em uma estrutura de 20 pastas com 130 arquivos. O SMA permitiu uma análise da conformidade dos artefatos gerados com RNFs estabelecidos em catálogo e gera resultados (relatório do SMA) que indicam artefatos que possuem ou não possuem as padronizações exigidas para satisfazer aos RNFs analisados.

5.2
Aplicação do Método nos Traços de Execução de Servidor Apache

5.2.1
Definição do Servidor Apache [5]

O servidor Apache é um servidor Web livre com protocolo HTTP compatível com os principais sistemas operacionais como Windows, Linux, Unix e FreeBSD, criado na *National Center for Supercomputing Applications* (NCSA). Em pesquisas de uso demonstram que o servidor poussi cerca de 47% das plataformas instaladas em 2007 com crescimento para 55%, aproximadamente, em 2010.

Para garantir segurança nas transações HTTP, o servidor dispõe de um módulo chamado *mod_ssl*, o qual adiciona a capacidade do servidor atender requisições utilizando o protocolo HTTPS. Este protocolo utiliza uma camada SSL para criptografar todos os dados transferidos entre o cliente e o servidor, provendo maior grau de segurança, confidencialidade e confiabilidade dos dados. A camada SSL é compatível com certificados X.509, que são os certificados digitais fornecidos e assinados por grandes entidades certificadoras no mundo. O servidor é configurado por um arquivo mestre nomeado *httpd.conf* e opcionalmente pode haver configurações para cada diretório utilizando arquivos com o nome *.htaccess*, onde é possível utilizar autenticação de usuário pelo próprio protocolo HTTP utilizando uma combinação de arquivo *.htaccess* com um arquivo *.htpasswd*, que guardará os usuários e senhas (criptografadas).

5.2.2
Aplicação do Método para Traços de Execução de Servidor Apache

Encontrar alternativas para as questões do catálogo no ambiente onde o *software* é executado também pode auxiliar em satisfazer os *softgoals* estabelecidos. No caso de aplicações de ambientes Web, avaliar características do servidor, ou os traços de execução do *software* nesses ambientes pode potencializar as soluções esperadas para as questões do catálogo.

Nesse estudo de caso é feita uma análise do *softgoal* de Acessibilidade do CTS [Grupo ER PUC-Rio 2013] através de um de seus desmembramentos, no caso o *softgoal* Disponibilidade. A definição de Acessibilidade segundo o CTS é a *"Capacidade de ser identificada pela aferição de práticas que implementem*

[5] Definições extraídas de http://pt.wikipedia.org/wiki/Servidor_Apache

características de portabilidade, disponibilidade e publicidade". Um dos desmembramentos da Acessibilidade se dá pela qualidade de Disponibilidade, dessa forma, pretende-se analisar, não no *software*, mas em seu ambiente de execução se há indícios de que tal característica está satisfeita.

O servidor Apache propicia padrões de codificação para os traços de execução *software* que é executado nele. Dessa forma, mesmo que não sejam implementadas soluções de contingência na aplicação, o servidor permite algum tipo de controle para o que está sendo executado e consequentemente uma possibilidade de análise de conformidades com alguma regra estabelecida. O Apache divide seus códigos a partir de classes que é identificada a partir do primeiro dígito. Alguns recortes de códigos gerados pelo servidor dos traços de execução da aplicação web podem ser vistos como abaixo[6] as frases sublinhadas são referentes às classes:

1xx Informational - Request received, continuing process

101 Switching Protocols: This means the requester has asked the server to switch protocols and the server is acknowledging that it will do so.

102 Processing (WebDAV; RFC 2518): As a WebDAV request may contain many sub-requests involving file operations, it may take a long time to complete the request. This code indicates that the server has received and is processing the request, but no response is available yet. This prevents the client from timing out and assuming the request was lost.

2xx Success - This class of status codes indicates the action requested by the client was received, understood, accepted and processed successfully.

200 OK: Standard response for successful HTTP requests. The actual response will depend on the request method used. In a GET request, the response will contain an entity corresponding to the requested resource. In a POST request the response will contain an entity describing or containing the result of the action.

201 Created: The request has been fulfilled and resulted in a new resource being created.

202 Accepted: The request has been accepted for processing, but the processing has not been completed. The request might or might not eventually be acted upon, as it might be disallowed when processing actually takes place.

203 Non-Authoritative Information (since HTTP/1.1): The server successfully processed the request, but is returning information that may be from another source.

[6] http://en.wikipedia.org/wiki/List_of_HTTP_status_codes

3xx Redirection - The client must take additional action to complete the request.

300 Multiple Choices: Indicates multiple options for the resource that the client may follow. It, for instance, could be used to present different format options for video, list files with different extensions, or word sense disambiguation.

301 Moved Permanently: This and all future requests should be directed to the given URI.

303 See Other (since HTTP/1.1): The response to the request can be found under another URI using a GET method. When received in response to a POST (or PUT/DELETE), it should be assumed that the server has received the data and the redirect should be issued with a separate GET message.

4xx Client Error - The 4xx class of status code is intended for cases in which the client seems to have erred. Except when responding to a HEAD request, the server should include an entity containing an explanation of the error situation, and whether it is a temporary or permanent condition.

400 Bad Request: The request cannot be fulfilled due to bad syntax.

401 Unauthorized: Similar to 403 Forbidden, but specifically for use when authentication is required and has failed or has not yet been provided. The response must include a WWW-Authenticate header field containing a challenge applicable to the requested resource. See Basic access authentication and Digest access authentication.

404 Not Found: The requested resource could not be found but may be available again in the future.[2] Subsequent requests by the client are permissible.

408 Request Timeout: The server timed out waiting for the request. According to W3 HTTP specifications: "The client did not produce a request within the time that the server was prepared to wait". The client MAY repeat the request without modifications at any later time."

5xx Server Error - The server failed to fulfill an apparently valid request.

500 Internal Server Error: A generic error message, given when an unexpected condition was encountered and no more specific message is suitable.

501 Not Implemented: The server either does not recognize the request method, or it lacks the ability to fulfill the request. Usually this implies future availability (e.g., a new feature of a web-service API).

504 Gateway Timeout: The server was acting as a gateway or proxy and did not receive a timely response from the upstream server.

505 HTTP Version Not Supported: The server does not support the HTTP protocol version used in the request.

O Apache geralmente trabalha com tipos diferentes de arquivos para armazenamento de traços de execução: *apache error log, apache access log* e *agente log.* Particularmente para o estudo de caso serão considerados apenas traços de execução de acesso a uma aplicação web. Nos dois exemplos a seguir são tratados traços de sucesso e falha de acesso. Os caracteres em evidência indicam alguns dos códigos da tabela apresentada acima.

Apache access log (success - code 200):

192.168.2.20 - - [28/Nov/2013:10:27:10 -0300] "GET /cgi-bin/try/ HTTP/1.0" **200** 3395

127.0.0.1 - - [28/Nov/2013:10:22:04 -0300] "GET / HTTP/1.0" **200** 2216

Apache access log (failure - code 4xx):

127.0.0.1 - - [28/Nov/2013:10:27:32 -0300] "GET /hidden/ HTTP/1.0" **404** 7218

A1 - Criar SIG

As qualidades sugeridas no CTS para Acessibilidade são: portabilidade, disponibilidade e publicidade, conforme decomposição apresentada na Figura 30. Dadas tais qualidades como entrada na A1 o seguinte grafo seria apresentado:

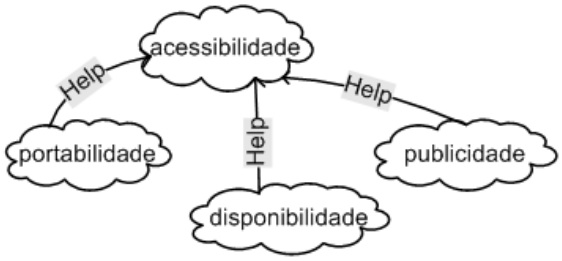


Figura 30: SIG Acessibilidade [CTS 2013].

O detalhamento de Acessibilidade propõe três qualidades que se relacionam com o *softgoal* principal através de um elo de ligação do tipo Help, o que indica que há uma contribuição positiva das qualidades. Para esse estudo de caso é feita uma análise do *softgoal* Disponibilidade.

A2 – Definir *patterns*

Para o *Softgoal* Disponibilidade

A definição de Disponibilidade no CTS corresponde à "*Capacidade de ser acessado quando for necessário*". A aplicação da A2 para o *softgoal* resulta

em um detalhamento de grupos, questões e alternativas de respostas para as questões.

O *softgoal* é decomposto em questões que se respondidas satisfazem à meta definida. Relacionadas às questões as mesmas são agrupadas em grupos que melhor a representam. No caso, Disponibilidade possui três grupos: 1) Identificação de meios de acesso; 2) Disponibilizar técnicas de segurança para disponibilidade ininterrupta da informação; e 3) Garantir acesso à informação em diferentes tecnologias e meios. Os grupos e as questões podem ser vistos conforme apresenta a Figura 31 no quadro Resultado. Tais detalhamentos são propostos no CTS [CTS 2013] e são definidos a partir da aplicação da **A2.1 Descrever questões** (descrever questões para *softgoals* folha, no caso Disponibilidade).

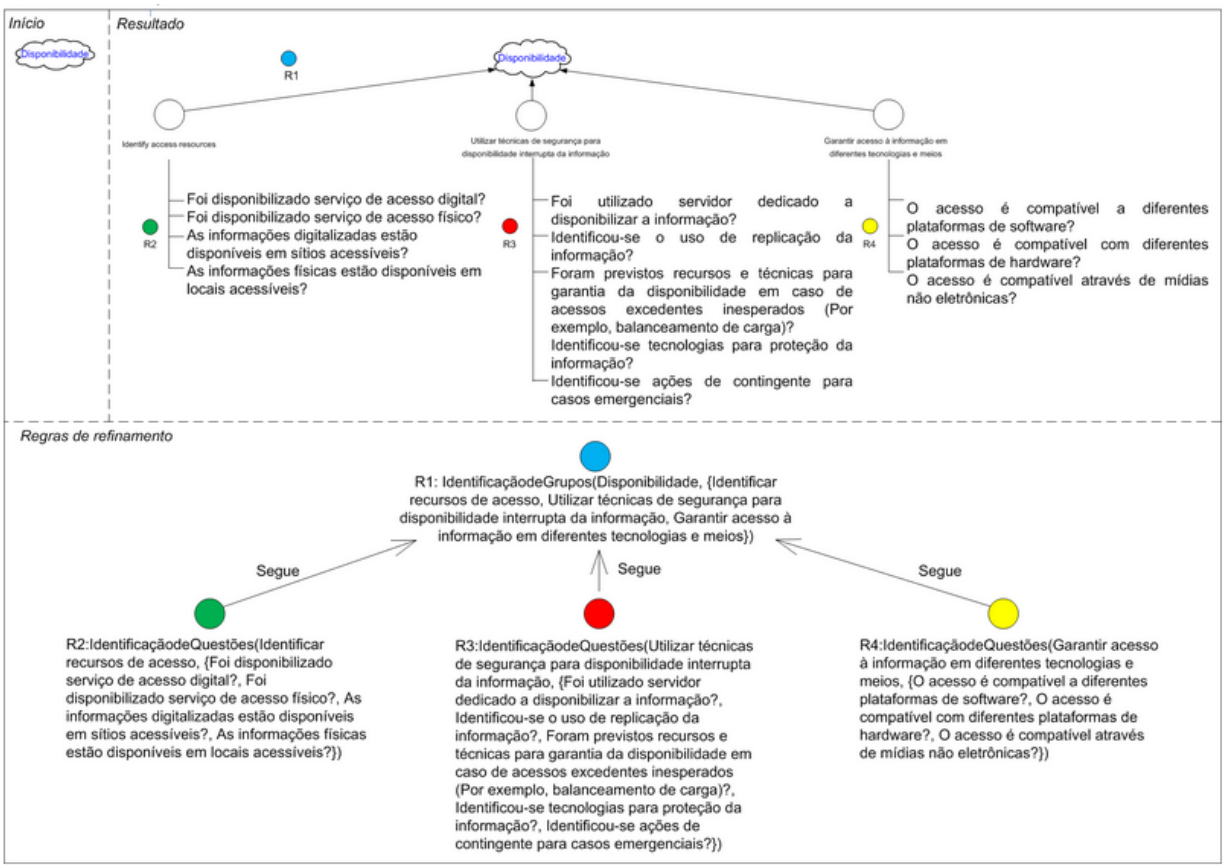


Figura 31: Patterns do *softgoal* Disponibilidade [CTS 2013].

A decomposição das questões a partir da definição das questões propõe a composição de respostas alternativas para as questões. Conforme descrito na **A2.2 Operacionalizar questões** da seção 3.4 Detalhamento de Atividades (Definir *patterns* A2) faz-se necessário o detalhamento das questões em alternativas ou operacionalizações que possam respondê-las.

As operacionalizações relatadas a seguir definidas a partir da aplicação da A2.2 foram elaboradas considerando alternativas para disponibilidade segundo o ambiente de execução de aplicação Web.

A questão e operacionalização escolhidas podem ser vistas na Tabela 8 que seguirá a numeração das tabelas descritas na seção anterior.

As operacionalizações foram retratadas de forma *ad-hoc* para efeito de estudo da aplicação do método, não necessariamente são estruturas que devem ser consideradas para reuso.

Tabela 8: Relação de grupos, questões e operacionalizações para o *softgoal* Disponibilidade.

Grupo:	**3 Utilizar medidas de segurança para disponibilidade da informação**
Questão 1:	3.1 Foram previstos recursos e técnicas para garantia da disponibilidade em caso de acessos excedentes inesperados (por exemplo, balanceamento de carga?)
Operacionalizações:	3.1.1 Analisar automaticamente os traços de execução do servidor Web para que sejam identificadas ocorrências de *timeout*.

A3 – Configurar XML

→ Configuração de grupos, questões e operacionalizações:

Definidas as estruturas anteriormente descritas, faz-se necessária sua transposição para uma arquitetura padronizada em XML definida como catálogo XML RNF (APÊNDICE A) que servirão como regras para os agentes do SMA. Sua transcrição é feita a partir das atividades propostas do detalhamento da A3 Configurar XML. Suas sub-atividades descritas como A3.1 Configurar *Objetive patterns* (estruturas XML), A3.2 Configurar *Questions patterns* (estruturas XML), A3.3 Configurar *Alternative patterns* (estruturas XML) e A3.4 Configurar Variáveis essenciais e seus sinônimos (estruturas XML) serão utilizadas para a composição da estrutura XML do catálogo de RNF que servirá como regra para as ações dos agentes do SMA.

O catálogo XML RNF para as estruturas definidas até o momento, após execução de A3.1, A3.2 e A3.3, segue conforme apresentado a seguir:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<catalogo tipo="RNF" objetivo="análise">
  <versao>1</versao>
      <patternObjetivos>
          <patternIdentificacao>Transparencia</patternIdentificacao>

          <patternDecomposicaoNivel_1>
                  <idDecomposicaoNivel_1>Acessibilidade</idDecomposicaoNivel_1>
                  <patternSelecao>HELP</patternSelecao>

                    ... (trecho suprimido propositalmente)

                    <patternDecomposicaoNivel_2>
                          <idDecomposicaoNivel_2>Disponibilidade</idDecomposicaoNivel_2>
                          <patternSelecao>HELP</patternSelecao>

                          <patternGrupo>
                                  <idGrupo>3</idGrupo>
                                  <tituloGrupo> Utilizar medidas de segurança para disponibilidade da informação
                                  </tituloGrupo>

                                  <patternQuestoes>
                                          <idQuestao>3.1</idQuestao>
                                          <tituloQuestao> Foram previstos recursos e técnicas para garantia da disponibilidade
                                          em caso de acessos excedentes inesperados (por exemplo, balanceamento de carga?)
                                          </tituloQuestao>

                                                  <patternAlternativas>
                                                      <idAlternativa>3.1.1</idAlternativa>
                                                      <tituloAlternativa>Analisar automaticamente os traços de execução do
                                                       servidor Web para que sejam identificadas ocorrências de timeout.
                                                      </tituloAlternativa>
                                                      <patternSelecao>RESPONDE</patternSelecao>
                                                      <variaveisEssenciais>
                                                          <idVariaveisEssenciais></idVariaveisEssenciais>
                                                          <nocaoVariaveisEssenciais> não se
                                                          aplica</nocaoVariaveisEssenciais>
                                                          <sinonimosVariaveisEssenciais>
                                                              <idSinonimosVariaveisEssenciais> não se aplica
                                                              </idSinonimosVariaveisEssenciais>
                                                          </sinonimosVariaveisEssenciais>
                                                      </variaveisEssenciais>

                                                      ... (trecho suprimido propositalmente)

                                                  </patternAlternativas>

                                  </patternQuestoes>
                          </patternGrupo>
                    </patternDecomposicaoNivel_2>

                    ... (trecho suprimido propositalmente)

          </patternDecomposicaoNivel_1>
      </patternObjetivos>
</catalogo>
```

→ Configuração de variáveis essenciais, sinônimos e padrões:

Após as execuções das atividades 3.1 a 3.3 é necessário executar A3.4 (Configurar Variáveis) para que se estabeleçam marcadores vinculados às operacionalizações para que esses possam ser identificadas no artefato.

A aplicação da atividade **A3.4 Configurar Variáveis essenciais e seus sinônimos** é feita inicialmente para o Grupo 1, Questão 1.1 e Operacinalização 1.1.1, bem como a atividade **A3.5 Configurar Padrões** deram origem às descritas conforme o resultado apresentado a seguir.

Tabela 9: Inserção de Variáveis Essenciais, Sinônimos e Padrões para o *softgoal* Disponibilidade.

Grupo:	**3 Utilizar medidas de segurança para disponibilidade da informação**
Questão 1:	3.1 Foram previstos recursos e técnicas para garantia da disponibilidade em caso de acessos excedentes inesperados (por exemplo, balanceamento de carga?)
Operacionalizações:	3.1.1 Analisar automaticamente os traços de execução do servidor Web para que sejam identificadas ocorrências de *timeout*.
Variável Essencial:	não se aplica
Noção:	não se aplica
Sinônimos:	não se aplica
Padrão:	% % % % % 408 %, % % % % % 504 %

Para o caso acima, não foram necessárias variáveis essenciais, consequentemente não há a descrição de noção nem sinônimos. Os padrões utilizados foram baseados na estrutura do padrão *Common Log Format*[7] do servidor Apache com códigos referentes à geração de *timeouts* pela aplicação. Os códigos 408 *Request timeout*, pelo lado cliente, e 504 *Gateway timeout*, pelo lado servidor, utilizados referem-se respectivamente a: "*The server timed out waiting for the request. According to W3 HTTP specifications: "The client did not produce a request within the time that the server was prepared to wait. The client MAY repeat the request without modifications at any later time."* e *"The server was acting as a gateway or proxy and did not receive a timely response from the upstream server".* Os códigos utilizados foram extraídos da Lista de HTTP *status codes*[8]. Os símbolos de % utilizados indicam outros parâmetros gerados pelo servidor e podem conter textos quaisquer. A vírgula entre as duas estruturas do padrão indicam que o SMA utilizará a lista dos dois padrões para a pesquisa.

A estrutura a estruturação em XML segue como apresentado abaixo:

```
... (trecho suprimido propositalmente)
<patternDecomposicaoNivel_2>
        <idDecomposicaoNivel_2> Disponibilidade </idDecomposicaoNivel_2>
```

[7] http://httpd.apache.org/docs/2.2/logs.html e http://httpd.apache.org/docs/2.2/mod/mod_log_config.html#customlog

[8] http://en.wikipedia.org/wiki/List_of_HTTP_status_codes

```
<patternSelecao>HELP</patternSelecao>

<patternGrupo>
        <idGrupo>3</idGrupo>
        <tituloGrupo> Utilizar medidas de segurança para disponibilidade da informação </tituloGrupo>

        <patternQuestoes>
                <idQuestao>3.1</idQuestao>
                <tituloQuestao Foram previstos recursos e técnicas para garantia da disponibilidade
                        em caso de acessos excedentes inesperados (por exemplo, balanceamento de carga?)
                </tituloQuestao>

                        <patternAlternativas>
                                <idAlternativa>3.1.1</idAlternativa>
                                <tituloAlternativa> Analisar automaticamente os traços de execução do
                                 servidor Web para que sejam identificadas ocorrências de timeout.
                                 </tituloAlternativa>
                                <patternSelecao>RESPONDE</patternSelecao>
                                <variaveisEssenciais>
                                        <idVariaveisEssenciais></idVariaveisEssenciais>
                                        <nocaoVariaveisEssenciais> </nocaoVariaveisEssenciais>
                                        <sinonimosVariaveisEssenciais>
                                                <idSinonimosVariaveisEssenciais> </idSinonimosVariaveisEssenciais>
                                        </sinonimosVariaveisEssenciais>

                                        <padrao>
                                                <idPadrao1> % % % % % 408 %,  % % % % % 504 %</ idPadrao1>
                                        </padrao>
                                </variaveisEssenciais>
... (trecho suprimido propositalmente)
```

A4 – Configurar *Software*

Nesse ponto há a necessidade da configuração do *software* para a geração automática dos traços de execução. Como estamos tratando de ambiente de execução a configuração é inerente a esse ambiente. Portanto, é necessário que o Apache esteja configurado para gerar os traços de execução em um arquivo e nesse caso a partir do seu tipo padrão *Common Log Format.* A localização do arquivo com os traços de execução do C&L deve ser configurada no próprio SMA, ou seja, o aquivo gerado deve estar presente na pasta *path* para que o agente MONITOR possa captura-lo e dar início ao processo de análise.

Ao tratar a configuração e a execução do SMA, caso os agentes encontrem a ocorrência do padrão estabelecido, entende-se que a propagação até o *softgoal* seja positiva, ou seja, que houve conformidade com o estabelecido no catálogo. Isso se dá nesse caso, porque, mesmo a aplicação entrando em *timeout* foi estabelecido um padrão de avaliação para tal ocorrência de acordo com as regras definidas, isso indica que a aplicação proveu um recurso para responder à questão.

Os traços de execução utilizados foram da aplicação C&L da versão PHP, uma vez que tal aplicação foi desenvolvida com suporte de execução sob

o servidor Apache. O exemplo a seguir é um traço de execução do C&L no servidor Apache que sinalize o sucesso da resposta do servidor diante da requisição de acesso à página web.

> 127.0.0.1 comp1 [10/Oct/2000:13:55:36 -0700] "GET / pes.inf.puc-rio.br/cel/index.htm HTTP/1.0" 200 2326

A5 – Operacionalizar Agentes

A execução dos agentes se dá pelas atividades A5.1, A5.2 e A5.3, basicamente as atividades representam o funcionamento desde a captura do artefato até sua análise a partir da comparação de marcações ou conteúdos com as regras definidas no catálogo.

O resultado final para o estudo de caso é apresentado em arquivo XML como apresentado a seguir:

Alternativa: 3.1.1 Usar cabeçalhos de códigos fonte comentados a partir de critérios técnicos

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<report tipo="RNF" objetivo="ANÁLISE">
    <artefato>httpd-access.log </artefato>
    <local>usr/local/apache/logs</local>
     <patternIdentificacao>Transparencia</patternIdentificacao>
        <patternDecomposicaoNivel_1>
            <idDecomposicaoNivel_1>Acessibilidade</idDecomposicaoNivel_1>
                <patternDecomposicaoNivel_2>
                    <idDecomposicaoNivel_2>Disponibilidade</idDecomposicaoNivel_2>
                       <idGrupo>3</idGrupo>
                       <tituloGrupo> Utilizar medidas de segurança para disponibilidade da informação</tituloGrupo>
                       <patternQuestoes>
                            <idQuestao>3.1</idQuestao>
                             <tituloQuestao> Foram previstos recursos e técnicas para garantia da disponibilidade
                                    em caso de acessos excedentes inesperados (por exemplo, balanceamento de
                                    carga?</tituloQuestao>
                            <patternAlternativas>
                                  <idAlternativa>3.1.1</idAlternativa>
                                  <tituloAlternativa>Analisar automaticamente os traços de execução do
                                          servidor Web para que sejam identificadas ocorrências de
                                          timeout.</tituloAlternativa>
                                 <situação> EM CONFORMIDADE </situação>
                                 <variáveis>
                                     <tagsVariável> </tagsVariável>
                                 </variáveis>
                                 <padrões>
                                      <tagPadrao>% % % % % 408 %</tagPadrao>
                                  </padrões>
                            </patternAlternativas>
                <patternDecomposicaoNivel_2>
            </patternDecomposicaoNivel_1>
</report>
```

Para o caso apresentado, há uma necessidade de melhoria na regulagem de filtro e apresentação com relação à não conformidade, pois, o SMA está considerando todo o rastro em desacordo com *% % % % % 408 %* e *% % % % % 504 %*, registrado em *Padrão*, considerando que % identifica qualquer conteúdo, como uma não conformidade, portanto, o SMA gera um arquivo de todo o rastro registrado pelo Apache.

5.2.3
Considerações Finais

Nesse estudo de caso foi realizada uma análise do *software* C&L em execução. A análise possibilitou verificar características de RNF não no *software* ou em seus artefatos, mas no ambiente onde ele opera. No caso os testes realizados para a configuração proposta foram estabelecidos nos traços de execução do *software* no servidor Apache para que a análise do RNF de Disponibilidade fosse avaliada. Diante do estudo de casos pode-se perceber que a questão e alternativa postas foram dadas em conformidade com o catálogo pré-estabelecido uma vez que o padrão pesquisado nos traços de execução do *software* estavam condizentes com a regra estabelecida.

Apesar da pesquisa ter sido feita para uma falha do *software* por gerar um *timeout*, o procedimento de prever recursos e técnicas para garantia da disponibilidade, indicado na questão 4.1, indica que o *software* estava configurado para perceber tal falha e permitir ocorrências de *timeout* fossem identificadas, conforme descrito na alternativa 4.1.1. Para esse estudo de caso foi simulado a falha do tipo *408 Request Timeout: The server timed out waiting for the request. According to W3 HTTP specifications: "The client did not produce a request within the time that the server was prepared to wait". The client MAY repeat the request without modifications at any later time."* gerada pelo servidor Apache.

5.3
Aplicação do Método Sobre Traços de Execução do Lattesscholar

5.3.1
Definição do LS

O LS [Lattesscholar 2013] é um SMA para contagem de citações científicas, que combina os serviços do sítio do *Lattes* [Lattes 2013] e do *Google Scholar* [Scholar 2013]. Sua execução é baseada na pesquisa inicial pelo nome do autor no sítio do *Lattes* e posteriormente uma pesquisa pela relação de artigos no *Google Scholar* com o objetivo de contabilizar citações sobre cada publicação do autor. O resumo do funcionamento do LS pode ser visto na Figura 32.

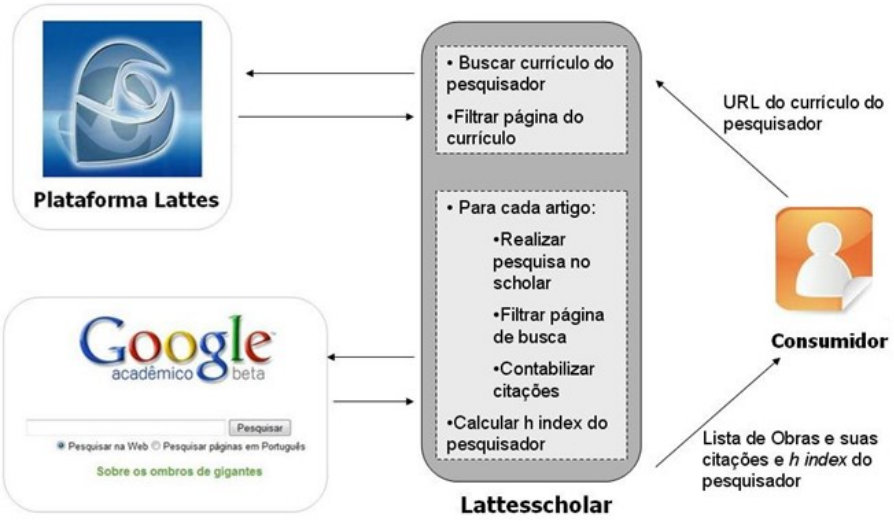


Figura 32: Funcionamento do LS [9].

O LS foi desenvolvido até o momento em três versões[10]. A primeira versão é a implementação de um algoritmo para ordenação, pelo número de citações, dos artigos publicados no *Workshop* de Engenharia de Requisitos (WER). Esse algoritmo foi implementado em páginas *web* na linguagem PHP para o WER Papers, uma biblioteca digital da conferência WER, e utiliza o serviço do *Google Scholar*. Em sua segunda versão *web* implementada na linguagem Lua[11] e utiliza serviço do *Lattes* como um repositório de currículos de pesquisadores.

A terceira versão do LS é um SMA implementado com base em intencionalidade a partir do *framework* i* implementado em Jade com extensão do uso de GQO. O estudo de caso é aplicado sob essa terceira versão do LS.

[9] http://transparencia.inf.puc-rio.br/wiki/index.php/Lattesscholar
[10] http://www2.dbd.puc-rio.br/pergamum/tesesabertas/0711310_2011_cap_8.pdf
[11] http://pt.wikipedia.org/wiki/Lua_(linguagem_de_programa%C3%A7%C3%A3o)

5.3.2
Aplicação do Método para o LS

A aplicação do método no *software* LS foi feita a partir da análise do RNF de Auditabilidade, definido e presente no CTS (2013) e formalizado em [Cappelli 2009]. O CTS traz a definição de Auditabilidade como "*Capacidade de ser identificada pela aferição de práticas que implementem características de validade, controlabilidade, verificabilidade, rastreabilidade e explicação*" [CTS 2013].

A aplicação do método pretende avaliar se há no *software*, objeto da análise, características que satisfazem o requisito de Auditabilidade, em particular tratando do *softgoal* de Rastreabilidade.

As atividades do método foram aplicadas conforme abaixo:

A1 - Criar SIG

Uma série de qualidades é entrada para a A1, nesse caso, Auditabilidade, Validade, Controlabilidade, Verificabilidade, Rastreabilidade e Explicação. Ao final das atividades essas características passam a ser *softgoals* que necessitam ser satisfeitos para atender à Auditabilidade. As relações de Help indicam como são os elos de contribuição entre os *softgoals* folha em relação à Auditabilidade. O resultado final da A1, além da relação dos *softgoals*, é o SIG apresentado na Figura 33.

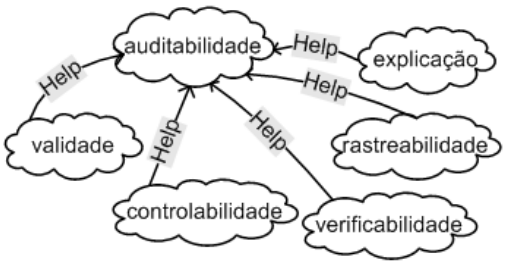


Figura 33: SIG Auditabilidade [Cappelli 2009].

O detalhamento de Auditabilidade propõe qualidades que se relacionam com o *softgoal* principal através de um elo de ligação do tipo Help, o que indica que há uma contribuição positiva das qualidades. Para esse estudo de caso é feita uma análise do *softgoal* Rastreabilidade e consequentemente a potencialização de Auditabilidade.

A2 – Definir *patterns*

Para o *Softgoal* Rastreabilidade

A definição de Disponibilidade no CTS corresponde à "*Capacidade de seguir a construção ou evolução de uma informação ou de um processo, suas mudanças e justificativas.*". A aplicação da A2 para o *softgoal* resulta em um detalhamento de grupos, questões e alternativas de respostas para as questões.

O *softgoal* é decomposto em questões que se respondidas satisfazem à meta definida. Relacionadas às questões as mesmas são agrupadas em grupos que melhor a representam. No caso, Rastreabilidade possui três grupos: 1) Fazer pré-rastreabilidade; 2) Fazer rastreabilidade em tempo de desenho; e 3) Fazer rastreabilidade em tempo de. Os grupos e as questões podem ser vistos conforme apresenta a Figura 34 no quadro Resultado. Tais detalhamentos são propostos no CTS [CTS 2013] e são definidos a partir da aplicação da **A2.1 Descrever questões** (descrever questões para *softgoals* folha, no caso Rastreabilidade).

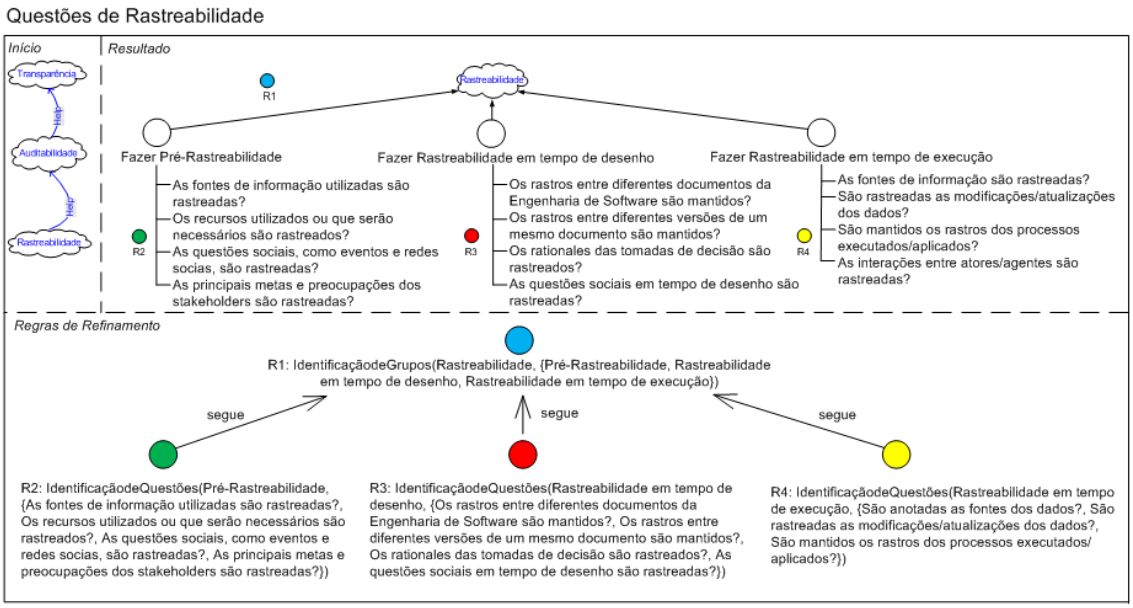


Figura 34: Patterns do *softgoal* Rastreabilidade [CTS 2013].

Como é descrito na atividade **A2.2 Operacionalizar questões**, a decomposição das questões devem ser feitas com alternativas ou operacionalizações que possam respondê-las.

As questões e operacionalizações escolhidas podem ser vistas na Tabela 10 que seguirá a numeração das tabelas descritas na seção anterior. As operacionalizações foram retratadas de forma *ad-hoc* para efeito de estudo da aplicação do método, não necessariamente são estruturas que devem ser consideradas para reuso.

Tabela 10: Relação de grupos, questões e operacionalizações para o *softgoal* Rastreabilidade.

Grupo:	**4 Fazer Rastreabilidade em Tempo de Execução.**
Questão 1:	4.1 As fontes de informação utilizadas são rastreadas?
Operacionalizações:	4.1.1 São identificados atores que passam a informação
	4.1.2 É identificado o momento em que a informação foi coleta
Questão 2:	4.2 As interações entre atores/agentes são rastreadas?
Operacionalizações:	4.2.1 Registrar troca de mensagens entre os atores/agentes interessados.

A3 – Configurar XML

Para esse estudo de caso serão apresentadas as etapas de A3 e ao final a estrutura do XML.

→ Configuração de grupos, questões e operacionalizações:

Definidas as estruturas anteriormente descritas, faz-se necessária sua transposição para uma arquitetura padronizada em XML definida como catálogo XML RNF (APÊNDICE A) que servem como regras para os agentes do SMA. Sua transcrição é feita a partir das atividades propostas do detalhamento da A3 Configurar XML. Suas sub-atividades descritas como A3.1 Configurar *Objetive patterns* (estruturas XML), A3.2 Configurar *Questions patterns* (estruturas XML), A3.3 Configurar *Alternative patterns* (estruturas XML) e A3.4 Configurar Variáveis essenciais e seus sinônimos (estruturas XML) são utilizadas para a composição da estrutura XML do catálogo de RNF que servirá como regra para as ações dos agentes do SMA. O XML será apresentado ao final.

→ Configuração de variáveis essenciais, sinônimos e padrões:

Após as execuções das atividades 3.1 a 3.3 é necessário executar A3.4 Configurar Variáveis para que se estabeleçam marcadores vinculados às operacionalizações para que as mesmas possam ser identificadas no artefato.

A aplicação da atividade **A3.4 Configurar Variáveis essenciais e seus sinônimos** é feita inicialmente para o Grupo 1, Questão 1.1 e Operacinalização 1.1.1, bem como a atividade **A3.5 Configurar Padrões** deram origem às descritas conforme o resultado apresentado a seguir.

Tabela 11: Inserção de Variáveis Essenciais, Sinônimos e Padrões para o *softgoal* Rastreabilidade.

Grupo:	**4 Fazer Rastreabilidade em Tempo de Execução.**
Questão 1:	4.1 As fontes de informação utilizadas são rastreadas?
Operacionalização:	4.1.1 São identificados atores que passam a informação.
Variável Essencial:	agente
Noção:	Agente autônomo que interage por troca de mensagens com outros agentes com objetivo de realizar alguma ação.
Sinônimos:	ator, nome do agente, nome do ator, agent name
Padrão:	não se aplica
Operacionalização:	4.1.2 É identificado o momento em que a informação foi coleta
Variável Essencial:	data de gravação
Noção:	Data em que a mensagem é gravada, registrada ou coletada.
Sinônimos:	data da coleta, data de registro
Padrão:	não se aplica
Questão 2:	4.2 As interações entre atores/agentes são rastreadas?
Operacionalizações:	4.2.1 Registrar troca de mensagens entre os atores/agentes interessados.
Variável Essencial:	conteúdo
Noção:	Conteúdo da mensagem trocada entre agentes.
Sinônimos:	content
Padrão:	não se aplica

O catálogo XML RNF para as estruturas definidas até o momento para o estudo de caso, após execução de A3.1, A3.2 e A3.3, segue conforme apresentado a seguir:

```
... (trecho suprimido propositalmente)
<patternDecomposicaoNivel_2>
        <idDecomposicaoNivel_2>Rastreabilidade</idDecomposicaoNivel_2>
        <patternSelecao>HELP</patternSelecao>

        <patternGrupo>
                <idGrupo>4</idGrupo>
                <tituloGrupo> Fazer Rastreabilidade em Tempo de Execução</tituloGrupo>

                <patternQuestoes>
                        <idQuestao>4.1</idQuestao>
                        <tituloQuestao> As fontes de informação utilizadas são rastreadas?</tituloQuestao>

                                <patternAlternativas>
                                        <idAlternativa>4.1.1</idAlternativa>
                                        <tituloAlternativa> São identificados atores que passam a informação
                                         </tituloAlternativa>
                                        <patternSelecao>RESPONDE</patternSelecao>
                                        <variaveisEssenciais>
```

```
                    <idVariaveisEssenciais>agente</idVariaveisEssenciais>
                    <nocaoVariaveisEssenciais> Agente autônomo que interage por troca de
                        mensagens com outros agentes com objetivo de realizar alguma ação.
                    </nocaoVariaveisEssenciais>
                    <sinonimosVariaveisEssenciais>
                        <idSinonimosVariaveisEssenciais>ator</idSinonimosVariaveisEssenciais>
                        <idSinonimosVariaveisEssenciais>nome do agente</idSinonimosVariaveisEssenciais>
                        <idSinonimosVariaveisEssenciais>nome do ator</idSinonimosVariaveisEssenciais>
                        <idSinonimosVariaveisEssenciais>agent name</idSinonimosVariaveisEssenciais>
                    </sinonimosVariaveisEssenciais>

                    <padrao>
                        <idPadrao1>não se aplica</ idPadrao1>
                    </padrao>
                </variaveisEssenciais>

                <idAlternativa>4.1.2</idAlternativa>
                <tituloAlternativa>É identificado o momento em que a informação foi coleta
                 </tituloAlternativa>
                <patternSelecao>RESPONDE</patternSelecao>
                <variaveisEssenciais>
                    <idVariaveisEssenciais> data de gravação </idVariaveisEssenciais>
                    <nocaoVariaveisEssenciais> Data em que a mensagem é gravada, registrada ou coletada.
                    </nocaoVariaveisEssenciais>
                    <sinonimosVariaveisEssenciais>
                        <idSinonimosVariaveisEssenciais>data da coleta</idSinonimosVariaveisEssenciais>
                        <idSinonimosVariaveisEssenciais>data de registro</idSinonimosVariaveisEssenciais>
                    </sinonimosVariaveisEssenciais>

                    <padrao>
                        <idPadrao1>não se aplica</ idPadrao1>
                    </padrao>
                </variaveisEssenciais>

            <idQuestao>4.2</idQuestao>
            <tituloQuestao As interações entre atores/agentes são rastreadas?</tituloQuestao>

                    <idAlternativa>4.2.1</idAlternativa>
                    <tituloAlternativa> Registrar troca de mensagens entre os atores/agentes interessados.
                     </tituloAlternativa>
                    <patternSelecao>RESPONDE</patternSelecao>
                    <variaveisEssenciais>
                        <idVariaveisEssenciais>conteúdo</idVariaveisEssenciais>
                        <nocaoVariaveisEssenciais> Conteúdo da mensagem trocada entre agentes.
                        </nocaoVariaveisEssenciais>
                        <sinonimosVariaveisEssenciais>
                            <idSinonimosVariaveisEssenciais>content</idSinonimosVariaveisEssenciais>
                        </sinonimosVariaveisEssenciais>

                        <padrao>
                            <idPadrao1>não se aplica</ idPadrao1>
                        </padrao>
                    </variaveisEssenciais>
```

... (trecho suprimido propositalmente)

A4 – Configurar *Software*

A partir dos padrões estabelecidos em catálogo faz-se necessária a configuração do software para que possa gerar estruturas de traços de execução compatíveis com marcadores passíveis de interpretação pelo SMA. Os traços de execução gerados pelo próprio sistema LS para as ações dos agentes

Por exemplo, abaixo é apresentado uma primeira seção gerada pelo Agente1 do LS a partir de sua execução. Basicamente estão nessa seção as condutas ou ações, baseadas no modelo elaborado para o LS em i*, que estão sendo executadas a cada momento pelo agente.

```
20130913 15:43:24: state=2
20130913 15:43:24: service type=istar.impl.Resource:Citacoes
name=scholar:istar.impl.Resource:Citacoes
20130913 15:43:24: service(s) registered
20130913 15:43:25: behaviour
type=istar.behaviour.MeansEndUniqueBehaviour name=UrlsBusca adding
subBehaviour
20130913 15:43:25: behaviour
type=basicelement.MontarUrlBusca$myBehaviour
name=ObterObrasTopicos
20130913 15:43:25: behaviour
type=istar.behaviour.SequentialTaskBehaviour name=ObterCitacoes adding
subBehaviour
20130913 15:43:25: behaviour
type=istar.behaviour.MeansEndUniqueBehaviour name=UrlsBusca
20130913 15:43:25: behaviour
type=istar.behaviour.MeansEndUniqueBehaviour name=CitacoesPorObra
adding subBehaviour
20130913 15:43:25: behaviour
type=basicelement.ObterCitacoesPorObra$myBehaviour
name=ObterCitacoesPorObra
20130913 15:43:25: behaviour
type=istar.behaviour.SequentialTaskBehaviour name=ObterCitacoes adding
subBehaviour
20130913 15:43:25: behaviour
type=istar.behaviour.MeansEndUniqueBehaviour name=CitacoesPorObra
20130913 15:43:25: behaviour
type=istar.behaviour.MeansEndUniqueBehaviour
name=SolitacoesCitacoesAtendidas adding subBehaviour
20130913 15:43:25: behaviour
type=istar.behaviour.SequentialTaskBehaviour name=ObterCitacoes
20130913 15:43:25: behaviour type=class
maingoal.SolitacoesCitacoesAtendidas$WaitingRequestMessage
name=Initial state=READY
```

Quando LS termina a geração dos traços de exeução descritos acima, ele inicia a geração de traços de execução do agente quando este pesquisa por obras, publicações científicas e o número de citações. O exemplo pode ser visto a seguir:

20130913 15:45:24: Agent name=agente1@VIVALDI:1099/JADE: executing. Ticket=agente1@VIVALDI:1099/JADE_1
20130913 15:45:24: Agent name=agente1@VIVALDI:1099/JADE received a REQUEST for obra: Lexicon Based Ontology Construction.
20130913 15:45:24: Agent name=agente1@VIVALDI:1099/JADE received a REQUEST for obra: Ontology as a Requirements Engineering Product.
20130913 15:45:24: Agent name=agente1@VIVALDI:1099/JADE received a REQUEST for obra: Catologuing Non-Functional Requirements as Softgoal Network.
20130913 15:45:24: Agent name=agente1@VIVALDI:1099/JADE received a REQUEST for obra: Experiences Using Scenarios to Enhance Traceability.
20130913 15:45:24: Agent name=agente1@VIVALDI:1099/JADE received a REQUEST for obra: Um Mecanismo de Rastreamento da Evolução de Cenários Baseado em Transformações.
20130913 15:45:24: Agent name=agente1@VIVALDI:1099/JADE received a REQUEST for obra: Enriquecendo o Código com Cenários.
20130913 15:45:24: Agent name=agente1@VIVALDI:1099/JADE received a REQUEST for obra: Geração de ontologias subsidiada pela Engenharia de Requisitos.
20130913 15:45:24: Agent name=agente1@VIVALDI:1099/JADE received a REQUEST for obra: Indicadores para a Gerencia de Requisitos.
20130913 15:45:24: Agent name=agente1@VIVALDI:1099/JADE received a REQUEST for obra: Domain Networks in the Software Development Process.

Para que os traços de exeução do LS pudessem ser utilizados no estudo de caso dessa tese foi necessário uma transformação dos arquivos em ASCii em formato XML. Esse procedimento foi feito com base nos registros gerados na segunda seção do Agente1, conforme descrito no exemplo imediatamente acima.

Esse procedimento não faz parte do método e está descrito aqui para que seja possível o entendimento de como foram trabalhados os arquivos do LS para que pudessem ser lidos pelo SMA do método proposto nessa tese.

O arquivo com os traços de execução do Agente1 do LS foi mapeado para o formato XML com *tags* compatíveis com o catálogo para indicar pontos em que os desenvolvedores do LS tenham inserido características do RNF desejado. Um exemplo de traço de execução para o LS pode ser visto a seguir:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<lattesScholar>

  <mensagem>
     <data do registro>20130913 15:45:24</data do registro >
     <agente> agente1</agente>
     <origem>@VIVALDI:1099/JADE</origem>
     <receiver>server</receiver>
```

```
        <destino>@VIVALDI:1099/JADE</destino>
        <tipo>REQUEST for obra</tipo>
        <conteúdo>Ontology as a Requirements Engineering Product</conteúdo>
        <status>received</status>
      </mensagem >

      <menssagem>
        <data do registro>20130913 15:45:24</data do registro>
        <agente> agente1</agente>
        <origem>@VIVALDI:1099/JADE</origem>
        <receiver>server</receiver>
        <destino>@VIVALDI:1099/JADE</destino>
        <tipo>REQUEST for obra</tipo>
        <conteúdo> Catologuing Non-Functional Requirements as Softgoal
                   Network.</conteúdo>
        <status>received</status>
      </messagem>

      <menssagem>
        <data de registro>20130913 15:45:24</data de registro>
        <agente> agente1</agente>
        <origem>@VIVALDI:1099/JADE</origem>
        <receiver>server</receiver>
        <destino>@VIVALDI:1099/JADE</destino>
        <tipo>REQUEST for obra</tipo>
        <conteúdo> Experiences Using Scenarios to Enhance Traceability.</conteúdo>
        <status>received</status>
      </messagem>
    </lattesScholar>
```

A5 – Operacionalizar Agentes

As atividades A5.1, A5.2 e A5.3 do método proposto representam o funcionamento desde a captura do artefato até sua análise a partir da comparação de marcações ou conteúdos com as regras definidas no catálogo.

O resultado final para o estudo de caso é apresentado em arquivo XML como apresentado a seguir. As conformidades foram obtidas a partir da comparação do artefato em formato XML com os traços de execução do LS com as variáveis essenciais configuradas no catálogo de RNF, conforme descrito na atividade A3.

Alternativa: 4.1.1 São identificados atores que passam a informação

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<report tipo="RNF" objetivo="ANÁLISE">
    <artefato>agente1-log-2013-09-13 15-45.log </artefato>
    <local>svn/trunk/LattesScholar/file/logs</local>
     <patternIdentificacao>Transparencia</patternIdentificacao>
        <patternDecomposicaoNivel_1>
            <idDecomposicaoNivel_1>Auditabilidade</idDecomposicaoNivel_1>
                <patternDecomposicaoNivel_2>
                    <idDecomposicaoNivel_2>Rastreabilidade</idDecomposicaoNivel_2>
                     <idGrupo>4</idGupo>
```

```
                    <tituloGrupo> Fazer Rastreabilidade em Tempo de Execução</tituloGrupo>
                    <patternQuestoes>
                        <idQuestao>4.1</idQuestao>
                        <tituloQuestao> As fontes de informação utilizadas são rastreadas?</tituloQuestao>
                        <patternAlternativas>
                            <idAlternativa>4.1.1</idAlternativa>
                            <tituloAlternativa> Agente autônomo que interage por troca de
                                mensagens com outros agentes com objetivo de realizar alguma ação.
                            </tituloAlternativa>
                            <situação> EM CONFORMIDADE </situação>
                            <variáveis>
                                <tagsVariável>agente </tagsVariável>
                            </variáveis>
                            <padrões>
                                <tagPadrao></tagPadrao>
                            </padrões>
                        </patternAlternativas>
                    <patternDecomposicaoNivel_2>
                </patternDecomposicaoNivel_1>
</report>
```

Alternativa: 4.1.2 É identificado o momento em que a informação foi coleta

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<report tipo="RNF" objetivo="ANÁLISE">
    <artefato> agente1-log-2013-09-13 15-45.log</artefato>
    <local> svn/trunk/LattesScholar/file/logs </local>
     <patternIdentificacao>Transparencia</patternIdentificacao>
        <patternDecomposicaoNivel_1>
            <idDecomposicaoNivel_1>Auditabilidade</idDecomposicaoNivel_1>
                <patternDecomposicaoNivel_2>
                    <idDecomposicaoNivel_2>Rastreabilidade</idDecomposicaoNivel_2>
                    <idGrupo>4</idGrupo>
                    <tituloGrupo> Fazer Rastreabilidade  em Tempo de Execução</tituloGrupo>
                    <patternQuestoes>
                        <idQuestao>4.1</idQuestao>
                        <tituloQuestao> As fontes de informação utilizadas são rastreadas?</tituloQuestao>
                        <patternAlternativas>
                            <idAlternativa>4.1.2</idAlternativa>
                            <tituloAlternativa> Data em que a mensagem é gravada, registrada ou coletada.
                                </tituloAlternativa>
                            <situação> EM CONFORMIDADE </situação>
                            <variáveis>
                                <tagsVariável>data de registro </tagsVariável>
                            </variáveis>
                            <padrões>
                                <tagPadrao> conteúdo</tagPadrao>
                            </padrões>
                        </patternAlternativas>
                    <patternDecomposicaoNivel_2>
                </patternDecomposicaoNivel_1>
</report>
```

Alternativa: 4.2.1 Registrar troca de mensagens entre os atores/agentes interessados

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<report tipo="RNF" objetivo="ANÁLISE">
    <artefato> agente1-log-2013-09-13 15-45.log</artefato>
    <local> svn/trunk/LattesScholar/file/logs</local>
     <patternIdentificacao>Transparencia</patternIdentificacao>
        <patternDecomposicaoNivel_1>
            <idDecomposicaoNivel_1>Auditabilidade</idDecomposicaoNivel_1>
                <patternDecomposicaoNivel_2>
                    <idDecomposicaoNivel_2>Rastreabilidade</idDecomposicaoNivel_2>
```

```
                    <idGrupo>4</idGrupo>
                    <tituloGrupo> Fazer Rastreabilidade em Tempo de Execução </tituloGrupo>
                    <patternQuestoes>
                        <idQuestao>4.2</idQuestao>
                        <tituloQuestao> As interações entre atores/agentes são rastreadas?</tituloQuestao>
                        <patternAlternativas>
                            <idAlternativa>4.2.1</idAlternativa>
                            <tituloAlternativa> Registrar troca de mensagens entre os atores/agentes interessados.
                                </tituloAlternativa>
                            <situação> EM CONFORMIDADE </situação>
                            <variáveis>
                                <tagsVariável>conteúdo </tagsVariável>
                            </variáveis>
                            <padrões>
                                <tagPadrao></tagPadrao>
                            </padrões>
                        </patternAlternativas>
                    <patternDecomposicaoNivel_2>
                </patternDecomposicaoNivel_1>
</report>
```

Um exemplo do tipo de análise do SMA para o caso estudado está relacionada às assinaturas serão feitas por assinaturas não conhecidas em artefatos. Para esse estudo de caso foram pesquisadas tags delimitadas por "<" e ">" com conteúdos não registrados no catálogo. Os registros foram feitos no arquivo USBase.xml (*UnkownStructureBase)* com a seguinte estrutura.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<report tipo="unkown base" objetivo="assinaturas não conhecidas">
    <local>svn/trunk/LattesScholar/file/logs</local>
        <situação>NÃO ATENDE
            <artefato> agente2-log-2013-09-13 15-52.log
                <UnknownStructureBase>
                    <UnknownStructure>
                        <index>0</index>
                        <name>receiver</name>
                        <index>1/index>
                        <name>origem</name>
                        <index>2/index>
                        <name>destino</name>
                        <index>3/index>
                        <name>tipo</name>
                    </UnknownStructure>
                </UnknownStructureBase>
            </artefato>
        </situação>
</report>
```

5.3.3
Considerações Finais

O estudo de caso realizado contempla a análise dos traços de execução do LS e permite uma comparação automática de *tags* internas com o catálogo de RNF. Nesse estudo de caso, há também a necessidade da avaliação por interação humana para indicar se os conteúdos das *tags* correspondem com o que indica a *tag*. Tal

análise de conteúdos para estruturas de maior volume de registros não está contemplada no SMA, uma vez que seriam necessárias estruturas de maior complexidade em substituição às estruturas de *tags* do tipo <*padrão*>.

Apesar disso, o SMA identifica artefatos gerados pelos traços de execução e informa se há nesse artefato características que condizem com o catálogo pré-estabelecido. Caso não sejam encontradas evidências (conformidades) de *tags* compatíveis com o catálogo, o SMA registra tal ocorrência e informa que o artefato não está em conformidade (<*situação*>*NÃO ATENDE*</*situação*>) às regras estabelecidas para as alternativas registradas.

A análise a partir da interação humana do LS indica que tal *software* está em conformidade com as regras estabelecidas para o *softgoal* de rastreabilidade no que se refere à partes do grupo *Fazer Rastreabilidade em Tempo de Execução*.

5.4
Aplicação do Método Sobre a Artefatos da Arquitetura do Lattesscholar

5.4.1
Especificação da Arquitetura do LS

A especificação da arquitetura do LS é feito com base no *iStarJade*[12], que muito se assemelha ao trabalho de Baia et al. (2012) no que diz respeito ao mapeamento entre diagramas i* e a linguagem iStarML [Cares et al. 2007]. Os agentes tem sua arquitetura implementada a partir de modelos i* com apoio de um padrão de representação textual *iStarML* [Cares et al. 2007]. Tal representação é compatível com XML e a partir desse são criadas as estruturas da arquitetura dos agentes. Com a arquitetura pronta, compatível com os modelos i*, são implementadas os comportamentos dos agentes, geralmente codificados em 'classes Java'[13]. A Figura 35 apresenta uma caracterização do funcionamento da infraestrutura. As estrandas para a infraestrutura *iStarJade* são feitas a partir dos modelos i* e as customizações ou implementações dos comportamentos dos agentes.

[12] www.les.inf.puc-rio.br/wiki/images/e/eb/Eduardo_2010_2.ppt
[13] http://code.google.com/p/istarjade/source/browse/trunk/IstarJade/?r=34#IstarJade%2Fsrc%2Fistar

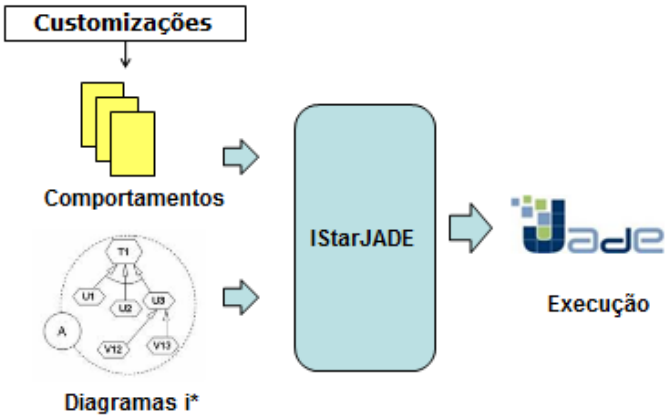


Figura 35: Mecanismo de Funcionamento com uso do iStarJade [Almentero e Cunha 2010].

Dois diagramas são utilizados para representar as classes envolvidas no *iStarJade*. No primeiro (Figura 36) são representados a classe ator, que é especializada em Agente, Papel e Position; e os elementos do diagrama podem ser Recurso, Tarefa, Meta flexivel ou Meta. No Segundo (Figura 37) são detalhadas as classes para que se possa perceber a decomposição dos elementos em camadas de interface, objeto e classe implementada.

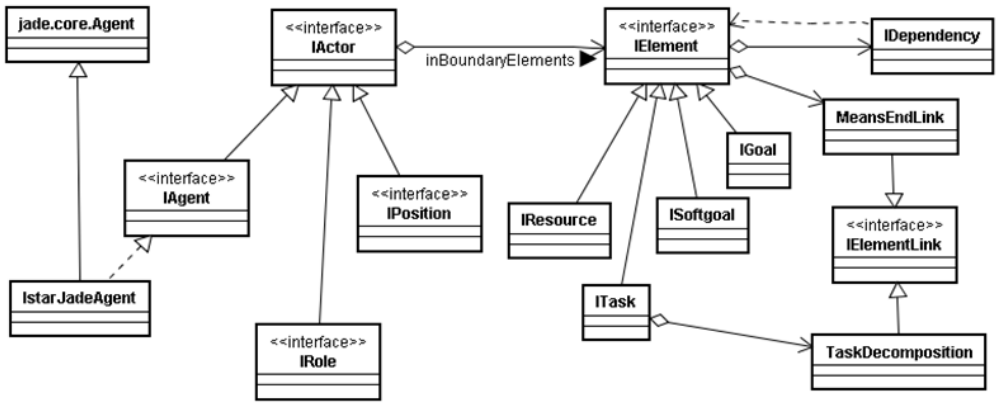


Figura 36: Classes do iStarJade [Almentero e Cunha 2010].

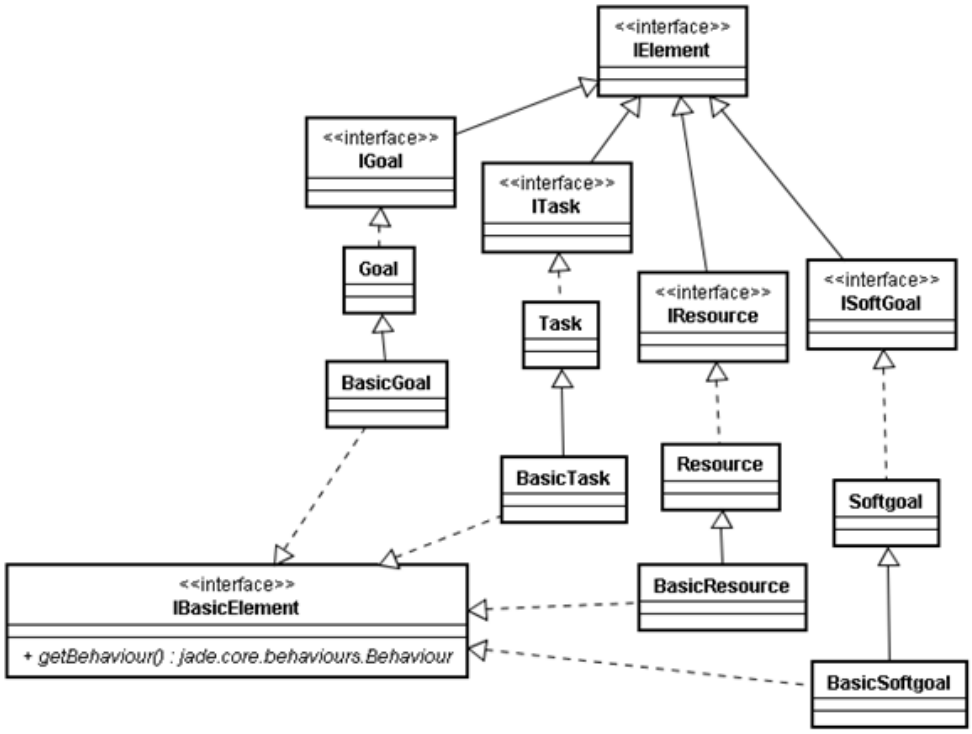


Figura 37: Detalhamento das classes [Almentero e Cunha 2010].

A Figura 38 apresenta o mapeamento entre os principais elementos de i* em *JADE/Java*. A classe *IstarJADEAgent* é uma especialização da classe *JADE.core.Agent* e implementa a interface *IAgent* (que representa agentes em i*) [Almentero e Cunha 2010].

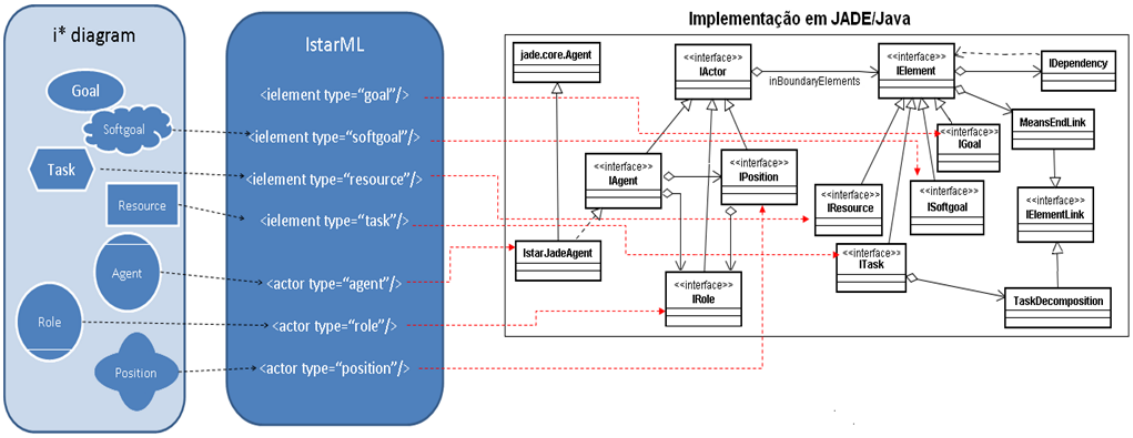


Figura 38: Mapeamento entre os principais elementos de i* [Almentero e Cunha 2010].

A Figura 39 apresenta o mapeamento dos *links* entre elementos (*ielementLink*) e os comportamentos que são instanciados automaticamente nos agentes (*IStarJADEAgent*). Os links *means-end* (que representam a seleção de alternativas – "ou" lógico) são mapeados para comportamentos *MeansEndUniqueBehaviour*. Estes comportamentos determinam a alternativa a ser escolhida. Uma vez escolhida a

alternativa, o comportamento correspondente ao elemento é instanciado. Os *links* decomposition (que representam decomposição de tarefas - “e” lógico) são mapeados em comportamentos *SequentialTaskBehaviour*. Estes últimos determinam que todos os comportamentos dos subelementos devem ser executados [Almentero e Cunha 2010].

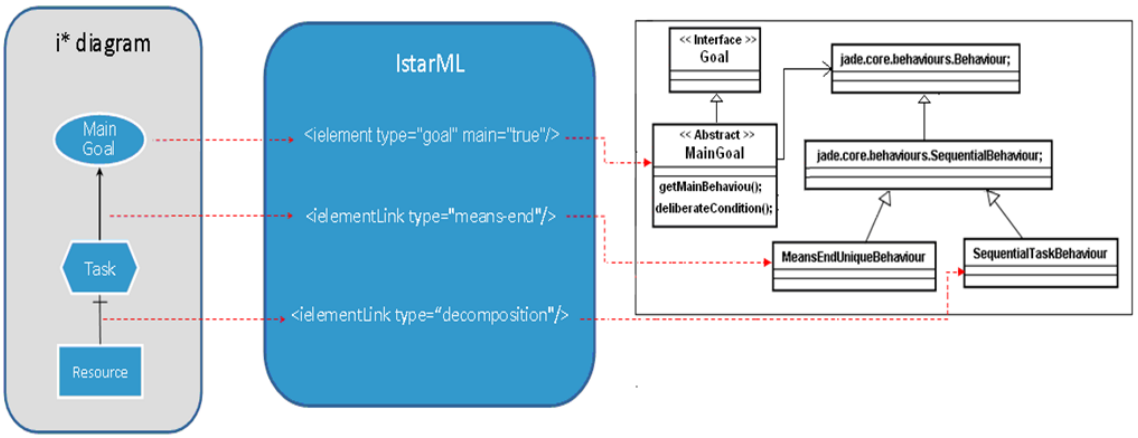


Figura 39: Mapeamento dos links entre elementos (ielementLink) e os comportamentos dos agentes [Almentero e Cunha 2010].

Os comportamentos são adicionados aos agentes de forma recursiva até que se chegue a um elemento básico (nó folha no modelo SR). Os elementos básicos devem possuir uma classe que implemente a interface *IBasicElement*, que possui o método *getBehaviour()*: *JADE.core.Behaviour*. A classe destes elementos e adicionada aos respectivos agentes via *Java Reflection*. A Figura 40 a seguir mostra o mapeamento uma tarefa básica (elemento básico) e a classe abstrata *BasicTask* [Almentero e Cunha 2010].

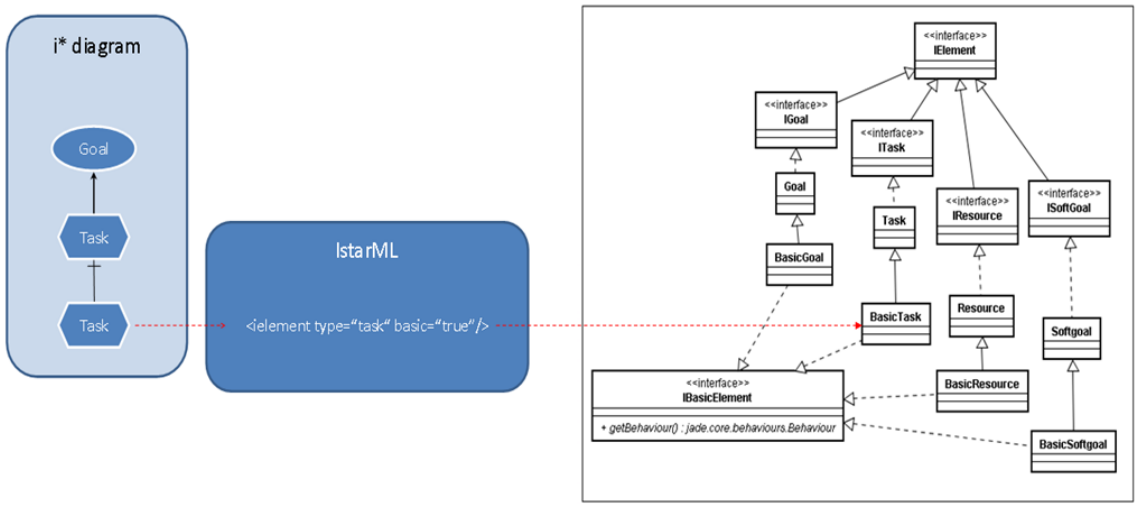


Figura 40: Mapeamento uma tarefa básica (elemento básico) e a classe abstrata BasicTask [Almentero e Cunha 2010].

O modelo SR do LS baseado em i* segue conforme apresentado na Figura 41. É possível perceber os agentes envolvidos: Lattes, Consolidador e Scholar. As tarefas e objetivos de cada agentes estão presentes em seu campo de atuação demarcado pelo circulo pontilhado [Almentero e Cunha 2010].

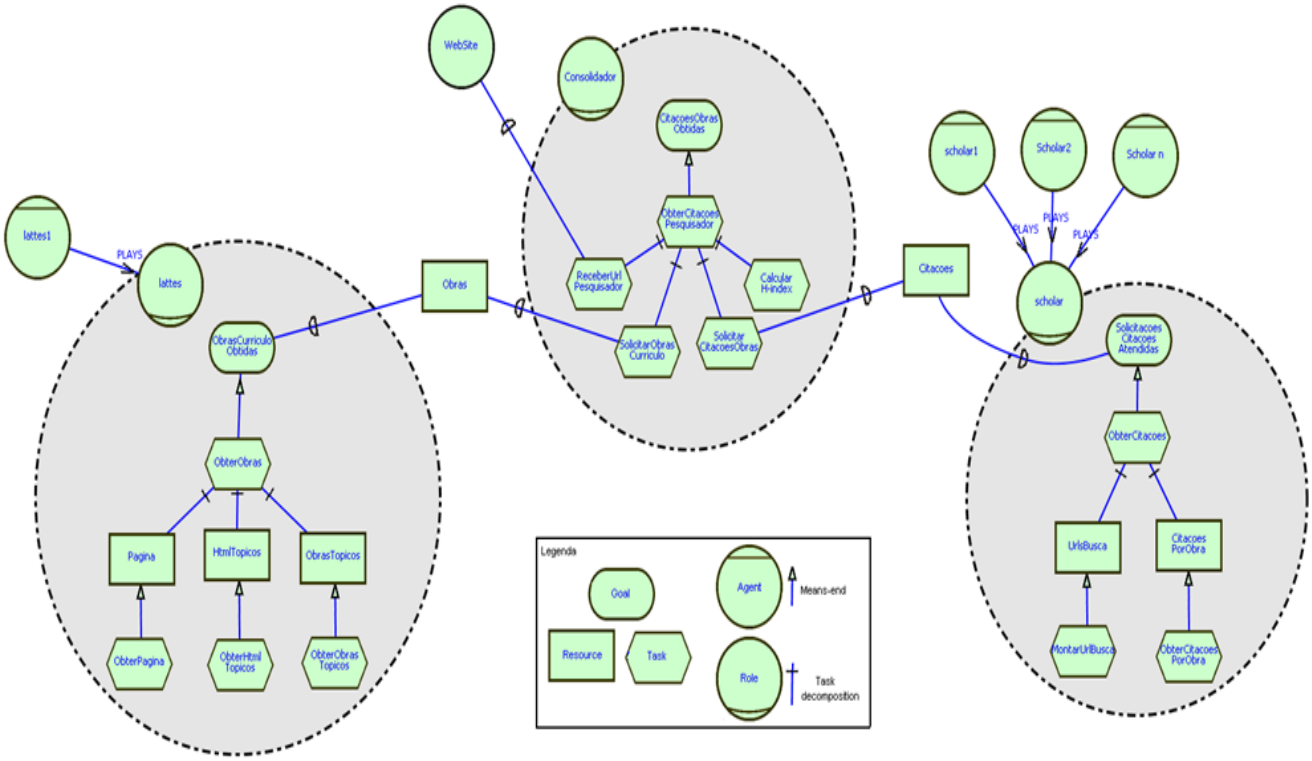


Figura 41: Modelo SR do LS [Almentero e Cunha 2010].

O XML mapeado do modelo representa os elementos disponíveis no modelo SR do LA [Almentero e Cunha 2010]. É possível perceber estruturas do tipo *Role*, *Decomposition*, *Task*, *Resource*, *Goal*, *Position*, *Actor, Agent, Soft-Goal, Contribution, Actor Boundary, Means-end* [Yu 1995] que estão presentes no modelo. O artefato em XML [Almentero e Cunha 2010] faz parte de arquivos com extensão .iml (iStarML) propostos para a implementação da arquitetura do LS.

```
<?xml version="1.0"?>
<istarml version="1.0">
      <diagram name="lattes-scholar">
            <actor id="2" type="role" name="scholar">
                  <boundary>
                        <ielement id="2.1.1" type="task" name="MontarUrlBusca" basic="true"/>
                        <ielement id="2.1" type="resource" name="UrlsBusca">
                              <ielementLink type="means-end" run-mode="unique">
                                    <ielement iref="2.1.1"/>
                              </ielementLink>
                        </ielement>
                        <ielement id="2.2.1" type="task" name="ObterCitacoesPorObra" basic="true"/>
                        <ielement id="2.2" type="resource" name="CitacoesPorObra">
                              <ielementLink type="means-end" run-mode="unique">
                                    <ielement iref="2.2.1"/>
                              </ielementLink>
                        </ielement>
                        <ielement id="2.3" type="task" name="ObterCitacoes">
                              <ielementLink type="decomposition" run-mode="sequential">
                                    <ielement iref="2.1"/>
                                    <ielement iref="2.2"/>
                              </ielementLink>
                        </ielement>
                        <ielement id="2.4" type="goal" name="CitacoesObras" main="true">
                              <ielementLink type="means-end" run-mode="unique">
                                    <ielement iref="2.3"/>
                              </ielementLink>
                        </ielement>
                  </boundary>
            </actor>
            <actor id="11" type="agent" name="scholar1">
                  <actorLink type="plays" aref="2"/>
            </actor>
      </diagram>
</istarml>
```

O estudo de caso a ser descrito nas seções posteriores tem por objetivo analisar se *software* criado a partir das estruturas sugeridas estão em conformidade com características pré-estabelecidas no catálogo de RNF.

5.4.2
Aplicação do Método em Artefatos da Arquitetura do LS

A aplicação do método no *software* LS foi feita a partir da análise dos RNFs de Entendimento, como: "*Capacidade é identificada a partir da aferição de práticas que implementem características de dependência, compositividade, detalhamento, divisibilidade, validade, controlabilidade, verificabilidade, rastreabilidade e explicação*"; e Informativo, "*Capacidade é identificada a partir da aferição de práticas que implementem características clareza, consistência, integridade, corretude, acurácia, atualidade, completeza, comparabilidade."* [CTS 2013], ambos definidos no CTS (2013) e formalizado em [Cappelli 2009]

Foram feitas análises das qualidades de Detalhamento, presente no SIG de Entendimento; e Consistência, presente no SIG de Informativo. A análise das

características do catálogo de RNF no artefato da arquitetura do LS é feita de forma automática pelo SMA a partir das configurações presentes no catálogo.

As atividades do método foram aplicadas conforme abaixo:

A1 - Criar SIG

O resultado final da A1, além da relação dos *softgoals*, é o SIG apresentado na Figura 42 a partir da representação da decomposição das qualidades que caracterizam Entendimento.

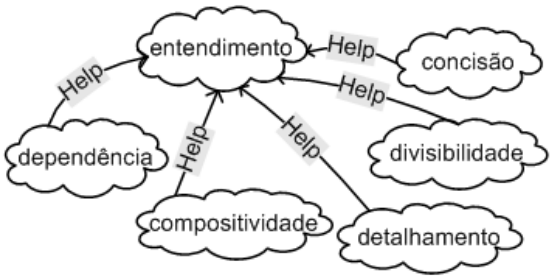


Figura 42: SIG para Entendimento [Cappelli 2009].

A decomposição de Entendimento é feito por qualidades que se relacionam com o *softgoal* principal através de um elo de ligação do tipo Help, o que indica que há uma contribuição positiva das qualidades. Para esse estudo de caso é feita uma análise do *softgoal* Detalhamento e consequentemente a potencialização de Auditabilidade.

O resultado final de A1 para as decomposições das qualidades que caracterizam Informativo são apresentados na Figura 43. Para esse caso é estudado a aferição para o *softgoal* de Consistência.

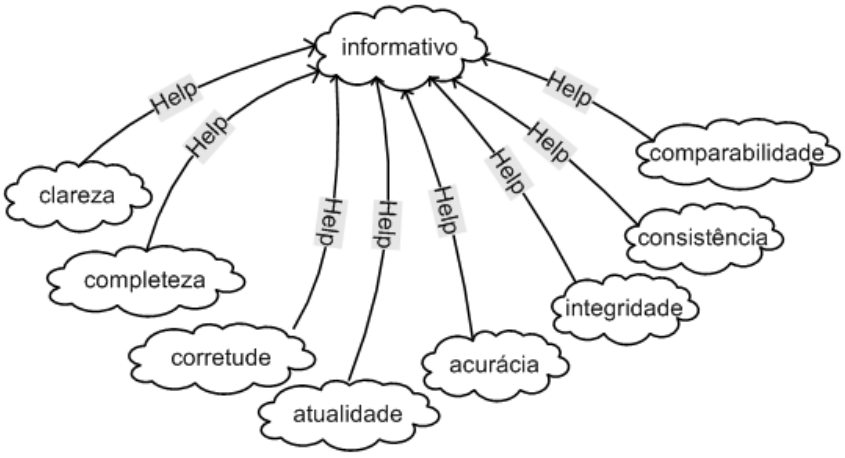


Figura 43: SIG Informativo [Cappelli 2009].

A2 – Definir *patterns*

Para o *Softgoal* Detalhamento

A definição de Detalhamento no CTS corresponde à "*Capacidade de ser descrito em minúcias.*". A aplicação da A2 para o *softgoal* resulta em um detalhamento de grupos, questões e alternativas de respostas para as questões. Os grupos são: 1) Usar estruturação de dados (variáveis) e funções (comandos); 2) Usar nomes adequados; 3) Explicar o *software.* Os grupos e questões podem ser vistos na Figura 44.

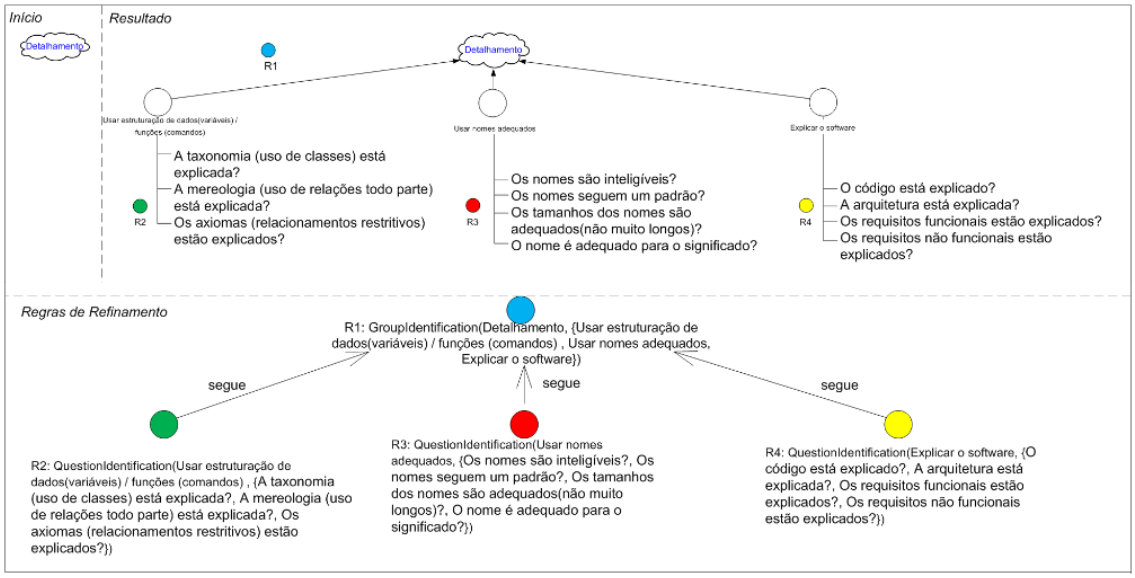


Figura 44: Patterns do *softgoal* Detalhamento [CTS 2013].

A definição de Consistência no CTS corresponde à "*Capacidade de ser isento de contradição, e ao longo do tempo obter resultados equivalentes para várias medições de um mesmo item.*". A aplicação da A2 para o *softgoal* resulta em um detalhamento de grupos, questões e alternativas de respostas para as questões. Os grupos são: 1) Identificar relacionamentos entre partes; 2) Organizar estruturas; 3) Implementar restrições previamente definidas; 4) Monitorar consistência. Os grupos e questões podem ser vistos na Figura 45.

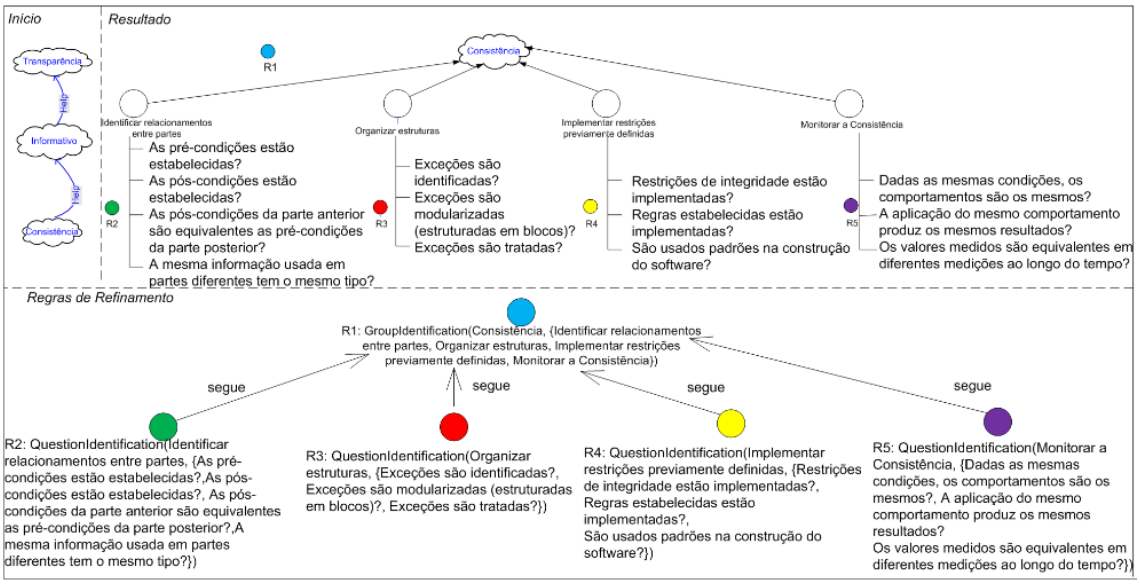


Figura 45: Patterns do *softgoal* Consistência [CTS 2013].

A aplicação do método segue conforme estudos descritos anteriormente e propõem a criação das estruturas conforme Tabelas 12 e 13.

Tabela 12: Relação de grupos, questões e operacionalizações para o *softgoal* Detalhamento.

Grupo:	**5 Usar nomes adequados.**
Questão 1:	5.1 Os nomes seguem um padrão?
Operacionalizações:	5.1.1 Usar os mesmos nomes dos tipos de elementos do diagrama i* para a definição da arquitetura de agentes em SMA.

Tabela 13: Relação de tópicos, questões e operacionalizações para o *softgoal* Consistência.

Grupo:	**6 Organizar estruturas.**
Questão 1:	6.1 Exceções são tratadas?
Operacionalizações:	6.1.1 Usar, em linguagem Java, a obrigatoriedade de tratamento de exceções a partir de declaração do tipo *checked-exception*[14] com uso de estruturas do tipo *catch* ou *trows.*

A3 – Configurar XML

Para esse estudo de caso serão apresentadas as etapas de A3 e ao final a estrutura do XML.

→ Configuração de grupos, questões e operacionalizações:

Definidas as estruturas anteriormente descritas, faz-se necessária sua transposição para uma arquitetura padronizada em XML definida como

[14] https://wikis.oracle.com/pages/viewpage.action?pageId=30146595

catálogo XML RNF (APÊNDICE A) que servirão como regras para os agentes do SMA. Sua transcrição é feita a partir das atividades propostas do detalhamento da A3 Configurar XML. Suas sub-atividades descritas como A3.1 Configurar *Objetive patterns* (estruturas XML), A3.2 Configurar *Questions patterns* (estruturas XML), A3.3 Configurar *Alternative patterns* (estruturas XML) e A3.4 Configurar Variáveis essenciais e seus sinônimos (estruturas XML) serão utilizadas para a composição da estrutura XML do catálogo de RNF que servirá como regra para as ações dos agentes do SMA. O XML será apresentado ao final.

→ Configuração de variáveis essenciais, sinônimos e padrões:

Após as execuções das atividades 3.1 a 3.3 é necessário executar A3.4 Configurar Variáveis para que se estabeleçam marcadores vinculados às operacionalizações para que as mesmas possam ser identificadas no artefato.

A aplicação da atividade **A3.4 Configurar Variáveis essenciais e seus sinônimos** é feita inicialmente para o Grupo 1, Questão 1.1 e Operacinalização 1.1.1, bem como a atividade **A3.5 Configurar Padrões** deram origem às descritas conforme o resultado apresentado a seguir.

Tabela 14: Inserção de Variáveis Essenciais, Sinônimos e Padrões para o *softgoal* Detalhamento.

Grupo:	**5 Usar nomes adequados.**
Questão 1:	5.1 Os nomes seguem um padrão?
Operacionalização:	5.1.1 Usar os mesmos nomes dos tipos de elementos do diagrama i* para à definição da arquitetura de agentes em SMA.
Variável Essencial:	tipoElemento
Noção:	Tipos de elementos utilizados para criação de diagramas i*.
Sinônimos:	typeElement, type, tipo
Padrão:	*Role*, *Decomposition*, *Task*, *Resource*, *Goal*, *Position*, *Actor, Agent, Soft-Goal, Contribution, Actor Boundary, Means-end*

Tabela 15: Inserção de Variáveis Essenciais, Sinônimos e Padrões para o *softgoal* Consistência.

Grupo:	**6 Organizar estruturas.**
Questão 2:	6.1 Exceções são tratadas?
Operacionalizações:	6.1.1 Usar, em linguagem Java, a obrigatoriedade de tratamento de exceção a partir de declaração do tipo *checked-exception* com uso de estruturas do tipo *cath* ou

	trow.
Variável Essencial:	Não se aplica
Noção:	Não se aplica
Sinônimos:	Não se aplica
Padrão:	catch, trows

O catálogo XML RNF para as estruturas definidas até o momento para o estudo de caso, após execução de A3.1, A3.2 e A3.3, segue conforme apresentado a seguir:

```
... (trecho suprimido propositalmente)
<patternDecomposicaoNivel_2>
        <idDecomposicaoNivel_2>Detalhamento</idDecomposicaoNivel_2>
        <patternSelecao>HELP</patternSelecao>

        <patternGrupo>
                <idGrupo>5</idGrupo>
                <tituloGrupo> Usar nomes adequados.</tituloGrupo>

                <patternQuestoes>
                        <idQuestao>5.1</idQuestao>
                        <tituloQuestao>Os nomes seguem um padrão?</tituloQuestao>

                                <patternAlternativas>
                                        <idAlternativa>5.1.1</idAlternativa>
                                        <tituloAlternativa> Usar os mesmos nomes dos tipos de elementos do diagrama i* para
                                                        à definição da arquitetura de agentes em SMA.
                                         </tituloAlternativa>
                                        <patternSelecao>RESPONDE</patternSelecao>
                                        <variaveisEssenciais>
                                                <idVariaveisEssenciais>tipoElemento=</idVariaveisEssenciais>
                                                <nocaoVariaveisEssenciais>Tipos de elementos utilizados para criação de
                                                        diagramas i*. </nocaoVariaveisEssenciais>
                                                <sinonimosVariaveisEssenciais>
                                                        <idSinonimosVariaveisEssenciais>typeElement=</idSinonimosVariaveisEssenciais>
                                                        <idSinonimosVariaveisEssenciais>type=</idSinonimosVariaveisEssenciais>
                                                        <idSinonimosVariaveisEssenciais>tipo=</idSinonimosVariaveisEssenciais>
                                                </sinonimosVariaveisEssenciais>

                                                <padrao>
                                                        <idPadrao>  Role, Decomposition, Task, Resource, Goal, Position, Actor,
                                                        Agent, Soft-Goal, Contribution, Actor Boundary, Means-end</ idPadrao>
                                                </padrao>
                                        </variaveisEssenciais>

... (trecho suprimido propositalmente)


... (trecho suprimido propositalmente)
<patternDecomposicaoNivel_2>
        <idDecomposicaoNivel_2>Consistência</idDecomposicaoNivel_2>
        <patternSelecao>HELP</patternSelecao>

        <patternGrupo>
                <idGrupo>6</idGrupo>
                <tituloGrupo> Organizar estruturas.</tituloGrupo>

                <patternQuestoes>
                        <idQuestao>6.1</idQuestao>
                        <tituloQuestao>Exceções são tratadas?</tituloQuestao>

                                <patternAlternativas>
                                        <idAlternativa>6.1.1</idAlternativa>
                                        <tituloAlternativa>Usar, em linguagem Java, a obrigatoriedade de tratamento de
```

```
exceção a partir de declaração do tipo checked-exception com uso de estruturas
do tipo cath ou trow.
 </tituloAlternativa>
<patternSelecao>RESPONDE</patternSelecao>
<variaveisEssenciais>
        <idVariaveisEssenciais> </idVariaveisEssenciais>
        <nocaoVariaveisEssenciais> </nocaoVariaveisEssenciais>
        <sinonimosVariaveisEssenciais>
            <idSinonimosVariaveisEssenciais> </idSinonimosVariaveisEssenciais>
            <idSinonimosVariaveisEssenciais> </idSinonimosVariaveisEssenciais>
        </sinonimosVariaveisEssenciais>

        <padrao>
            <idPadrao>catch, trows</ idPadrao>
        </padrao>
</variaveisEssenciais>
```

... (trecho suprimido propositalmente)

A4 – Configurar *Software*

Como discutido na introdução sobre a arquitetura do LS, seção 5.4.1 Especificação da Arquitetura do LS, há em sua implementação duas estruturas. A primeira baseada em iStarML que definia arquitetura dos agentes envolvidos no LS e a segunda a implementação do comportamento dos mesmo. Dessa forma, a arquitetura do LS possui arquivos com estruturas diferenciadas.

O trecho de código XML a seguir, também disponível de forma aberta na web[15], apresenta a configuração do papel *Scholar* em uma estrutura de arquivo baseada no iStarML. Outros arquivos com a extensão .iml são utilizados no LS para a implementação das diversas arquiteturas de agentes e papeis do SMA.

```
<?xml version="1.0"?>
 <istarml version="1.0">
        <diagram name="lattes-scholar">
            <actor id="2" type="role" name="scholar">
                <boundary>
                    <ielement id="2.1.1" type="task" name="MontarUrlBusca" basic="true"/>
                    <ielement id="2.1" type="resource" name="UrlsBusca">
                        <ielementLink type="means-end" run-mode="unique">
                            <ielement iref="2.1.1"/>
                        </ielementLink>
                    </ielement>
                    <ielement id="2.2.1" type="task" name="ObterCitacoesPorObra"
                basic="true"/>
                    <ielement id="2.2" type="resource" name="CitacoesPorObra">
                        <ielementLink type="means-end" run-mode="unique">
                            <ielement iref="2.2.1"/>
                        </ielementLink>
```

[15] http://code.google.com/p/istarjade/source/browse/trunk/IstarJade/file/scholar.iml?spec=svn4&r=4

```
                    </ielement>

                    <ielement id="2.3" type="task" name="ObterCitacoes">
                        <ielementLink type="decomposition" run-mode="sequential">
                            <ielement iref="2.1"/>
                            <ielement iref="2.2"/>
                        </ielementLink>
                    </ielement>
                    <ielement id="2.4" type="resource" name="CitacoesObras" main="true">
                        <ielementLink type="means-end" run-mode="unique">
                            <ielement iref="2.3"/>
                        </ielementLink>
                    </ielement>
                </boundary>
            </actor>

            <actor id="11" type="agent" name="scholar1">
                <actorLink type="plays" aref="2"/>
            </actor>
        </diagram>
</istarml>
```

De outro lado, estão arquivos com implementações dos comportamentos dos agentes codificados em Java, disponível de forma aberta na web[16], conforme pode ser visto em um exemplo:

```
package istar.onto;
import istar.IActor;
import istar.IElement;
import istar.impl.AbstractElement;
import istar.impl.AbstractElementLink;
//import istar.impl.AbstractKnowledge;
import istar.impl.Belief;
import istar.impl.ContributionLink;
import istar.impl.Dependency;
import istar.impl.Goal;
import istar.impl.MeansEndLink;
import istar.impl.Resource;
import istar.impl.Softgoal;
import istar.impl.Task;
import istar.impl.TaskDecompositionLink;
import jade.content.onto.BasicOntology;
import jade.content.onto.CFReflectiveIntrospector;
import jade.content.onto.Ontology;
import jade.content.onto.OntologyException;
import jade.content.schema.AgentActionSchema;
import jade.content.schema.ConceptSchema;
import jade.content.schema.ObjectSchema;
import jade.content.schema.PrimitiveSchema;
public class IstarOntology extends Ontology implements IstarVocabulary{
    // The name identifying this ontology
    public static final String ONTOLOGY_NAME = "Istar-Ontology";

    private static Ontology theInstance = new IstarOntology();
```

[16] http://code.google.com/p/istarjade/source/browse/trunk/IstarJade/src/istar/onto/IstarOntology.java?r=34

```
    public static Ontology getInstance(){
        return theInstance;
    }

    // Private constructor
    private IstarOntology(){
    // The istar ontology extends the basic ontology
        super(ONTOLOGY_NAME, BasicOntology.getInstance(), new CFReflectiveIntrospector());
        try {

            PrimitiveSchema stringSchema  = (PrimitiveSchema)getSchema(BasicOntology.STRING);
            PrimitiveSchema integerSchema = (PrimitiveSchema)getSchema(BasicOntology.INTEGER);
            PrimitiveSchema floatSchema = (PrimitiveSchema)getSchema(BasicOntology.FLOAT);
            PrimitiveSchema booleanSchema = (PrimitiveSchema)getSchema(BasicOntology.BOOLEAN);

            ConceptSchema aidSchema = (ConceptSchema)getSchema(BasicOntology.AID);

            // Concepts
            ConceptSchema iactorSchema = new ConceptSchema(IACTOR);
            ConceptSchema iElementSchema = new ConceptSchema(IELEMENT);
            ConceptSchema abstractElementSchema = new ConceptSchema(ABSTRACTELEMENT);
            ConceptSchema goalSchema = new ConceptSchema(GOAL);
            ConceptSchema resourceSchema = new ConceptSchema(RESOURCE);
            ConceptSchema softgoalSchema = new ConceptSchema(SOFTGOAL);
            ConceptSchema taskSchema = new ConceptSchema(TASK);
            ConceptSchema abstractElementLinkSchema = new
ConceptSchema(ABSTRACTELEMENTLINK);
            ConceptSchema taskDecompositionSchema = new
ConceptSchema(TASKDECOMPOSITIONLINK);
            ConceptSchema meanEndSchema = new ConceptSchema(MEANSENDLINK);
            ConceptSchema contributionSchema = new ConceptSchema(CONTRIBUTIONLINK);
            ConceptSchema dependenceSchema = new ConceptSchema(DEPENDENCY);
            //ConceptSchema geographicalContextSchema = new
ConceptSchema(GEOGRAPHICALCONTEXT);
            //ConceptSchema knowledgeSchema = new ConceptSchema(KNOWLEDGE);
            ConceptSchema beliefSchema = new ConceptSchema(BELIEF);

            iactorSchema.add(IACTOR_ID, stringSchema);
            iactorSchema.add(IACTOR_NAME, stringSchema);
            iactorSchema.add(IACTOR_DEPENDEEELEMENTS, iElementSchema, 0,
ObjectSchema.UNLIMITED);
            iactorSchema.add(IACTOR_DEPENDERELEMENTS, iElementSchema, 0,
ObjectSchema.UNLIMITED);
            iactorSchema.add(IACTOR_INBOUNDARYELEMENTS, iElementSchema, 0,
ObjectSchema.UNLIMITED);

            iElementSchema.add(IELEMENT_ID, integerSchema);
            iElementSchema.add(IELEMENT_NAME, stringSchema);
            iElementSchema.add(IELEMENT_TOPIC, stringSchema);
            iElementSchema.add(IELEMENT_OWNER, iactorSchema);
            iElementSchema.add(IELEMENT_DEPENDENCIES, dependenceSchema, 0,
ObjectSchema.UNLIMITED);
            iElementSchema.add(IELEMENT_MEANSENDLINKS, meanEndSchema, 0,
ObjectSchema.UNLIMITED);
            iElementSchema.add(IELEMENT_SUPPORTS, abstractElementSchema, 0,
ObjectSchema.UNLIMITED);

            abstractElementSchema.addSuperSchema(iElementSchema);
            goalSchema.addSuperSchema(abstractElementSchema);
            goalSchema.add(GOAL_STATE, integerSchema);
            resourceSchema.addSuperSchema(abstractElementSchema);
            resourceSchema.add(RESOURCE_VALUE, floatSchema);
```

```
                resourceSchema.add(RESOURCE_QUANTITY, integerSchema);
                softgoalSchema.addSuperSchema(abstractElementSchema);
                softgoalSchema.add(SOFTGOAL_STATE, integerSchema);
                softgoalSchema.add(SOFTGOAL_CONTRIBUTESFOR, contributionSchema, 0,
ObjectSchema.UNLIMITED);

                taskSchema.addSuperSchema(abstractElementSchema);
                taskSchema.add(TASK_COST, floatSchema);
                taskSchema.add(TASK_PROCESSED, booleanSchema);
                taskSchema.add(TASK_PARAMS, iElementSchema, 0, ObjectSchema.UNLIMITED);
                taskSchema.add(TASK_MYDECOMPOSITIONLINK, taskDecompositionSchema, 0,
ObjectSchema.UNLIMITED);
                taskSchema.add(TASK_MEANSFOR, meanEndSchema, 0, ObjectSchema.UNLIMITED);
                //geographicalContextSchema.add(GEOGRAPHICALCONTEXT_NAME, stringSchema);
                //geographicalContextSchema.add(GEOGRAPHICALCONTEXT_TYPE, integerSchema);
                //geographicalContextSchema.add(GEOGRAPHICALCONTEXT_ADDRESS, stringSchema);

                //knowledgeSchema.addSuperSchema(abstractElementSchema);
                //knowledgeSchema.add(KNOWLEDGE_CONTEXT, geographicalContextSchema);
                //knowledgeSchema.add(KNOWLEDGE_TOPIC, iElementSchema);

                beliefSchema.addSuperSchema(abstractElementSchema);
                beliefSchema.add(BELIEF_TOPIC, iElementSchema);
                beliefSchema.add(BELIEF_PROBABILITY, floatSchema);

                abstractElementLinkSchema.add(ABSTRACTELEMENTLINK_TARGETELEMENT,iElementSchem
a);
                abstractElementLinkSchema.add(ABSTRACTELEMENTLINK_NAME, stringSchema);
                abstractElementLinkSchema.add(ABSTRACTELEMENTLINK_OWNER, iElementSchema);
                meanEndSchema.addSuperSchema(abstractElementLinkSchema);

                taskDecompositionSchema.addSuperSchema(abstractElementLinkSchema);
                taskDecompositionSchema.add(TASKDECOMPOSITIONLINK_TYPE, integerSchema);

                contributionSchema.addSuperSchema(abstractElementLinkSchema);
                contributionSchema.add(CONTRIBUTIONLINK_VALUE, integerSchema);

                dependenceSchema.add(DEPENDENCY_NAME, stringSchema);
                dependenceSchema.add(DEPENDENCY_TYPE, integerSchema);
                dependenceSchema.add(DEPENDENCY_DEPENDUM, iElementSchema);
                dependenceSchema.add(DEPENDENCY_DEPENDER, iactorSchema);
                dependenceSchema.add(DEPENDENCY_DEPENDEE, iactorSchema);
                dependenceSchema.add(DEPENDENCY_DEPENDERELEMENT, iElementSchema);
                dependenceSchema.add(DEPENDENCY_DEPENDEEELEMENT, iElementSchema);

                add(iactorSchema, IActor.class);
                add(iElementSchema, IElement.class);
                add(goalSchema, Goal.class);
                add(abstractElementSchema, AbstractElement.class);
                add(resourceSchema, Resource.class);
                add(softgoalSchema, Softgoal.class);
                add(taskSchema, Task.class);
                //add(geographicalContextSchema, GeographicalContext.class);
                //add(knowledgeSchema, AbstractKnowledge.class);
                add(beliefSchema, Belief.class);
                add(abstractElementLinkSchema, AbstractElementLink.class);
                add(meanEndSchema, MeansEndLink.class);
                add(taskDecompositionSchema, TaskDecompositionLink.class);
                add(contributionSchema, ContributionLink.class);
                add(dependenceSchema, Dependency.class);
                // Actions
                AgentActionSchema accomplishSchema = new AgentActionSchema(ACCOMPLISH);
                accomplishSchema.add(ACCOMPLISH_ELEMENT, iElementSchema);
```

```
                accomplishSchema.add(ACCOMPLISH_AGENT, aidSchema);
                AgentActionSchema accomplishTaskSchema = new
AgentActionSchema(ACCOMPLISH_TASK);
                accomplishTaskSchema.addSuperSchema(accomplishSchema);
                accomplishTaskSchema.add(ACCOMPLISH_PARAMS, iElementSchema, 0,
ObjectSchema.UNLIMITED);

                add(accomplishSchema);
                add(accomplishTaskSchema);

            }
            catch (OntologyException oe) {
                oe.printStackTrace();
            }
        }

}
```

Os dois tipos de artefatos, com extensões e formatos distintos, foram submetidos para análise. O primeiro, referente análise de Detalhamento, foi feito a partir das configurações no catálogo com base no item *6.1.1. Usar os mesmos nomes dos tipos de elementos do diagrama i* para à definição da arquitetura de agentes em SMA* e feita a análise a partir dos arquivos com extensão.iml em formato XML. O segundo, referente à análise de Consistência, aplicados a partir da alternativa *7.1.1 Usar, em linguagem Java, a obrigatoriedade de tratamento de exceção a partir de declaração do tipo checked-exception com uso de estruturas do tipo catch ou trow* foi feito a partir das classes Java desenvolvidas, conforme segundo exemplo demonstrado acima.

A5 – Operacionalizar Agentes

A execução atividades A5.1, A5.2 e A5.3 representam o funcionamento desde a captura do artefato até sua análise a partir da comparação de marcações ou conteúdos com as regras definidas no catálogo.

O resultado satisfatório na análise de conformidade é dado pela identificação de variáveis essenciais e padrão encontrados nos modelos da arquitetura do LS. Para a análise da alternativa 5.1.1, a seguir, foi gerado um resultado onde, na seção <variáveis></variáveis> são reistrados as *tags* com variáveis essenciais encontradas nos arquivos XML dos modelos LS, enquanto na seção <padrões></padrões> as *tags* referentes aos conteúdos dos padrões registrados no catálogo. As variáveis essenciais e os padrões associados às alternativas, e essas associadas às questões, demonstram que os modelos satisfazem à implementação do RNF Detalhamento.

Alternativa: 5.1.1 Usar os mesmos nomes dos tipos de elementos do diagrama i* para à definição da arquitetura de agentes em SMA.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<report tipo="RNF" objetivo="ANÁLISE">
    <artefato>scholar.iml</artefato>
    <local>usr/local/logs</local>
     <patternIdentificacao>Transparencia</patternIdentificacao>
        <patternDecomposicaoNivel_1>
            <idDecomposicaoNivel_1>Informativo</idDecomposicaoNivel_1>
                <patternDecomposicaoNivel_2>
                    <idDecomposicaoNivel_2>Detalhamento</idDecomposicaoNivel_2>
                       <idGrupo>5</idGrupo>
                       <tituloGrupo> Usar nomes adequados. </tituloGrupo>
                       <patternQuestoes>
                            <idQuestao>5.1</idQuestao>
                             <tituloQuestao> Os nomes seguem um padrão?</tituloQuestao>
                             <patternAlternativas>
                                  <idAlternativa>5.1.1</idAlternativa>
                                   <tituloAlternativa> Usar os mesmos nomes dos tipos de elementos do diagrama i* para
                                            à definição da arquitetura de agentes em SMA.</tituloAlternativa>
                                    <situação> EM CONFORMIDADE </situação>
                                   <variáveis>
                                       <tagsVariável>type=</tagsVariável>
                                    </variáveis>
                                   <padrões>
                                        <tagPadrao>actor</tagPadrao>
                                        <tagPadrao>task</tagPadrao>
                                        <tagPadrao>role</tagPadrao>
                                        <tagPadrao>means-end</tagPadrao>
                                        <tagPadrao>resource</tagPadrao>
                                        <tagPadrao>decomposition</tagPadrao>
                                        <tagPadrao>agent</tagPadrao>
                                  </padrões>
                                 </patternAlternativas>
                <patternDecomposicaoNivel_2>
            </patternDecomposicaoNivel_1>

</report>
```

A mesma sistemática foi utilizada para a alternativa 6.1.1, só que para essa alternativa apenas o uso da seção <padrões></padrões> foi registrada, uma vez que o SMA do método para análise de conformidade identificou apenas o padrão *catch* nos modelos do LS. Padrão esse configurado inicialmente no catálogo XML para o requisito Consistência.

Alternativa: 6.1.1 Usar, em linguagem Java, a obrigatoriedade de tratamento de exceção a partir de declaração do tipo checked-exception com uso de estruturas do tipo *catch* ou *trow*.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<report tipo="RNF" objetivo="ANÁLISE">
    <artefato> IstarOntology.java</artefato>
    <local>usr/local/logs</local>
     <patternIdentificacao>Transparencia</patternIdentificacao>
        <patternDecomposicaoNivel_1>
            <idDecomposicaoNivel_1>Informativo</idDecomposicaoNivel_1>
                <patternDecomposicaoNivel_2>
                    <idDecomposicaoNivel_2>Consistência</idDecomposicaoNivel_2>
                       <idGrupo>6</idGrupo>
                       <tituloGrupo>Organizar estruturas.</tituloGrupo>
                       <patternQuestoes>
                            <idQuestao>6.1</idQuestao>
                             <tituloQuestao>Exceções são tratadas?</tituloQuestao>
```

```
                    <patternAlternativas>
                        <idAlternativa>6.1.1</idAlternativa>
                        <tituloAlternativa> Usar, em linguagem Java, a obrigatoriedade de tratamento de
                                exceção a partir de declaração do tipo checked-exception com uso de
                                estruturas do tipo cath ou trow.</tituloAlternativa>
                        <situação> EM CONFORMIDADE </situação>
                        <variáveis>
                            <tagsVariável></tagsVariável>
                        </variáveis>
                        <padrões>
                            <tagPadrao>catch</tagPadrao>
                        </padrões>
                    </patternAlternativas>
                <patternDecomposicaoNivel_2>
            </patternDecomposicaoNivel_1>
</report>
```

Nesse estudo de caso, foi considerada como não conformidade àquelas assinaturas no código XML do modelo LS que não correspondesse às *tags* presentes na seção <idPadrão> presentes no catálogo XML.

Para o estudo de caso foi necessário alterar o código do LS e inserir assinaturas fictícias de nomes “agente”, “plays” e “mean-end”, simulando erro insediro pelo desenvolvedor, para que o SMA do método pudesse fazer a comparação e distinguir esses pontos como não conformidade.

Após a execução do SMA do método o resultado foi gerado e as inconsistências registradas na seção <UnkonwStructureBase></UnkonwStructureBase> do arquivo USBase.xml, conforme pode ser visto a seguir. O arquivo do LS analisado foi o *scholar.iml* presente na pasta *usr/local/logs*.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<report tipo="unkown base" objetivo="assinaturas não conhecidas">
    <local> usr/local/logs </local>
        <situação>NÃO ATENDE
            <artefato>scholar.iml
                <UnknownStructureBase>
                    <UnknownStructure>
                        <index>1</index>
                        <name>plays</name>
                    </UnknownStructure>
                    <UnknownStructure>
                        <index>2</index>
                        <name>agente</name>
                    </UnknownStructure>
                    <UnknownStructure>
                        <index>3</index>
                        <name>mean-end</name>
                    </UnknownStructure>
                    <UnknownStructure>
                </UnknownStructureBase>
            </artefato>
        </situação>
</report>
```

5.4.3
Considerações Finais

Nesse estudo de caso foi realizada uma análise do *software* LS considerando seus artefatos de implementação no que diz respeito à arquitetura dos agentes e a seu comportamento, enquanto ações que executa. A análise foi baseada nos *softgoals* de Detalhamento e de Consistência. Os resultados apontam artefatos de implementação da arquitetura estão em conformidade com o catálogo no que se refere aos *patterns* de Detalhamento. De outro lado, artefatos da implementação das classes Java estão em conformidade com o catálogo no que se refere aos *patterns* de Consistência.

5.5
Considerações Finais sobre o Estudos de Casos

O estudo de casos da aplicação do método utilizaram as características dos padrões do CTS. Como pode ser percebido, nos resultados (*report XML*), há a referência ao termo Transparência na *tag patternIdentificacao*. Isso ocorre porque cada RNF apresentado no trabalho é uma decomposição do CTS, portanto, *softgoals* identificados para que Transparência seja satisfeita [Grupo ER PUC-Rio 2013].

Como pode-se avaliar nas Figuras 46, 47, 48 e 49 a seguir, a decomposição de Transparência em outras qualidades fornece uma variedade ampla de material para estudo de analise de RNFs. As figuras representam recortes do SIG de Transparência [CTS 2013].

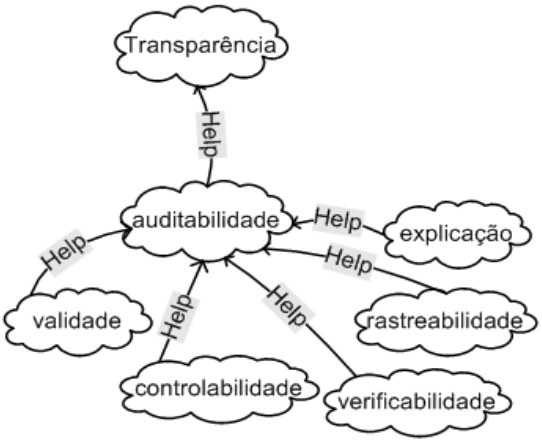


Figura 46: SIG Auditabilidade para Transparência.

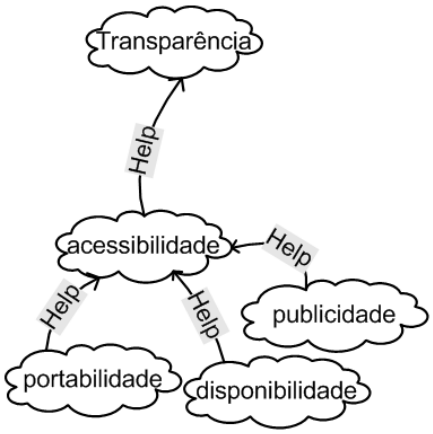


Figura 47: SIG Acessibilidade para Transparência.

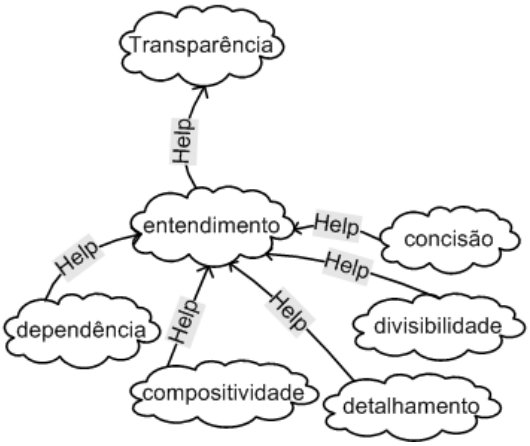


Figura 48: SIG Entendimento para Transparência.

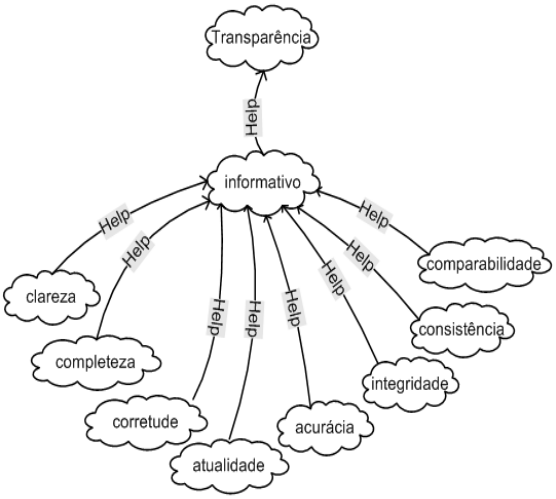


Figura 49: SIG Informativo para Transparência.

O estudo de casos foi feito a partir de dois níveis de *softgoals* do CTS *patternDecomposicaoNivel_1,* composto por Auditabilidade, Acessibilidade, Entendimento e Informativo; e *patternDecomposicaoNivel_2*, por Explicação, Rastreabilidade, Disponibilidade, Detalhamento e Consistência, esses utilizados no material de estudo a partir da exploração de seus *patterns* de nível inferior (grupos, questões e alternativas).

A sistemática da aplicação do método indica que: se o nível de Decomposição 2 (*patternDecomposicaoNivel_2*) tem seus elementos satisfeitos, os nível imediatamente acima também tem seus elementos satisifeitos. Dessa forma, a propagação segue a mesma cadeia de valores até o nível mais alto da árvore. Portanto, consequentemente Transparência é satisfeita a contento (*satisficed*) [Mylopoulos 2006], [Cunha 2007].

Essa perspectiva indica que o trabalho está intimamente relacionado à análise de RNFs. Como é utilizado o CTS (2013), organizado e sistematizado de acordo com características de Transparência de *Software* a partir de seus padrões, a tese posiciona-se também positivamente em relação à análise de conformidades do RNF de Transparência nesse contexto de *software.*

As estruturas apresentadas nas figuras acima são parte do SIG de Transparência presente na seção 2.2 RNF *framework* e apresentam alguns de seus RNFs.

É importante observar que para a sistematização do processo de análise de artefatos por SMA foi necessária a criação de uma instância de padrões. Na seção 2.4 padrões de RNFs há a figura que representa a especialização dos detalhamentos doa padrões propostos por [Supakkul et al. 2010] e [Serrano et al. 2010], já nesse trabalho

de tese, foi utilizada uma instanciação abaixo de *Padrões Alternativas/Problemas* para servir de camada de representação de estruturas passíveis de análise sintática. Essa última instância foi representada no modelo conceitual dos *patterns* de RNF na Figura 19 pelas classes Variáveis Essenciais e Padrões.

Tal procedimento pode ser visto no estudo de casos e também na representação da arquitetura (proposta na seção 4.1 Modelo Conceitual dos *Patterns)* onde variáveis essenciais, sinônimos e padrões recebem um tratamento especializado como uma última camada de representação.

O método sugerido atualmente não contempla a análise de impacto entre *softgoals*, portanto, não se pode analisar se o grau de conformidade é menor ou maior caso um ou mais *softgoals* que estejam correlacionados sejam satisfeitos. Principalmente, se *sofgoals* antagônicos como Transparência e Segurança [Capelli et al. 2010] estiverem correlacionados na arquitetura.

6.
Conclusões

Esse capítulo apresenta uma contextualização trabalho, seu resumo, apresenta trabalhos correlatos, descreve as contribuições percebidas, discorre a respeito das limitações e apresenta propostas de trabalhos futuros.

6.1
Contextualização

O presente trabalho propôs um método baseado em passos sitemáticos, desde a representação conceitula de RNFs e suas operacionalizações até a implementação de um mecanismo de *software* baseado em SMA para análise de RNF implementado em software.

A análise de RNFs implementados em *software* apresenta-se como um desafio, uma vez que sua natureza subjetiva não permite uma análise direta. É necessário, conforme sugerem Pfleeger (1997), Lamsweerde (2001) e Lamsweerde e Letier (2000) um maior detalhamento dos RNFs em elementos de menor grau de subjetividade, como é o caso de RFs.

O presente trabalho insere-se nesse contexto e propõe um método de união de técnicas de especificação registradas em catálogo utilizado por SMA para a análise dos RNFs em *software*.

6.2
Resumo

O presente trabalho de pesquisa apresentou uma proposta de método sistêmico para análise de RNFs, definidos em catálogo, que une abordagem de GORE à analise automática ou semi-automática a partir de operacionalizações efetuadas por SMA com agentes definidos pelo modelo BDI.

Aspectos conceituais foram considerados a partir de RNF *framework,* GQM, GQO e padrões de RNFs que dão suporte teórico ao que é construído enquanto método sistêmico proposto na tese. Por outro lado, para a construção do SMA foi fundamento sobre modelagem intencional, *framework* i*, painel de intencionalidades, e BDI.

Entende-se que o trabalho possuim uma prática de construção *TOP-DOWN* uma vez que inicia a partir da especificação de modelos conceituais com o detalhamento das características de determinado RNF, até às definições de camadas físicas de mais baixo nível da aplicação computacional.

Para que um mecanismo de *software* fazer a análise das conformidades de RNFs foi necessário a inserção de uma nova instância nos padrões de RNF que contempla variáveis essenciais e conteúdos padronizados que possam ser pesquisados em artefatos de Software. Esses conteúdos são associados aos padrões Alternativas (*Alternative patterns*).

Os padrões utilizados no método são registrados de forma sistemática em um catálogo XML.

6.3
Trabalhos Correlatos

6.3.1
Uso de Catálogos

No contexto da proposta da solução, outros trabalhos de pesquisa se relacionam com os temas utilizados com certo grau de proximidade. Cysneiros et al. (2003) aponta que RNFs são negligenciados ou esquecidos em projetos de *software*. Cysneiros et al. (2003) propõem ainda que RNF devem ser tratados de forma organizadas para facilitar seu entendimento, compartilhamento do conhecimento, evolução e reuso. Em Cysneiros et al (2003) há a representação de catálogos de Privacidade e de Rastreabilidade. Os autores propõem decomposições dos RNFs principais em *softgoals*, bem como operacionalizações que satisfaçam tais metas, nesse trabalho os autores fazem uso do RNF *framework* [Chung et al. 2000].

Outra representação de catálogo de RNF é o proposto por Xavier et al. (2010), onde os autores exploram conceitos de Usabilidade a partir de metas, princípios de *design* e heurísticas. As metas foram colocadas em nível mais alto de abstração para definir fatores de satisfação e estabelecem critérios de usabilidade para aceitabilidade de um sistema; os princípios de desenho são tratados como lembretes do que fornecer e do que evitar durante o *design* da interface; e heurísticas que foram tratados como princípios da usabilidade utilizados na prática, servem para avaliar a aceitabilidade das interfaces [Xavier et al. 2010]. Os autores utilizam a mesma estratégica de Cysneiros et al. (2003) quando utilizam do RNF *framework* para a representação das decomposições e relações utilizadas no catálogo.

Da mesma forma Cappelli (2009) utiliza a representação do RNF *framework* para a criação de um catálogo para Transparência de Processos Organizacionais.

A proximidade desse trabalho de tese com os anteriores citados refere-se ao uso de GORE para se estabelecer catálogos. Diferente dos trabalhos apresentados pelos autores, que focam a construção do catálogo apenas com o uso da proposta do RFN

framework [Chung et al. 2000], nessa tese são exploradas as características de construção de catálogos a partir de *patterns* [Supakkul et al. 2010] e [Serrano et al. 2010], o que conforme sugere [Serrano et al. 2010], minimiza o *gap* existente entre o modelo conceitual e as operacionalizações a partir do uso do padrão de Questões.

A elicitação de RNFs foi preocupação do trabalho de Gonzales-Baixauli et al. (2005) e os autores propõem representações de valoração a partir do que chamam de *Teoria de Constructos Personales* (TCP). A teoria é apoiada na psicologia construtiva proposta em 1955 que trata de explicar a visão do mundo de uma pessoa por meio de *constructo* (em psicologia, qualquer entidade hipotética de difícil definição dentro de uma teoria científica). Pela teoria o conjunto de conhecimentos de uma pessoa e da relação entre eles dão a base das decisões e comportamentos da pessoa, uma vez que permitem que você faça suposições e tire conclusões [Gonzales-Baixauli et al. 2005]. Os autores propõem que os conhecimento sejam adaptados para *sofgoals* para se construir SIGs de RNFs com uso de relações entre elementos funcionais e *softgoals* a partir de cinco elos de ligação com valorações do tipo: satisfeito, parcialmente satisfeito, neutro, parcialmente insatisfeito e insatisfeito. No que se refere à análise de RNF o estudo é importante, pois trata do uso do RNF *framework* a partir de uma visão diferenciada sobre a construção dos modelos de RNF e também das valorações possíveis no modelo para efeito de propagação. O artigo demonstra claramente as regras de propagação propostas pelos autores, ou seja, a combinação de elementos estereotipados por cadeias de -, --, + e ++, e os resultados esperados a partir dessas combinações. As regras de propagação propostas pelo artigo facilitam a compreensão de como estabelecer heurísticas para a verificação das diversas propagações em modelos SIG. Outras abordagens de propagação de valores estão descritos no próprio RNF *framework* [Chung et al. 2000].

A tese baseia-se em elos de ligação com estereótipos propostos por [Chung et al. 2010] para a análise das propagações dos padrões até a satisfação do RNF, similiar aos estereótipos de [Gonzales-Baixauli et al. 2005].

6.3.2
Análise de Conformidade de Requisitos

O trabalho de Baia et al. (2012) possui similitudes com a presente pesquisa. Nele são abordadas o tema de verificação de RNF, uso de base de conhecimento e motor de inferência para as análises. Apesar disso, o foco do trabalho está na análise de características do RNF de Transparência de Processos [Cappelli 2009] em diagramas i* [Yu 1995]. Os autores fazem um mapeamento entre atributos do SIG de

Transparência de Processos [Cappelli 2009] para elementos do diagrama i* (atores, metas, tarefas, recursos, entre outros), modelado a partir de um determinado domínio, e em seguida para definição de fatos da linguagem *C Language Integrated Production System* (CLIPS) [CLIPS 1991]. Dessa forma, a solução trabalha com dois modelos, o primeiro, a base de conhecimento composto pelo mapeamento dos atributos de Transparência para os elementos i*; o segundo, de elementos i* para fatos da linguagem CLIPS. Ao executarem o motor de inferência da linguagem, os autores conseguem identificar quais relações de tarefas com determinado ator. Por exemplo, O Cliente deseja fornecer financiamento adequado, tomado "Cliente" como ator e "fornecer financiamento adequado" como uma tarefa do modelo. Assim, conseguem fazer uma interpretação textual mais significativa a partir de resultados expostos em linguagem natural, sem a necessidade de análise do modelo. A análise da Transparência é dada a partir da definição de regras, em que indicam regras para as decomposições dos atributos de Transparência relacionados aos elementos do i*. Com essas relações estabelecidas mais os fatos configurados no CLIPS, o motor de inferência propõe quais atribuitos de Transparência são atendidos pelo diagrama i* definido a partir de uma realidade organizacional. Os autores não retratam no trabalho como é a relação de contribuição negativa ou positiva entre metas flexíveis antagônicas, ou seja, como um atributo de qualidade decomposto intefere negativamente no resultado do outro e consequentemente em Transparência.

Em [Baia et al 2013] e [Oliveira et al 2013] os autores do trabalho anterior, extendem a base de conhecimento e associam sinônimos aos atributos de Transparência. A mesma sistemática é aplicada para que outras qualidades possam ser verificadas a partir do motor de inferência.

No campo da análise, apesar de não focar em RNFs, o trabalho de pesquisa de Lingxiao et al. (2013) faz uso de GORE a partir do *framework* i* para a monitoração de requisitos. Os autores propõem um *framework* baseado modelos em i* para promover a análise de objetivos, no caso com foco em *hard goals*, de determinados domínios. A sistemática da abordagem é importante uma vez que propõe regras de consistência para os estados dos objetivos analisados. Dessa forma, fazem uma estruturação do *framework* a partir do modelo conceitual dos objetivos até a implementação de um algoritmo de análise das propagações existentes com elementos reparam a falha em caso do objeto monitorado não estar de acordo com os modelos estabelecidos. Tais ações de reparação são propostas para ações de *skip*, *ignore*, *retry* e *alternate* utilizadas em orquestração de computação orientada a serviços. A proposta propõe um sistema de autoadaptação a partir da monitoração efetuada.

No trabalho de [Souza 2012] há a preocupação do desenvolvimento de sistemas autoadaptativos a partir de analise de requisitos. O trabalho é fundamentado em GORE, *framework* i*, e propõe à aplicação do que chamam de *awareness requirements* (AwReqs). Os autores fazem interpretações de diagramas i*, modelados sobre determinado domínio, como agenda de reuniões, e em seguida um mapeamento de seus requisitos para padrões de comportamento funcional. Estratégias de adaptação, em caso de falhas, são vinculadas aos padrões de comportamento e o sistema proposto indica alternativas. De certa forma, caso o reconhecimento dos padrões de comportamento não sejam contrariados há uma identificação de que o requisito foi atendido.

A verificação automatica de *design patterns* é tratada nos trabalhos de [Balanyi e Ferenc 2003], [Buchli 2003], [Bansiya 1998] e [Tsantalis et al. 2006]. Nesses, aplicações automáticas analisam por comparação, conteúdos de código fonte para prognosticar se os mesmos fazem uso de estruturas propostas em *design patterns*. Por tratarem de análise de padrões, esses trabalhos assemelham-se ao objetivo do agente Consolidador na análise de padrões especificados em catálogo. Particularmente no trabalho de [Balanyi e Ferenc 2003], o trabalho tem maior semelhança uma vez que os autores também fazem uso de estruturas baseadas em linguagem XML para compor bases de conhecimento a respeito de *design patterns* e posteriormente fazer comparações com código fonte.

O método proposto na tese diferencia-se dos demais citados uma vez que não está focada apenas na análise de modelos i*, mas sim exploram a especificação de *softgoals* a partir de modelos baseados em RNF *framework* e padrões que passam a decompor os *softgoals* até operacionalizações em nível de implementação que podem ser utilizadas em *software.* O conjunto dos padrões é modelado em XML que passa a servir de regra para que agentes autônomos possam analisar se *software* está em conformidade com padões estabelecidos.

6.3.3 SMA

O trabalho de [Serrano 2011] propõe uma abordagem para anexar requisitos intencionais ao código através de heurísticas transformacionais para se obter um processo dirigido a modelos. O enfoque do trabalho baseia-se na rastreabilidade entre os artefatos e entre os elementos dos artefatos, considerando as heurísticas transformacionais como os elos, ou o rastro utilizados a partir do *framework* i*. A partir do mapeamento entre modelos e códigos pelas heurísticas, o autor propõe uma

máquina de raciocínio qualitativa para capacitar o *software* a raciocinar em termos de metas flexíveis em tempo de execução. O enfoque desse estudo de casos foi representar formalmente a subjetividade e a incerteza inerentes às metas flexíveis e ao raciocínio qualitativo. A partir do momento que foram estabelecidas as estratégias de mapeamento entre modelos e código, um motor de verificação, foram inseridos no modelos metas flexíveis alinhadas com características de Transparência abordados no trabalho de [Cappelli 2009]. O motor de verificação automatica proposto no trabalho foi desenvolvido por lógica nebulosa para verificar se diagramas i* estariam condizentes com os atributos de Transparência. O trabalho foi desenvolvido com o uso de SMA a partir da proposta de BDI, que segundo Serrano (2009), são capazes de raciocinar sobre metas e critérios de qualidade. O autor propõe que o tratamento formal para a incerteza e as subjetividades intrínsecas às metas flexíveis foi a principal contribuição, uma vez que foi aplicada a lógica nebulosa, uma teoria matemática para lidar formalmente com a incerteza, visando representar as metas flexíveis, as contribuições das tarefas para as metas flexíveis e os impactos que propagam através das contribuições quando se seleciona e executa uma tarefa [Serrano 2009]. Apesar da inovação pelo uso de agentes e motor de inferência baseado em lógica nebulosa, o trabalho se assemelha ao de Baia et al (2013), Souza (2012), Wang et al. (2007) e Lingxiao et al. (2013), uma vez que foca sua análise em diagramas de intencionalidade baseados no i*.

A análise de requisitos a partir de SMA é proposto por Sayao (2007). No trabalho estão descritas estratégias para avaliar se os requisitos especificados estão com determinado grau de qualidade. Para isso, o SMA desenvolvido utiliza de técnicas que possibilitem: criação de visões de requisitos a partir de determinado recorte da especificação por meio de buscas de palavras-chave ou expressões; detecção de mudanças de requisitos, criação e atualização de léxico da aplicação e detecção de erros, omissões e discrepâncias no conjunto de requisitos; uso intensive de técnicas de processamento de linguagem natural para avaliação de documentos gerados ou manipulados no processo de requisitos. Apesar da aparente semelhança, o trabalho de Sayao (2007) foca no documento de especificação e não propõe a análise automatica ou semi-automática de RNFs. O trabalho também não contém estruturas de especificação baseadas em GORE, uma vez que analises das especificações são feitas a partir de cenários, léxicos, casos de uso ou estórias de usuários.

No trabalho de [Reynolds et al. 2006] são tratadas evidências de ocorrências em traços de execução na camada que faz a interação entre o software e demais aplicações (*middleware*) para que sejam detectados problemas de divergência entre comportamento planejado e realizado sobre desempenho de sistemas distribuídos. A

ferramenta proposta no trabalho permite que os desenvolvedores expressem, em uma linguagem declarativa, as expectativas sobre a estrutura do sistema de comunicações, tempo de execução e consumo de recursos. Ela inclui ainda sistema de instrumentação e ferramentas de anotação (registros de *logs*) para registrar o comportamento do sistema em execução, além de sistemas de visualização que permitem o desenvolvedor depurar rapidamente as inconsistências. No sentido de verificação de análise de desempenho, tratado como RNF, e o uso de registradores de traços de execução, a proposta se assemelha com a presente tese quando do tratamento de registro da execução do agente analisador. Outro ponto que se assemelha é com relação de elementos envolvidos na proposta do trabalho de Reynols et al. (2006), quando propõem que na arquitetura do sistema há elementos tais como arquivos com traços; expectativas dos envolvidos, quando sugerem que há um planejamento com relação ao requisito de desempenho, de certa forma tratam de uma base de conhecimento para tratamento de comparaçãoes entre esperado e realizado.

6.3.4
Proveniência

Miles et al. (2011) cita que a verificação estática de código fonte ou um *workflow* de trabalho é amplamente utilizado pela comunidade de linguagem de programação através de estruturas de análises de tipos, inferência e análise de modelos. Assim consegue-se estabelecer que o programa/fluxo de trabalho satisfaz algumas propriedades. Eles normalmente dependem da sintaxe e semântica da linguagem de programação que está sendo analisado.

Por outro lado, a verificação estática é complementada pela validação em tempo de execução, que é realizada quando o programa é executado e verifica-se os valores de dados gerados para satisfazer as restrições ou convergências com determinados parâmetros. Para esse ultimo caso o trabalho de Miles et al. (2011) propõe que a proveniência armazena partes de dados que indicam como um processo está sendo executado. O trabalho cita que aplicações chamadas de *provenance-aware* (de *awareness*) são aquelas que há a documentação ou registro de sua execução, de modo que a proveniência dos dados que eles produzem pode ser obtida e fundamentada. Os autores propõem no trabalho uma abordagem *provenance-based* para análise de *workflow*, *o*u seja, rastros de um *workflow* são armazeados para que se obtenha o conhecimento em cada ponto da execução do fluxo. Nesse ponto, há uma semelhança com a tese proposto, uma vez que é feita a utilização de

proveniência para se obter rastros da execução do agente para otimizar o processo de análise.

Em Miles et al. (2007) é proposto o uso de estratégia baseada em estruturas XML criadas a partir de orientação à metas de forma *ad-hoc*, ou seja, sem uso de modelos GORE formalizados. Abordam a questão do uso de agentes em SMA para a análise de tais modelos para responder perguntas a partir de rastros de proveniência deixados por determinado sistema. No trabalho utilizam algoritmos para responder a perguntas comuns sobre a responsabilidade e o sucesso de um processo e avaliam a abordagem com um exemplo simulado da area de saúde. Perguntas, geralmente utilizados no trabalho consistem em exemplos como: Quem foi o responsável para efeito *E*? Será que o efeito *E* corresponde ao que foi destinado a acontecer? Qual é a razão (ambos causal e intencional) do efeito *E*? Por causa da estrutura de dados e a semântica do que é registrado, uma consulta pode combinar a documentação para responder a perguntas importantes sobre se o aplicativo está cumprindo as metas de quem o utiliza.

O uso de SMA com proveniência também é citado no trabalho de [Jami 2011] onde os autores exploram a autonomia de agentes quando interagem entre si a partir de acompanhamento de traços de execução desses agentes. O estudo é feito a partr de uma aplicação colaborativa onde usuários estão compartilhando seus pensamentos a partir de lousa (*whiteboards*). Os autores também propõem o uso de desenvolvimento em JADE com base em FIPA para a plataforma dos agentes. Os agentes atuam na gravação de incidências o uso da lousa compondo assim uma base de proveniência. A intenção do trabalho é avaliar aspectos de colaboração dos usuários feitas por agentes autônomos a partir de registros feitos na lousa. Os agentes são programados para responder, a partir dos registros de proveniência, perguntas do tipo: Quem criou ou iniciou a lousa? Em qual momento foi criada/iniciada? Quem editou a lousa em determinado momento? Quais usuários comentaram em documento particular? Que conjunto de ações o usuário realiza nessa colaboração? Os agentes trabalham em uma estratégia *provenance-aware* para capturar a interação dos colaboradores.

6.4
Contribuições da Pesquisa

No início do trabalho foram levantadas algumas expectativas com relação à contribuições da pesquisa como por exemplo o problema da análise de RNF diante de

seu grau de subjetividade de julgamento. Nessa seção é feita uma análise sobre como tais contribuições foram satisfeitas.

- **Método sistêmico:** o método sistêmico propõe a especificação de RNF com base em ténicas já consolidadas, acrescenta uma instância para associar variáveis essenciais e padrões de conteúdos de artefatos de *software.* O trabalho também sistematiza um catálogo no formato XML para que possa ser utilizado por SMA para análise de conformidade de RNF implementado em *software*.

O método foi concebido em uma abordagem de análise de artefatos de *software* não apenas focados em diagramas i*, como sugere trabalhos de Baia et al (2013), Souza (2012), Wang et al. (2007) e Lingxiao et al. (2013), mas uma estrutura aberta, que abrange artefatos de *software*, tais como modelos em XML, código fonte e traços de execução, que a partir de uma definição de regras, pudessem ser comparados à uma estrutura pré-estabelecida, no caso um catálogo de RNF com um com metas a serem cumpridas, a fim de se obter de forma automática ou semi-automática a análise da satisfação de RNF .

- **Padrões Sugeridos:** o método apresenta a necessidade da criação de uma instância de representação de mais baixo nível, com valores absolutos verificáveis, associados às alternativas. Essa instância é passível julgamento dicotômico, do tipo atende/não atende, conforme/não conforme ou está implementado/não está implementado.

Tal proposta justifica-se uma vez que alternativas propostas nos *patterns* como por exemplo, “uso de cenários em requisitos”, “uso de banco de dados relacional”, “uso de algoritmo de ordenação” não são passíveis de análise, a não ser que se tenha definido sintática e semanticamente como são suas estruturas. Nesse ponto, a sugestão de variáveis essenciais, sinônimos ou padrões podem auxiliar a identificação dessas estruturas, o que foi sugerido nessa tese a partir da abordagem de Léxicos [Leite et al. 1997];

- **SMA:** a criação de um mecanismo baseado na arquitetura de agentes no modelo BDI são uma contribuição, uma vez que o uso desses agentes autônomos auxiliam nas decisões com relação às análises de elos de ligação nas propagações entre *sofgoals* e suas operacionalizações.

Além disso, o uso de agentes inteligentes baseados em heurísticas foi elaborado a partir dos registros do agente ANALISADOR em uma base de proveniência, ou seja, a partir da execução do agente suas ações de análise de conformidade ou não conformidade são gravadas com informações de qual meta foi cumprida e por qual variável essencial ou padrão foi satisfeita.

Os registros servem como base para novas análises e permitem a otimização do processo, já que traços de execução ou artefatos de outros *software* analisados, que por ventura tenham a mesma estrutura de outros já analisados, passam a ser considerados conforme análises anteriores.

- **Catálogo:** a tese propõe uma arquitetura única como um *metadado* registrado em catálogo XML de RNF fundamentado em abordagem de RNF *framework* e GQO para servir de regras para agentes autônomos na análise de RNFs.

- **Transparência do *Software*:** a partir do uso de modelos intencionais, seja por Paineis de Intencionalidades [Oliveira et al. 2008] ou por diagramas i* [Yu 1995], tornar explícita a intencionalidade dos agentes bem como sua arquitetura de funcionamento, em uma proposta de Transparência da Informação [Leite e Cappelli 2010] [CTS 2013], ou seja, utilizar das características de Transparência do CTS (2013) para tornar transparente a informação a respeito do *software* desenvolvido;

- **Bases de Conhecimento:** feito o uso de proveniência [Miles 2007] persistidas nas ações do agente ANALISADOR em estruturas XML para a criação de bases de conhecimento que auxiliam em dois pontos principais: primeiro deixar explícito o rastro entre o RNF, sua operacionalização e marcadores que o satisfazem; segundo, utilizar heurísticas formadas na base de conhecimento para otimização do processo de análise.

- **Aplicação do CTS:** pelo que se conhece da literatura, foi o primeiro trabalho que tratou de aplicar de forma prática o uso do CTS [CTS 2013] a partir da construção por reuso em diferentes RNFs. Isso fato viabilizado a partir de um catálogo XML com metadados que suportam a agregação das diferentes técnicas de especificação, RNF *framework* e GQO, reutilizado para modelar diferentes RNFs utilizados no Estudo de Casos.

6.5
Limitações

O problema de análise de software é um problema que desafia pesquisadores e vem sendo tratado na literatura há muito tempo. O espectro da verificação vai desde provas de programas contra especificações, com um sentido mais formal, até a inspeção visual de programas, num sentido menos formal. Assim como diferentes propostas de verificação, nossa proposta de análise possui limitações, mas que podem sugerir trabalhos de pesquisa futuros:

- O agente Consolidador, tal como foi estruturado, necessita de maior esforço de implementação para que possa tratar da análise de conteúdos a partir de sintaxe e

semânticas estabelecidas. Por exemplo, incorporar a esse agente outros planos mais especializados para o reconhecimento automático de padrões a partir da análise de linguagem natural;

- O método foi aplicado em estruturas de média complexidade com nós de relação aparentemente simples, por exemplo, foi feito o detalhamento do RNF considerando apenas dois graus de decomposição em *softgoals*.

Inclusive tal iniciativa necessitaria de uma avaliação sobre teoria de grafos para entender a complexidade de tais relações para analisar a complexidade dos algoritmos de analise de propagação, o que também não foi foco desse trabalho de tese.

O tipo de relação onde vários tipos de *softgoals* tiverem que ser satisfeitos poderá influenciar no grau de conformidade, principalmente, havendo algum nível antagônico entre os *softgoals*. Tal fato não foi estudado nessa tese;

- Uma avaliação ainda não estudada foi a escalabilidade da arquitetura proposta no catálogo XML de RNF. Foram utilizados arquivos XML para compor o catálogo com a arquitetura dos padrões, o que compromete uma avaliação de modelos com maior nível de detalhamento das decomposições dos RNFs. Seria importante utilizar bancos de dados que suportam XML nativo, como por exemplo: Berkley DB XML, eXist, Tamino, Timber, Xindice [Bourret, R. 2010] para testar modelos mais complexos de decomposição e relacionamentos de RNFs;

- O método não foi aplicado em escala corporativa para determinar facilidade de uso ou aplicação no ambiente empresarial, portanto, há uma necessidade desse tipo de estudo para avaliar a sua real contribuição para a indústria;

- Os estudos de caso não contemplaram realidades de aplicações orientadas a aspectos, ou seja, como *software* desenvolvidos a partir do paradigma podem ser analisados para verificar sua conformidade com catálogos pré-estabelecidos de RNFs.

Como em todo trabalho existem limitações a serem tratadas, mas o desenvolvimento do método permitiu a aquisição do conhecimento e uma visibilidade de sua aplicação.

6.6
Trabalhos Futuros

Os desafios de análises de RNF são muitos e sinalizam para o estudo contínuo e sistemático. Os estudos realizados nesse trabalho de tese vislumbram alguns trabalhos que podem ser iniciados e evoluídos futuramente:

- Tratar a análise de RNFs a partir da associação de dois ou mais catálogos. Por exemplo, como encontrado nos estudos de Cappelli et al. (2010) com a definição

da catálogos antagônicos entre Transparência e Segurança. Ou até mesmo para confirmar relações de potencialização entre RNF do tipo Transparência e Confiança, Proveniência e Confiança. Estruturar o método sistêmico para que incorpore dois catálogos de RNF para que possam ser analisados e confirmados os graus de impacto positivos e propostas minimizações dos impactos negativos;

- O foco inicial foi a análise de conformidades dos registros em relação aos *patterns* de RNF. Os registros de não conformidades estão apenas registrados no sistema e poderão ser explorados em trabalhos futuros com a proposta do uso de sistemas de recomendação [Adomavicius e Tuzhilin 2005] [Burke 2002]. A intenção é criar agentes autônomos inteligentes que possam sugerir alternativas a partir do catalogo de RNF para que as não conformidades encontradas sejam mitigadas. Para isso as estratégias de proveniência e bases de conhecimento devem ser reestruturadas para agentes possam analisar ações pré-estabelecidas e ações executadas para que se minimize o desvio entre planejado e realizado. Tal proposta se assemelha aos trabalhos de Lingxiao et al. (2013) e Souza (2012);

- Extender o catálogo XML para que incorpore estruturas que permitam análise sintática e semântica baseadas em ontologias sem que percam seu grau de valor absoluto que lhes permitem ser analisáveis. Isso poderia minimizar ou extinguir a necessidade de interação humana em analises feitas, por exemplo, a partir das atuais variáveis essenciais;

- Aplicar o método sistêmico de análise de RNF para criar catálogos que possam ser aderentes a modelos de maturidade ou ciclos de vida de software como CMMI[17], MPS.BR[18] ou ISO 12207[19]. As estruturas criadas poderiam contemplar catálogos que atendessem a processos de gestão de requisitos ou gestão da configuração, para que o SMA analisasse se artefatos gerados pelos processos estão condizentes com normas estabelecidas e atendem as exigências dos modelos de maturidade;

- Na área de agentes, um campo de pesquisa tem sido a associação de normas em SMA com o objetivo de regular o comportamento de agentes sem que os mesmos percam sua autonomia. As normas podem ser definidas por restrições, responsabilidades ou padrões de comportamento para atingir determinado objetivo, a fim de que sejam satisfeitos ou mesmo evitados [Camino et al. 2009] [Neto et al. 2011]. O catálogo XML de RNF permite a inserção de normas para que regras de

[17] http://www.sei.cmu.edu/cmmi/
[18] http://www.softex.br/mpsbr/
[19] http://www.iso.org/iso/catalogue_detail?csnumber=43447

análise sejam postas a fim de que os agentes operem baseado em contexto pré-estabelecido a partir de uma abordagem NBDI [Neto et al. 2011].

7.
Referências

[Adomavicius e Tuzhilin 2005] ADOMAVICIUS, G.; TUZHILIN, A.. **Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions**. IEEE Transactions on Knowledge and Data Engineering, Vol. 17(6), 2005, pp. 734-749.

[Almentero e Cunha 2010] ALMENTERO, E. K.; CUNHA, H. de S.. IstarJADE. **IStarJADE - Infraestrutura para implementação de agentes a partir de modelos i***. Relatório Técnico. Pontifícia Universidade Católica do Rio de Janeiro, 2010. <Disponível em: http://code.google.com/p/istarjade/source/browse/#svn%2Fsrc%2Fistar> <Acessado em: 20/01/2014>.

[Anton 1997] ANTÓN, A. I.. **Goal Identification and Refinement in the Specification of Software-Based Information Systems**, Ph.D. Thesis, Georgia Institute of Technology, Atlanta, GA, June 1997.

[Baia et al. 2012] BAIA, J. W.; BRAGA, J. L.; CARVALHO, L. F. **Verificação de Requisitos de Transparência em Modelos iStar**. In: XV WER - Workshop em Engenharia de Requisitos (XIV CibSE - Congresso Ibero-Americano em Engenharia de Software), 2012, Bueno Aires . Proceedings XV CibSE. Buenos Aires, 2012. v.XV.<Disponível em: http://wer.inf.puc-rio.br/WERpapers/papers_by_conference.lp?conference=WER12> <Acessado em: 20/01/2014>.

[Baia et al 2013] BAIA, J. W., BRAGA, J. L.. **Uso de sinônimos na identificação de atributos de transparência**. In: 16th WER - Workshop em Engenharia de Requisitos (15th CibSE - Congresso IberoAmericano em Engenharia de Software), pp.94-104, Montevideo (2013).

[Balanyi e Ferenc 2003] Balanyi Z.; Ferenc, R.. **Mining Design Patterns from C++ Source Code**. 19th IEEE International Conference on Software Maintenance, 2003.

[Bansiya 1998] BANSIYA, J.. **Automating Design-Pattern Identification**. Dr. Dobb's Journal, vol. 23, no. 6, June, 1998.

[Basili 1992] BASILI, V. R.. **Software Modeling and Measurement: The Goal Question Metric Paradigm**. Computer Science Technical Report Series, CS-TR-2956 (UMIACS-TR-92-96), University of Maryland, College Park, MD, September 1992.

[Bourret, R., 2010] **XML Database Products**. Copyright 2000-2010 by Ronald Bourret Last updated on: June 20, 2010. <Disponível em: http://www.rpbourret.com/xml/XMLDatabaseProds.htm#native> <Acessado em: 23/05/2014>.

[Bratman 1999] BRATMAN, M. E. **Intention, Plans, and Practical Reason**. University of Chicago, ISBN: 1575861925, 208 pages, 1999.

[Braubach et al. 2003] BRAUBACH, L.; LAMERSDORF, W.; POKAHR, A.. **Jadex: Implementing a BDI Infrastructure for JADE Agents**. Distributed Systems and Information Systems, vol. 3, n. 3, pp.76-85, September 2003.

[Bresciani 2004] BRESCIANI, P.; GIORGINI, P.; GIUNCHIGLIA, F.; MYLOPOULOS, J.; PERINI, A.. **TROPOS: An Agent-Oriented Software Development Methodology**. In: Journal of Autonomous Agents and Multi-Agent Systems. May 2004. Kluwer Academic Publishers.

[Buchli 2003] BUCHLI, F.. **Detecting Software Patterns using Formal Concept Analysis**, Diploma Thesis, University of Bern, 2003.

[Burke 2002] BURKE, R.. **Hybrid Recommender Systems: Survey and Experiments**. User Modeling and User-Adapted Interaction, Vol. 12(4), 2002, pp. 331-370.

[CLIPS 1991] CLIPS Version 5.1 **CLIPS User's Guide**, NASA Lyndon B. Johnson Space Center, Software Technology Branch, Houston, TX, 1991.

[Camino et al. 2009] CAMINO, A. G.; **Normative regulation of open multi-agent systems**, Ph.D. dissertation, Artificial Intelligence Research Institute (IIIA), Spain, 2009.

[Cappelli 2009] CAPPELLI, C.. **Uma Abordagem para Transparência em Processos Organizacionais Utilizando Aspectos**. Rio de Janeiro. 328 p. Tese de Doutorado – Departamento de Informática, PUC-Rio, 2009.

[Cappelli et al. 2010] CAPPELLI, C. ; CUNHA, H. S.; GONZÁLEZ-BAIXAULI, B; LEITE, J. C. S. P.. **Transparency versus security: early analysis of antagonistic requirements**, In: Symposium on Applied Computing, Sierre, Switzerland. vol. XXV. pp. 298-305, 2010.

[Cares et al. 2007] CARES, C.; FRANCH, X.; PERINI, A.; SUSI, A.. **iStarML: The i* Mark-up Language: References Guide**. Barcelona, Spain, August 2007.

[Castro 2002] CASTRO, J.; KOLP, M.; MYLOPOULOS, J.. **Towards Requirements-Driven Information Systems Engineering: The Tropos Project**. In: The 13th international conference on advanced information systems engineering, Oxford: Elsevier Science Ltd, v.27, n.6., 2002, p. 365-389.

[Chung et al. 2000] CHUNG, L.; NIXON, B. A.; YU, E., and MYLOPOULOS, J. **Non-Functional Requirements in Software Engineering**. Springer, 2000.

[CTS 2013] CTS. **Catálogo de Transparência de Software**. 2013. <Disponível em: http://transparencia.inf.puc-rio.br/wiki/index.php/Cat%C3%A1logo_Transpar%C3%AAncia> <Acessado em: 17/10/2013>.

[Cunha 2007] CUNHA, H.S.. **Uso de estratégias orientadas a metas para modelagem de requisitos de segurança**. Rio de Janeiro, 2007. 145p. Dissertação de Mestrado - Departamento de Informática, PUC-Rio.

[Cunha et al 2013] CUNHA, H.; LEITE, J. C. S. P; DUBOC, L.; WERNECK, V.. **The challenges of representing transparency as patterns**. In.: Requirements Patterns (RePa), 2013 IEEE Third International Workshop on (pp. 25-30). IEEE, July, 2013.

[Cysneiros et al. 2003] CYSNEIROS, L. M.; YU, E.; LEITE, J.C.S.P.. **Cataloguing Non-Functional Requirements as Networks.** In.: Proceedings of Requirements Engineering for Adaptable Architectures @11th International Requirements Engineering Conference, 2003 p.13-20.

[Cysneiros e Leite 2004] CYSNEIROS, L. M.; Leite, J.C.S.P.. **Non-Functional Requirements: From Elicitation to Conceptual Model**, IEEE Trans. Software Engineering 30(5). pp. 328-350, May 2004.

[Dardenne et al. 1993] DARDENNE, A.; van LAMSWEERDE A.; FICKAS, S.. **Goal-Directed Requirements Acquisition**, Science of Computer Programming, Vol. 20, 1993, 3-50.

[DeMarco 1978] DEMARCO. T., **Structured Analysis and System Specification**. Yourdon Press. 1978.

[Felicissimo et al. 2004] FELICISSIMO, C. H. ; LEITE, J. C. S. P. ; BREITMAN, K. K. ; FERNANDES, S. L. . C&L: **Um Ambiente para Edição e Visualização de Cenários e Léxicos**. In: XVIIII Simpósio Brasileiro de Engenharia de Software (SBES), 2004, Brasilia. Anais do XVIIII Simpósio Brasileiro de Engenharia de Software (SBES), 2004. p. 43-48.

[Fenton e Pfleeger 1997] FENTON, N. E.; PFLEEGER, S.L.. **Software Metrics: a Rigorous and Practical Approach**, 2nd Ed., PWS Publishing Company. 1997.

[Fidge e Lister 1993] FIDGE, C.; LISTER, A.. **The Challenges of Non-Functional Computing Requirements**. Seventh Australian Software Engineering Conference (ASWEC93), 1993.

[Glinz 2007] GLINZ, M. **On Non-Functional Requirements**. In 15th IEEE International Volume , Issue , 15-19 Oct, pages 21–26, 2007.

[Gonzales-Baixauli et al. 2005] Gonzalez-Baixauli, B.; Laguna, M.; Leite, J.C.S. do P..Prado Leite. **Aplicación de la Teoría de Constructos Personales a la Elicitación de Requisitos**. IEEE Latin America Transactions. Vol. 3, Issue 1. March, 2005.

[Google Scholar 2013] **Google Scholar**. <Disponível em: www.scholar.google.com > <Acessado em: 12/12/2013>

[Grupo ER PUC-Rio 2013]. **Grupo de Engenharia de Requisitos da PUC do Rio de Janeiro**. 2013. <Disponível em: http://transparencia.inf.puc-rio.br/wiki/index.php/P%C3%A1gina_principal> <Acessado em: 17/10/2013>.

[IEEE Std-828 1998] ANSI/IEEE Std-828. **IEEE Standard for Software Configuration Management Plans**, 1998. <Disponível em: http://standards.ieee.org/reading/ieee/std_public/description/se/828-1990_desc.html> <Acessado em: 20/01/2014>

[ISO 10007] International Organization for Standardization (ISO), **Quality Management Systems – Guidelines for Configuration Management**, ISO 10007:2003(E), Geneva, Switzerland, 2003.

[ISO 12207] ISO/IEC 12207 - Information technology - **Software life cycle processes**. International Organization for Standardization. ISO, 1995b.

[Jami 2011] JAMI, S. I.; SHAIKH, Z. A.. **An Autonomous Provenance Tracking System for Collaborative Environment**. Australian Journal of Basic and Applied Sciences, 5(6): 1668-1674, 2011.

[Jennings 2002] JENNINGS, N. R.; WOOLDRIDGE, M.. **Agent-Oriented Software Engineering in Handbook of Agent Technology**. (ed. J. Bradshaw) AAAI/MIT Press. 2002.

[Kaiya e Kaijiri 1999] KAIYA, H.; KAIJIRI, K... **Refining Behavioral Specification for Satisfying Non-functional Requirements of Stakeholders**. In IEICE transactions on information and systems Vol.E85-D, No.4(20020401), pages 623–636, 1999.

[Lattesscholar 2013] **Lattesscholar: Requirements Engineering Group at PUC-Rio**. <Disponível em: http://www.er.les.inf.puc-rio.br/~wiki/index.php/Lattesscholar> <Acessado em: 12/12/2013>

[Lattes 2013] Lattes. **Plataforma Lattes: Currículo Lattes.** <Disponível em: lattes.cnpq.br > <Acessado em: 12/12/2013>

[Lamsweerde e Letier 2000] van Lamsweerde , A.; Letier, E**.. Handling Obstacles in Goal-Oriented Requirements Engineering**. IEEE Trans. Software Eng., vol. 26, pp. 978-1005, 2000.

[Lamsweerde 2001] van LAMSWEERDE, A.. **Goal-Oriented Requirements Engineering: A Guided Tour**. 5th IEEE International Symposium on RE'01, pp. 249-262, August 2001.

[Lapouchnian 2005] LAPOUCHNIAN, A.. **Goal-Oriented Requirements Engineering: An Overview of the Current Research**. Technical report, Universidade de Toronto, Canadá, 2005.

[Leite et al. 1997] LEITE, J. C. S. P.; ROSSI, G.; MAIORANA, V.; BALAGUER, F.; KAPLAN, G.; HADAD, G.; OLIVEROS, A.. **Enhancing a requirements baseline with**

scenarios. In: IEEE INTERNATIONAL SYMPOSIUM ON REQUIREMENTS ENGINEERING – RE97, 3rd, 1997, Annapolis, MD. Proceedings. IEEE Computer Society Press, 1997. p. 44-53.

[Leite et al. 2000] LEITE, J.C.S.P.; HADAD, G.; DOORN, J.. KAPLAN, G**.. A Scenario Construction Process**. In.: Requirements Engineering Journal. Springer-Verlag London Limited: 2000, Vol. 5, n.1, Pags. 38-61.

[Leite et al. 2005] LEITE, J. C. S. P. ; Yu, Y ; LIU, L. ; YU, E. ; Mylopoulos, J **. Quality-Based Software Reuse**. In: Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005, 2005, Porto, Portugal. Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005, Porto, Portugal, June 13-17, 2005. Proceedings, 2005. v. 17. p. 535-545.

[Leite 2006] LEITE, J. C. S. P.. **Sistemas de Software Transparentes**. In: Simpósio Brasileiro de Engenharia de Software, 2006, Florianópolis. Anais do XX Simpósio Brasileiro de Engenharia de Software. Porto Alegre : Sociedade Brasileira de Computação, 2006. v. XX. p. 319-319.

[Leite e Cappelli 2010] LEITE, J.C.S.P.; CAPPELLI, C.. **Software Transparency. Business & Information Systems Engineering**, Springer, 2010, pp 127-139.

[Leal et al. 2013a] LEAL, A. L. C.; SOUSA, H. P; LEITE, J. C. S. P; LUCENA, J. P.. **Aplicação de modelos intencionais em sistemas multiagentes para estabelecer políticas de monitoração de transparência de software**. Revista de Informática Teórica e Aplicada, vol. 20, no. 2, 2013, pp. 111-138.

[Leal et al. 2013b] LEAL, A. L. C.; Sousa, H. P.; LEITE, J. C. S. P.. **Desafios de monitoração de requisitos não funcionais: avaliação em Transparência de Software**. In: Requirements Engineering@Brazil 2013, 2013, Rio de Janeiro. CEUR Workshop Proceedings, 2013. v. 1005. p. 25-30.

[Leon 2000] LEON, A.. **A Guide to Software Configuration Management**. Norwood, MA, Artech House Publishers, 2000.

[Lingxiao et al. 2013] LINGXIAO, F.; XIN, P.; YIJUN, YU; WENYUN, Z.. **Stateful Requirements Monitoring for Self-Repairing of Software Systems**. Technical report, # FDSE-TR201101, Fudan University, China, 2013.

[Matoussi e Laleau 2008] MATOUSSI, A.; LALEAU, R.. **A survey of Non-Functional Requirements in Software Development Process**, Universita Paris, Faculté des Science et Technologie, Paris 2008

[MASearch 2013] **Microsoft Academic Search**. <Disponível em: http://academic.research.microsoft.com> <Acessado em: 12/12/2013>

[Miles et al. 2005] MILES, S.; GROTH, P.; BRANCO, M.; MOREAU, L.. **The requirements of recording and using provenance in e-Science experiments**. In.: Technical Report: Electronics and Computer Science, University of Southampton. 2005.

[Miles et al. 2007] MILES, S.; MUNROE, S.; LUCK, M.; MOREAU, L.. **Modelling the provenance of data in autonomous systems**. In.: Proceedings of Autonomous Agents and Multi-Agent Systems 2007, pp. 243–250, Honolulu, Hawai'i, May.

[Miles et al. 2011] MILES, S.; WONG, S. C.; FANG, W.; GROTH, P.; ZAUNER, K. P.; MOREAU, L.. **Provenance-based validation of e-science experiments**. Web Semantics: Science, Services and Agents on the World Wide Web, 5(1), 2011.

[Mylopoulos 2008] MYLOPOULOS, J.. **Goal-Oriented Requirements Engineering**. XI Conferencia Iberoamericana de Software Engineering, pp. 13-17, Brazil, February 2008.

[Mylopoulos 1992] MYLOPOULOS, J.; CHUNG, L.; NIXON, B.. **Representing and Using Non-Functional Requirements: A Process-Oriented Approach**. IEEE Transactions on Software Engineering, 18(6), June 1992.

[Neto et al. 2011] NETO, B. F. dos S.; da SILVA, V. T.; de LUCENA, C. J. P.. **NBDI: An architecture for goal-oriented normative agents**. In.: ICAART 2011 - Proceedings of the 3rd International Conference on Agents and Artificial Intelligence, Volume 1 – Artificial Intelligence, Rome, Italy, January 28-30, 2011, pp. 116–125.

[Oliveira et al. 2007] OLIVEIRA, A. P. A.; LEITE, J. C. S. P.; CYSNEIROS, L. M.; CAPPELLI, C.. **Eliciting Multi-Agent Systems Intentionality: from Language Extended Lexicon to i* Models**. In: Jornadas Chilenas de Computação 2007 - UNAP, 2007, In.: Proceedings of the XXVI International Conference of the Chilean Computer Science Society. Iquique: Los Alamitos: IEEE Computer Society Press, 2007, v. 16. p. 40-49.

[Oliveira et al. 2008] OLIVEIRA, A.P.A.; LEITE, J.C.S.P.; CYSNEIROS, L. M.. **Engenharia de Requisitos Intencional: Um Método de Elicitação, Modelagem e Análise de Requisitos**. Rio de Janeiro, 2008. 261 p. Tese de Doutorado - Departamento de Informática, PUC-Rio.

[Oliveira et al. 2013] OLIVEIRA, A. P. A.; Braga, J. L.; LANA, C. A; CUNHA, L. G.. **Utilizando sistemas de conhecimento para a identificação de presença de metas flexíveis em uma linguagem de domínio**. In: ER@BR2013 - Requirements Engineering@Brazil 2013, 2013, Rio de Janeiro. Proceedings of Requirements Engineering Brazil 2013. Rio de Janeiro, RJ: CEUR-WS.org, 2013. v. 1005

[Oxford 2013] Oxford English Dictionary, "**provenance**, **n**". Oxford University Press.

[Pinheiro et al. 2003] PINHEIRO da S.; MCGUINNESS, P.; MCCOOL, D.. **Knowledge Provenance Infrastructure**. IEEE Data Engineering Bulletin 26(4), 2003, pp. 26-32.

[Publish and Perish 2013] **Publish and Perish**. <Disponível em: http://www.harzing.com/pop.htm> <Acessado em: 12/12/2013>

[Rao 1996] RAO, A. S.. **AgentSpeak: BDI agents speak out in a logical computable language**, in 'MAAMAW '96: Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world: agents breaking away', Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996, pp. 42–55.

[Reynolds et al. 2006] Reynolds, P.; Killian, C.; Wiener, J. L.; Mogul, J. C.; Shah, M. A.; Vahdat, A.. **Pip: Detecting the Unexpected in Distributed Systems**. In.: Proceedings of 3rd Symposium on Networked Systems Design and Implementation (NSDI), San Jose, CA, May, 2006.

[Ross 1977] ROSS, D.. **Structured Analysis (AS): A Language for Communicating Ideas (SADT)**. IEEE Transactions on Software Engineering, vol. 3, no.1, 1977, pp. 16-34.

[Rumbaugh et al. 1991] RUMBAUGH, J.; BLAHA, M.; PREMERLANI, W.; EDDY, F.; LORENSEN, W.. **Object-Oriented Modeling and Design**. Prentice Hall. 1991.

[ScholarCitation 2013] **Google Scholar Citation** <Disponível em: http://scholar.google.com/intl/en/scholar/citations.html> <Acessado em: 12/12/2013>

[Serrano e Leite 2011a] SERRANO, M.; LEITE; J.C.S.P.. **Capturing transparency-related requirements patterns through argumentation**. In: First International Workshop on Requirements Patterns (RePa), pp.32-41, 29 Aug. 2011.

[Serrano e Leite 20011b] SERRANO, M.; LEITE, J. C. S. P.. **Development of Agent-Driven Systems: from i* Architectural Models to Intentional Agents Code**. In: Fifth International istar Meeting, 2011, Itália. Fifth International istar Meeting.

[Serrano 2011] SERRANO, M.; LEITE, J. C. S. P.. **Transparent Software Development Based on Intentional Argumentation**. Rio de Janeiro, 2011. 282p. DSc Thesis - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

[Simon 1996] SIMON H. A.. **The Sciences of the Artificial**. MIT Press, Cambridge, MA, USA, 3rd ed., 1996.

[Sommerville 2011] Sommerville, I.. **Engenharia de Software**, 9ª Edição. Pearson Education, 2011.

[Souza 2012] SOUZA, V. E. S.. **Requirements-based Software System Adaptation**. Phd thesis, University of Trento, Italy 2012.

[Supakkul et al. 2010] SUPAKKUL, S.; HILL, T; CHUNG, L.; THAN TUN, T.; LEITE, J.C.S.P.. **An NFR Pattern Approach to Dealing with NFRs**. In: 18th IEEE International Requirements Engineering Conference, 2010, Sydney. Procceedings of the 18th IEEE International Requirements Engineering Conference. los alamitos : ieee computer society press, 2010. v. 18. p. 179-188.

[Tsantalis et al. 2006] TSANTALIS, N.; CHATZIGEORGIOU, A.; STEPHANIDES, G.; HALKIDIS, S. T.. **Design Pattern Detection Using Similarity Scoring**. IEEE Transactions on Software Engineering, November 2006.

[Vasques et al. 2005] VASQUEZ I.; GOMADAM K.; PATTERSON S.. **Framework for representing provenance for web services and processes**. Technical Report, LSDIS Lab, 2005.

[Xavier et al. 2010] XAVIER, L.; ALENCAR, F. M.; CASTRO, J.; PIMENTEL, J.. **Integração de Requisitos Não-Funcionais a Processos de Negócio: Integrando BPMN and NFR**. In.: Workshop de Engenharia de Requisitos, WER, 2010.

[Yu 1995] YU, E.S.K.. **Modelling Strategic Relationships For Process Reengineering**. Ph.D. dissertation. Dept. of Computer Science, University of Toronto, 1995.

[Yu et al. 2004] YU Y., LEITE, J. C. S. P.; MYLOPOULOS J.. **From goals to aspects: discovering aspects from requirements goal models**, Proceedings of IEEE International Symposium on Requirements Engineering (RE'04), Japan, 2004, pp. 38-47.

[Yu et al. 2008] YU, E.; CASTRO, J., PERINI, A.. **Strategic Actors Modeling with i***, Tutorial Notes, 16th Intl. Conf. on Requirements Engineering, IEEE Computer Society, Spain, pp.01-60, 2008.

[Wooldridge 2002] WOOLDRIDGE, M.. **An Introduction to Multi-Agent Systems**. John Wiley and Sons. 2002.

APÊNDICE A - Estrutura XML do cátalogo de RNF

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalogo tipo="" objetivo="">
  <versao>1</versao>
        <patternObjetivos>
                <patternIdentificacao></patternIdentificacao>

                <patternDecomposicaoNivel_1>
                        <idDecomposicaoNivel_1></idDecomposicaoNivel_1>
                        <patternSelecao></patternSelecao>

                                <patternDecomposicaoNivel_2>
                                        <idDecomposicaoNivel_2></idDecomposicaoNivel_2>
                                        <patternSelecao></patternSelecao>
                                </patternDecomposicaoNivel_2>

                                <patternDecomposicaoNivel_2>
                                        <idDecomposicaoNivel_2></idDecomposicaoNivel_2>
                                        <patternSelecao></patternSelecao>
                                </patternDecomposicaoNivel_2>

                                <patternDecomposicaoNivel_2>
                                        <idDecomposicaoNivel_2></idDecomposicaoNivel_2>
                                        <patternSelecao></patternSelecao>
                                </patternDecomposicaoNivel_2>

                                <patternDecomposicaoNivel_2>
                                        <idDecomposicaoNivel_2></idDecomposicaoNivel_2>
                                        <patternSelecao></patternSelecao>

                                        <patternGrupo>
                                                <idGrupo></idGrupo>
                                                <tituloGrupo></tituloGrupo>
```

```
<patternQuestoes>
        <idQuestao></idQuestao>
        <tituloQuestao></tituloQuestao>
                <patternAlternativas>
                        <idAlternativa></idAlternativa>
                        <tituloAlternativa></tituloAlternativa>
                        <patternSelecao></patternSelecao>
                        <variaveisEssenciais>

                                <idVariaveisEssenciais></idVariaveisEssenciais>
                                <nocaoVariaveisEssenciais>        <nocaoVariaveisEssenciais>
                                <sinonimosVariaveisEssenciais>
                                        <idSinonimosVariaveisEssenciais></idSinonimosVariaveisEssenciais>
                                        <idSinonimosVariaveisEssenciais></idSinonimosVariaveisEssenciais>
                                </sinonimosVariaveisEssenciais>
                                <padrao>
                                        <idPadrao1></idPadrao1>
                                        <idPadrao2></idPadrao2>
                                </padrao>

                        </variaveisEssenciais>

                </patternAlternativas>

                ...
</patternQuestoes>

<patternQuestoes>
        <idQuestao></idQuestao>
        <tituloQuestao></tituloQuestao>
                <patternAlternativas>
                        <idAlternativa></idAlternativa>
                        <tituloAlternativa></tituloAlternativa>
                        <patternSelecao></patternSelecao>
```

```
                                                    <variaveisEssenciais>
                                                        <idVariaveisEssenciais></idVariaveisEssenciais>
                                                        <nocaoVariaveisEssenciais>          <nocaoVariaveisEssenciais>
                                                        <sinonimosVariaveisEssenciais>
                                                            <idSinonimosVariaveisEssenciais></idSinonimosVariaveisEssenciais>
                                                        </sinonimosVariaveisEssenciais>
                                                        <padrao>
                                                            <idPadrao1></idPadrao1>
                                                            <idPadrao2></idPadrao2>
                                                        </padrao>

                                                    </variaveisEssenciais>

                                                    ...

                                                </patternAlternativas>
                                            </patternQuestoes>

                                        </patternGrupo>
                                    </patternDecomposicaoNivel_2>

                                    <patternDecomposicaoNivel_2>
                                        <idDecomposicaoNivel_2></idDecomposicaoNivel_2>
                                        <patternSelecao></patternSelecao>
                                    </patternDecomposicaoNivel_2>

                                </patternDecomposicaoNivel_1>
                            </patternObjetivos>
                        </catalogo>
```

APÊNDICE B – Código Fonte do SMA

B.1 - Esta seção apresenta o código-fonte da implementação em XML das características BDI dos agentes.

```
<!--
        <H3>The Analyzer agent.</H3>

        This agent analyze the log for evaluate the RNF
-->

<agent xmlns="http://jadex.sourceforge.net/jadex"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                    http://jadex.sourceforge.net/jadex-0.96.xsd"
        name="AnalyzerAgent"
        package="jadex.examples.TMMAS.agent.analyzerAgent">

        <imports>
                <import>jadex.adapter.fipa.SFipa</import>
                <import>java.io.File</import>
                <import>handleClasses.CanonicalDefinition</import>
                <import>org.w3c.dom.Document</import>
        </imports>

        <beliefs>
                <belief name="stdPath" class="String">
                        <fact>"C:/Policy.xml"</fact>
                </belief>
                <belief name="policy" class="Document">
                        <fact>null</fact>
                </belief>
        </beliefs>

        <plans>
                <!-- Plan for analyze the log. -->
                <plan name="receiveLog">
                        <body class="ReceiveLogPlan"/>
                </plan>

                <!-- Plan for evaluate the RNF according to the operacionalizations. -->
                <plan name="evaluateRNFLog">
                        <body class="EvaluateRNFPlan"/>
                        <trigger>
                                <internalevent ref="call_EvaluateRNFPlan"/>
                        </trigger>
                </plan>

                 <!-- Plan for provenance registration. -->
                 <plan name="provenanceRegister">
                         <body class="provenanceRegisterPlan"/>
```

```
                    <trigger>
                        <internalevent ref="call_provenanceRegisterPlan"/>
                    </trigger>
                </plan>

                <!-- Plan for canonize the different log patterns. -->
                <plan name="SendRNFEvaluation">
                    <body class="SendRNFEvaluationPlan"/>
                    <trigger>
                        <internalevent ref="call_SendRNFEvaluationPlan"/>
                    </trigger>
                </plan>

                <!-- Plan to save the Unknown Elements. -->
                <plan name="receiveExecTraces">
                    <parameter name="execTraces" class="Document">
                    <messageeventmapping ref="receiveExecTracesMSG.content"/>
                    </parameter>
                    <body class="EvaluateRNFPlan"/>
                    <trigger>
                        <messageevent ref="receiveExecTracesMSG"/>
                    </trigger>
                </plan>
            </plans>

            <events>
                <messageevent name="receiveCanonicLogMSG" type="fipa"
direction="receive">
                    <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.INFORM</value>
                    </parameter>
                    <parameter name="content-class" class="Class" direction="fixed">
                        <value>CanonicalDefinition.class</value>
                    </parameter>
                </messageevent>

                <messageevent name="receiveExecTracesMSG" direction="receive"
type="fipa">
                    <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REQUEST</value>
                    </parameter>
                    <parameter name="content-class" class="Class" direction="fixed">
                        <value>Document.class</value>
                    </parameter>
                </messageevent>

                <!-- This internal event means the call for the Send RNF Evaluation Plan.-->
                <internalevent name="call_SendRNFEvaluationPlan"/>

                <!-- This internal event means the call for the Evaluate RNF Plan.-->
                <internalevent name="call_EvaluateRNFPlan"/>
```

```
		<!-- This message event sends the Canonic Model register to the Consolidator
Agent agent.-->
		<messageevent		name="sendCanonicModelMSG"		type="fipa"
direction="send">
			<parameter name="performative" class="String" direction="fixed">
				<value>SFipa.INFORM</value>
			</parameter>
			<parameter name="language" class="String" direction="fixed">
				<value>SFipa.NUGGETS_XML</value>
			</parameter>
		</messageevent>

		<!-- This message event sends the RNF Evaluation register to the Consolidator
Agent agent.-->
		<messageevent		name="sendRastEvaluationMSG"		type="fipa"
direction="send">
			<parameter name="performative" class="String" direction="fixed">
				<value>SFipa.INFORM</value>
			</parameter>
			<parameter name="language" class="String" direction="fixed">
				<value>SFipa.NUGGETS_XML</value>
			</parameter>
		</messageevent>
	</events>

</agent>

------------
<!--
	<H3>The Canonizer agent.</H3>

	This agent canonizer the unknown structures
-->

<agent xmlns="http://jadex.sourceforge.net/jadex"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://jadex.sourceforge.net/jadex
		http://jadex.sourceforge.net/jadex-0.96.xsd"
	name="CanonizerAgent"
	package="jadex.examples.TMMAS.agent.canonizerAgent">

	<imports>
		<import>jadex.adapter.fipa.SFipa</import>
		<import>java.io.File</import>
		<import>java.util.List</import>
		<import>org.w3c.dom.Document</import>

	<import>com.thoughtworks.xstream.persistence.PersistenceStrategy</import>

	<import>com.thoughtworks.xstream.persistence.FilePersistenceStrategy</import>
	</imports>
```

```
        <beliefs>
                <belief name="unkExecTracesFiles" class="PersistenceStrategy">
                        <fact>new                                   FilePersistenceStrategy(new
File("C:\\USBaseList.xml"))</fact>
                </belief>
        </beliefs>

        <plans>
                <!-- Plan for receive the log and register it in the belief base. -->
                <plan name="canonizeExecTraces">
                        <body class="CanonizeExecTracesPlan"/>
                </plan>

                <!-- Plan to save the Unknown Elements. -->
                <plan name="receiveExecTraces">
                        <parameter name="execTraces" class="Document">
                        <messageeventmapping ref="receiveExecTracesMSG.content"/>
                        </parameter>
                        <body class="ReceiveUnkStructuresPlan"/>
                        <trigger>
                                <messageevent ref="receiveExecTracesMSG"/>
                        </trigger>
                </plan>

                <!-- Plan to send the canonized log plan to the Analyzer agent. -->
                <plan name="sendLog">
                        <body class="SendCanonizedLog"/>
                        <trigger>
                                <internalevent ref="call_SendCanonizedLogPlan"/>
                        </trigger>
                </plan>
        </plans>

        <events>
                <messageevent        name="receiveExecTracesMSG"        direction="receive"
type="fipa">
                        <parameter name="performative" class="String" direction="fixed">
                                <value>SFipa.REQUEST</value>
                        </parameter>
                        <parameter name="content-class" class="Class" direction="fixed">
                                <value>Document.class</value>
                        </parameter>
                </messageevent>

                <!-- This internal event means the call for the Canonize Plan.-->
                <internalevent name="call_CanonizeExecTracesPlan"/>

                <!-- This internal event means the call for the Send Canonized Log Plan.-->
                <internalevent name="call_SendCanonizedLogPlan"/>

                <!-- This message event sends the canonic log to the Analyzer agent.-->
```

```
		<messageevent name="sendCanonicExecTracesMSG" type="fipa"
direction="send">
			<parameter name="performative" class="String" direction="fixed">
				<value>SFipa.INFORM</value>
			</parameter>
			<parameter name="language" class="String" direction="fixed">
				<value>SFipa.NUGGETS_XML</value>
			</parameter>
		</messageevent>
	</events>

	<configurations>
		<configuration name="default">
			<plans>
				<!-- Receive the log and register it in the belief base. -->
				<initialplan ref="canonizeExecTraces"/>
			</plans>
		</configuration>
	</configurations>

</agent>

-----------
<!--
	<H3>The Consolidate agent.</H3>

	This agent consolidates all the information handled in the process of RNF analyze
-->

<agent xmlns="http://jadex.sourceforge.net/jadex"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://jadex.sourceforge.net/jadex
		http://jadex.sourceforge.net/jadex-0.96.xsd"
	name="ConsolidatorAgent"
	package="jadex.examples.TMMAS.agent.consolidatorAgent">

	<imports>
		<import>jadex.adapter.fipa.SFipa</import>
		<import>handleClasses.CanonicalDefinition</import>
		<import>handleClasses.OpsOfRNF</import>
	</imports>

	<plans>
		<!-- Plan for receive the log and register it in the belief base. -->
		<plan name="ReceiveRaEvCaMo">
			<body class="ReceiveRaEvCaMoPlan"/>
		</plan>

		<plan name="ConsolidateInformation">
			<body class="ConsolidateInformationPlan"/>
			<trigger>
				<internalevent ref="call_ConsolidateInformationPlan"/>
```

```
                </trigger>
            </plan>
        </plans>

        <events>
            <messageevent name="receiveCanonicModelMSG" type="fipa"
direction="receive">
                <parameter name="performative" class="String" direction="fixed">
                    <value>SFipa.INFORM</value>
                </parameter>
                <parameter name="content-class" class="Class" direction="fixed">
                    <value>CanonicalDefinition.class</value>
                </parameter>
            </messageevent>

                    <messageevent name="receiveRastEvaluationMSG"
type="fipa" direction="receive">
                <parameter name="performative" class="String" direction="fixed">
                    <value>SFipa.INFORM</value>
                </parameter>
                <parameter name="content-class" class="Class" direction="fixed">
                    <value>OpsOfRNF.class</value>
                </parameter>
            </messageevent>

            <!-- This internal event means the call for the Send RNF Evaluation Plan.-->
            <internalevent name="call_ConsolidateInformationPlan"/>
        </events>

        <configurations>
            <configuration name="default">
                <plans>
                    <!-- Get the log. -->
                    <initialplan ref="ReceiveRaEvCaMo"/>
                </plans>
            </configuration>
        </configurations>

</agent>
-------
<!--
        <H3>The Monitor agent.</H3>

        This agent monitors the RNF
-->

<agent xmlns="http://jadex.sourceforge.net/jadex"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://jadex.sourceforge.net/jadex
                    http://jadex.sourceforge.net/jadex-0.96.xsd"
        name="MonitorAgent"
        package="jadex.examples.TMMAS.agent.monitorAgent">
```

```
<imports>
	<import>jadex.adapter.fipa.SFipa</import>
	<import>jadex.examples.TMMAS.util.handleClasses.*</import>
	<import>org.w3c.dom.Document</import>
	<import>java.util.ArrayList</import>
</imports>

<beliefs>
	<belief name="stdPath" class="String">
		<fact>"C:/logContent.xml"</fact>
	</belief>
	<belief name="KSBase" class="KnownStructuresBase">
		<fact>new KnownStructuresBase()</fact>
	</belief>
	<belief name="USBase" class="UnknownStructureBase">
		<fact>new UnknownStructureBase()</fact>
	</belief>
	<belief name="sign" class="ArrayList">
		<fact>null</fact>
	</belief>
	<belief name="execTraces" class="Document">
		<fact>null</fact>
	</belief>
</beliefs>

<plans>
	<!-- Plan to get the log automatically. -->
	<plan name="autoGetLog">
		<body class="AutoGetFilePlan"/>
	</plan>

	<!-- Plan to get the log manually. -->
	<plan name="manGetLog">
		<body class="ManualGetFilePlan"/>
		<trigger>
			<internalevent ref="request_manualLog"/>
		</trigger>
	</plan>

	<!-- Plan to create the sign. -->
	<plan name="createSign">
		<body class="ImplementSignPlan"/>
		<trigger>
			<internalevent ref="request_signImplementation"/>
		</trigger>
	</plan>

	<!-- Plan to verify the sign. -->
	<plan name="verifySign">
		<body class="VerifySignPlan"/>
		<trigger>
```

```
                        <internalevent ref="request_signVerification"/>
                    </trigger>
                </plan>
            </plans>

            <events>
                <!-- This message event sends the traces of execution to the Canonize agent.-->
                <messageevent name="sendExecutionTracesMSG" type="fipa"
direction="send">
                    <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REQUEST</value>
                    </parameter>
                    <parameter name="language" class="String" direction="fixed">
                        <value>SFipa.NUGGETS_XML</value>
                    </parameter>
                </messageevent>

                <!-- This internal event means the need of manual search for the file.-->
                <internalevent name="request_manualLog"/>

                <!-- This internal event means the need of manual search for the file.-->
                <internalevent name="request_signImplementation"/>

                <!-- This internal event means the need of manual verifying the file.-->
                <internalevent name="request_signVerification"/>
            </events>

            <configurations>
                <configuration name="default">
                    <plans>
                        <!-- Get the log in the standard place. -->
                        <initialplan ref="autoGetLog"/>
                    </plans>
                </configuration>
            </configurations>

</agent>
```

B.2 - Esta seção apresenta o código-fonte da implementação em Java das classes dos agentes para estruturação de seu comportamento ou ações executadas no SMA.

```
package jadex.examples.TMMAS.agent.analyzerAgent;

import jadex.runtime.Plan;

public class AnalyzerStart extends Plan {

	private static final long serialVersionUID = 1L;

	public void body() {
		// This will be a new plan. It will receive a message from another agent
		// and then make something


		System.out.println("Analyzer: Starting AnalyzerStart.");
		System.out.println("Analyzer: The Analyzer Agent is going to start the analyzing
process.");

		//IMessageEvent msg = waitForMessageEvent("receiveExecTracesMSG");
	}
}
---------------------------------

package jadex.examples.TMMAS.agent.analyzerAgent;

import jadex.examples.TMMAS.util.handleClasses.AuxFunctions;
import jadex.examples.TMMAS.util.handleClasses.ExtensionFileFilter;
import jadex.examples.TMMAS.util.rules.AlternativeEvaluation;
import jadex.runtime.Plan;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

import javax.swing.JFileChooser;
import javax.swing.filechooser.FileFilter;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
```

```
import org.xml.sax.SAXException;

import com.thoughtworks.xstream.XStream;

public class EvaluateRNFPlan extends Plan {

	private static final long serialVersionUID = 1L;
	private List<AlternativeEvaluation> evaluationList = new
ArrayList<AlternativeEvaluation>();
	Document policy = null;

	public void body() {

		System.out.println("Analyzer: Starting RNF Evaluation.");


			String path = (String) getBeliefbase().getBelief("stdPath").getFact();//
Get in the beliefset the standard policy path

				try {
					DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
					DocumentBuilder builder =
dbf.newDocumentBuilder();
					policy = builder.parse(new File(path));// Policy
Document on the hand
					getBeliefbase().getBelief("policy").setFact(policy);//
Save it in the belief
				} catch (IOException e) {
					System.out.println("Analyser: The policy file is not
present in the standard path or is an invalid XML file!");

				String dataHora = new AuxFunctions().getDateTime();
				System.out.println("Analyser: The policy file has been gotten
automatically: " + dataHora);
			} catch (ParserConfigurationException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				} catch (SAXException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}

		//--> Policy at hand! We hope...
			System.out.println("Analyser: OK");

		if (policy != null)//Id it had success on define the policy
		{
			System.out.println("Analyser: True!");

			DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
```

```
DocumentBuilder builder = null;
Document execTraces = null;

try {
	builder = dbf.newDocumentBuilder();
	execTraces = builder.parse(new File("C:\\logContent.xml"));
} catch (ParserConfigurationException e1) {
	// TODO Auto-generated catch block
	e1.printStackTrace();
} catch (SAXException e1) {
	// TODO Auto-generated catch block
	e1.printStackTrace();
} catch (IOException e1) {
	// TODO Auto-generated catch block
	e1.printStackTrace();
}// Execution Traces Document on the hand

System.out.println("Analyser: Done");

Element root = policy.getDocumentElement();// Get the ROOT element

NodeList patternTopicoNodes =
root.getElementsByTagName("patternTopico");// Get all the NODES

String question = "";
String alternative = "";

for (int i = 0; i < patternTopicoNodes.getLength(); i++)// For each node
{
	NodeList patternTopicoChildNodes =
patternTopicoNodes.item(i).getChildNodes();

	NodeList patternQuestoesChildNodes = null;
	for (int t = 0; t < patternTopicoChildNodes.getLength(); t++)//
For each node
	{
		if (patternTopicoChildNodes.item(t).getNodeName() ==
"patternQuestoes")
			patternQuestoesChildNodes =
patternTopicoChildNodes.item(t).getChildNodes();

			//-->>

	System.out.println(patternTopicoChildNodes.item(t).getNodeName() + "Test 1.");

			NodeList patternAlternativasChildNodes = null;
			for (int j = 0; j <
patternQuestoesChildNodes.getLength(); j++)// For each node
			{
				if
(patternQuestoesChildNodes.item(j).getNodeName() == "tituloQuestao")
```

```
                                                alternative =
patternQuestoesChildNodes.item(j).getNodeValue();

                                                //-->>

        System.out.println(patternTopicoChildNodes.item(j).getNodeName() + "Test 2.");

                                                if
(patternQuestoesChildNodes.item(j).getNodeName() == "patternAlternativas")
                                                patternAlternativasChildNodes =
patternTopicoChildNodes.item(j).getChildNodes();

                                                //-->>

        System.out.println(patternTopicoChildNodes.item(j).getNodeName() + "Test 3.");

                                                NodeList variaveisEssenciaisChildNodes = null;
                                                for (int k = 0; k <
patternAlternativasChildNodes.getLength(); k++)// For each node
                                                {
                                                        if
(variaveisEssenciaisChildNodes.item(k).getNodeName() == "tituloAlternativa")
                                                                alternative =
variaveisEssenciaisChildNodes.item(k).getNodeValue();

                                                        if
(patternAlternativasChildNodes.item(k).getNodeName() == "variaveisEssenciais")
                                                                variaveisEssenciaisChildNodes
= patternAlternativasChildNodes.item(k).getChildNodes();

                                                        NodeList
sinonimosVariaveisEssenciaisChildNodes = null;
                                                        for (int n = 0; n <
variaveisEssenciaisChildNodes.getLength(); n++)// For each node
                                                        {
                                                                if
(variaveisEssenciaisChildNodes.item(n).getNodeName() == "idVariaveisEssenciais")
                                                                        evaluationList.add(new
AlternativeEvaluation(question, alternative,
variaveisEssenciaisChildNodes.item(n).getNodeValue(), execTraces));

                                                                if
(variaveisEssenciaisChildNodes.item(n).getNodeName() == "sinonimosVariaveisEssenciais")

        sinonimosVariaveisEssenciaisChildNodes =
patternAlternativasChildNodes.item(k).getChildNodes();// TITULO idVariaveisEssenciais

                                                                for (int f = 0; f <
sinonimosVariaveisEssenciaisChildNodes.getLength(); f++)// For each node
                                                                {
```

```
                                                                    evaluationList.add(new
AlternativeEvaluation(question,                                                    alternative,
sinonimosVariaveisEssenciaisChildNodes.item(k).getNodeValue(), execTraces));
                                                                    }

                                                                }
                                                            }
                                                        }
                                                    }
                                                }

                    System.out.println("Analyser: RNF Evaluation complete.");

                    path = "C:\\TranspRastEval.xml";
                    XStream xstream = new XStream();

                    xstream.alias("Report", List.class);
                    xstream.alias("ItemOfRNF", AlternativeEvaluation.class);

                    try {
                            PrintWriter writer = new PrintWriter(path);
                            writer.println(xstream.toXML(this.evaluationList));
                            writer.flush();
                            writer.close();
                    } catch (FileNotFoundException e) {
                            e.printStackTrace();
                            System.out.println("Analyser: An error occured while saving the
report.");
                    }

                            System.out.println("Analyser:        Evaluation        exported        to
'C:\\TranspRastEval.xml'.");
                    }


            }
}
-----------------------------------


<html><head><title>

jadex.examples.analizerAgent

</title></head><body>

Agent for analyze all the information present in the log and evaluate the RNF according with
the expected in the operationalizations implemented.

</body></html>

-----------------------------------
```

```
package jadex.examples.TMMAS.agent.analyzerAgent;

import jadex.examples.TMMAS.util.handleClasses.CanonicalDefinition;
import jadex.model.IMBelief;
import jadex.model.IMBeliefbase;
import jadex.runtime.IInternalEvent;
import jadex.runtime.IMessageEvent;
import jadex.runtime.Plan;

public class ReceiveLogPlan extends Plan {

	private static final long serialVersionUID = 1L;

	public void body() {

		System.out.println("Analyser: The Analyser agent is ready.");
		System.out.println("Analyser: Waiting for a Canonic log.");
		while (true) {
			CanonicalDefinition CanonicModel = new CanonicalDefinition();
			// This will be a new plan. It will receive a message from another
			// agent and then make something

			IMessageEvent msg =
waitForMessageEvent("receiveCanonicLogMSG");

			System.out.println("Analyser: Log recebido.");

			CanonicModel = (CanonicalDefinition) msg.getContent();// Get the
content of the message

			// Create a new belief in run time to store the canonic log
			IMBeliefbase model = (IMBeliefbase) getBeliefbase()
					.getModelElement();
			IMBelief belief = model.createBelief("CanonicLog",
	CanonicalDefinition.class, 0, "false");// Create the belief
			System.out.println("Analyser: Belief Created.");
			getBeliefbase().registerBelief(belief);// Register belief at runtime
			System.out.println("Analyser: Belief Registreated.");
			getBeliefbase().getBelief("CanonicLog").setFact(CanonicModel);// Set
the log in the new belief

			// Internal event that represents the reception of the canonic log
			IInternalEvent event = createInternalEvent("call_EvaluateRNFPlan");
			dispatchInternalEvent(event);
		}
	}
}
```

```
package jadex.examples.TMMAS.agent.analyzerAgent;
```

```
import jadex.adapter.fipa.AgentIdentifier;
import jadex.adapter.fipa.SFipa;
import jadex.examples.TMMAS.util.handleClasses.CanonicalDefinition;
import jadex.examples.TMMAS.util.handleClasses.OpsOfRNF;
import jadex.runtime.IMessageEvent;
import jadex.runtime.Plan;

public class SendRNFEvaluationPlan extends Plan {

	private static final long serialVersionUID = 1L;

	public void body() {
		// TODO Auto-generated method stub
		System.out.println("Analyzer: Starting SendRastreabiliityEvaluationPlan.");
		CanonicalDefinition modeloCanonico = (CanonicalDefinition) getBeliefbase()
				.getBelief("CanonicLog").getFact();
		OpsOfRNF RNFEvaluation = (OpsOfRNF) getBeliefbase()
				.getBelief("RNFEvaluationRegister").getFact();

		System.out.println("Enviando agora.");
		IMessageEvent msgCanonicModel =
createMessageEvent("sendCanonicModelMSG");
		msgCanonicModel.getParameterSet(SFipa.RECEIVERS).addValue(
				new AgentIdentifier("ConsolidatorAgent", true));
		msgCanonicModel.setContent(modeloCanonico); // The Java object is

	// directly used as content.
		sendMessage(msgCanonicModel);

		IMessageEvent msgRastEvaluation =
createMessageEvent("sendRastEvaluationMSG");
		msgRastEvaluation.getParameterSet(SFipa.RECEIVERS).addValue(
				new AgentIdentifier("ConsolidatorAgent", true));
		msgRastEvaluation.setContent(RNFEvaluation); // The Java

				// object is

				// directly used

				// as content.
		sendMessage(msgRastEvaluation);

		// Eliminating the current belief to be ready to receive others logs
		getBeliefbase().deleteBelief("CanonicLog");
		getBeliefbase().deleteBelief("RNFEvaluationRegister");

		System.out.println("Analyser: Messages sent to Consolidador Agent.");
	}
}

---------------------------------
```

```
package jadex.examples.TMMAS.agent.canonizerAgent;

import java.io.File;
import java.util.List;

import org.w3c.dom.Document;

import jadex.runtime.Plan;

import com.thoughtworks.xstream.persistence.FilePersistenceStrategy;
import com.thoughtworks.xstream.persistence.PersistenceStrategy;
import com.thoughtworks.xstream.persistence.XmlArrayList;

public class CanonizeExecTracesPlan extends Plan {

	/**
	 * This class is responsible to translate the specific log to the canonic
	 * model
	 */
	private static final long serialVersionUID = 1L;

	public void body() {

		System.out.println("Canonizer: Starting CanonizeExecTracesPlan.");
		System.out.println("Canonizer: The Canonizer Agent is going to start the
canonizing process.");


		//Prepares the file strategy to directory
		PersistenceStrategy strategy = new FilePersistenceStrategy(new
File("C:\\USBaseList.xml"));
		// creates the list:
		@SuppressWarnings("unchecked")
		List<Document> USBaseList = new XmlArrayList(strategy);

		System.out.println("Canonizer: Trying to canonizing in a parallel process...");

	}
}

-----------------------------------
<html><head><title>

jadex.examples.canonizerAgent

</title></head><body>

Agent for standardize the multiple kinds of logs in a canonized log.

</body></html>

-----------------------------------
```

```
/* This class  has the plan to research for the unknown tags and if it is recognized,
 * update the known structure base
 */
package jadex.examples.TMMAS.agent.canonizerAgent;

public class ReasearchUnknownTag {


}

---------------------------------
package jadex.examples.TMMAS.agent.canonizerAgent;

import jadex.runtime.Plan;

import java.io.File;
import java.io.IOException;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;
import org.xml.sax.SAXException;

import com.thoughtworks.xstream.persistence.FilePersistenceStrategy;
import com.thoughtworks.xstream.persistence.PersistenceStrategy;
import com.thoughtworks.xstream.persistence.XmlArrayList;

public class ReceiveUnkStructuresPlan extends Plan {

	/**
	 *
	 */
	private static final long serialVersionUID = 1L;

	public void body() {
	// This will be a new plan. It will receive a message from another agent and then make
something
	// In this moment, the canonizer agent only save the execution traces having unknown
structure to treat it after.
			System.out.println("Canonizer: The Canonizer Agent is ready.");
			System.out.println("Canonizer: Starting ReceiveUnkStructuresPlan.");


			Document execTraces = null;
			try {
				DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
				DocumentBuilder builder = dbf.newDocumentBuilder();
```

```
                    execTraces = builder.parse(new File("c:/logContent.xml"));// Execution
Traces Document on the hand

                } catch (ParserConfigurationException e) {
                    // TODO Auto-generated catch block
                    System.out.println("Canonizer: The file is incorrect!");
                    e.printStackTrace();
                } catch (SAXException e) {
                    // TODO Auto-generated catch block
                    System.out.println("Canonizer: A SAX exception occurred!");
                    e.printStackTrace();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    System.out.println("Canonizer: The file 'c:/logContent.xml' was not
found!");
                }// Execution Traces Document on the hand


                //Document execTraces = (Document) getParameter("execTraces").getValue();

                System.out.println(execTraces.getDocumentElement().getNodeName());// Get
the ROOT element

                System.out.println("Canonizer: Unknown Execution Traces file received.");

                //Prepares the file strategy to directory
                PersistenceStrategy strategy = new FilePersistenceStrategy(new
File("/Temp"));
                // creates the list:
                List uSBaseList = new XmlArrayList(strategy);
                // adds the xml files
                uSBaseList.add(execTraces);

                System.out.println("Canonizer: New Unknown Structure Base registered.");
        }
}
```

```
package jadex.examples.TMMAS.agent.canonizerAgent;

import jadex.adapter.fipa.AgentIdentifier;
import jadex.adapter.fipa.SFipa;
import jadex.examples.TMMAS.util.handleClasses.CanonicalDefinition;
import jadex.runtime.IMessageEvent;
import jadex.runtime.Plan;

public class SendCanonizedLog extends Plan {

        private static final long serialVersionUID = 1L;

        public void body() {
                // TODO Auto-generated method stub
```

```
                System.out.println("Canonizer: Starting SenCanonizeLog.");
                CanonicalDefinition modeloCanonico = (CanonicalDefinition) getBeliefbase()
                                .getBelief("CanonicLog").getFact();

                IMessageEvent me = createMessageEvent("sendCanonicLogMSG");
                me.getParameterSet(SFipa.RECEIVERS).addValue(
                                new AgentIdentifier("AnalyzerAgent", true));
                me.setContent(modeloCanonico); // The Java object is directly used as
                                                                                // content.
                sendMessage(me);
        }
}
----------------------------------

package jadex.examples.TMMAS.agent.consolidatorAgent;

import jadex.examples.TMMAS.util.handleClasses.CanonicalDefinition;
import jadex.examples.TMMAS.util.handleClasses.OpsOfRNF;
import jadex.model.IMBelief;
import jadex.model.IMBeliefbase;
import jadex.runtime.IInternalEvent;
import jadex.runtime.IMessageEvent;
import jadex.runtime.Plan;

public class ReceiveRaEvCaMoPlan extends Plan {
        private static final long serialVersionUID = 1L;
        public void body() {

                System.out.println("Consolidator: Starting ReceiveRaEvCaMoPlan.");
                while (true) {
                        CanonicalDefinition CanonicModel = new CanonicalDefinition();
                        OpsOfRNF rastrEvaluation = new OpsOfRNF ();
                        // This will be a new plan. It will receive a message from another
                        // agent and then make something
                        System.out.println("Consolidator: The Consolidator Agent is waiting for
the registers.");
                        IMessageEvent                msgCanonicLog                =
waitForMessageEvent("receiveCanonicModelMSG");
                        IMessageEvent              msgRastEvaluation              =
waitForMessageEvent("receiveRastEvaluationMSG");

                        System.out.println("Consolidator: The files were received.");

                        CanonicModel = (CanonicalDefinition) msgCanonicLog.getContent();//
Get

                                                        // the

                                                        // content

                                                        // of
```

```
                                                    // the

                                                    // message
                        rastrEvaluation = (OpsOfRNF) msgRastEvaluation.getContent();// Get
the content of the message

                        // Create a new belief in run time to store the canonic log
                        IMBeliefbase model = (IMBeliefbase) getBeliefbase()
                                        .getModelElement();
                        IMBelief                        beliefCM                        =
model.createBelief("CanonicModel",CanonicalDefinition.class, 0, "false");// Create the belief
                        IMBelief beliefRR = model.createBelief("RNFRegister",OpsOfRNF.class,
0, "false");// Create the belief

                        System.out.println("Beliefs Created.");
                        getBeliefbase().registerBelief(beliefCM);// Register belief at

                // runtime
                        getBeliefbase().registerBelief(beliefRR);// Register belief at

                // runtime
                        System.out.println("Belief Registreated.");

                        getBeliefbase().getBelief("CanonicModel").setFact(CanonicModel);//
                        Set
                        getBeliefbase().getBelief("RNFRegister").setFact(rastrEvaluation);// Set
the log in the new belief

                        // Internal event that represents the reception of the canonic log
                        IInternalEvent                  event                           =
createInternalEvent("call_ConsolidateInformationPlan");
                        dispatchInternalEvent(event);
                }
        }
}
```

```
package jadex.examples.TMMAS.agent.monitorAgent;

import jadex.examples.TMMAS.util.handleClasses.AuxFunctions;

import jadex.runtime.IInternalEvent;
import jadex.runtime.Plan;

import java.io.File;
import java.io.IOException;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
```

```
import org.w3c.dom.Document;
import org.xml.sax.SAXException;

import com.thoughtworks.xstream.persistence.FilePersistenceStrategy;
import com.thoughtworks.xstream.persistence.PersistenceStrategy;
import com.thoughtworks.xstream.*;
public class AutoGetFilePlan extends Plan {

	private static final long serialVersionUID = 1L;
	public void body() {
		System.out.println("Monitor: Starting auto get traces plan.");

		String path = (String) getBeliefbase().getBelief("stdPath").getFact();// Get beliefset the standard log path
		Boolean firstTime = true;

		PersistenceStrategy pst = new FilePersistenceStrategy(new File("C:\\USBaseList.xml"));
		System.out.println("Monitor: USBaseList loaded!");

		while (true) // Infinite time of monitoring
		{
			try {
				DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
				DocumentBuilder builder = dbf.newDocumentBuilder();
				Document execTraces = builder.parse(new File(path));// Execution Traces Document on the hand
				getBeliefbase().getBelief("execTraces").setFact(execTraces);// Save it in the belief

				System.out.println("Monitor: Execution traces obtained...");
				System.out.println("Monitor: Creating the file sign...");
				// --> CALL - Internal event that means the agent can make the sign of the XML
				IInternalEvent event = createInternalEvent("request_signImplementation");
				dispatchInternalEvent(event);
			} catch (IOException e) {
				System.out.println("Monitor: The log is not present in the standard path or is an invalid XML file!");
				System.out.println("Monitor: Set the log file or wait the next monitor agent act!");

				if (firstTime) {
					System.out.println("Monitor: Preparing to open file manually...");

					// --> CALL - "Internal event that means the agent doesn't find the log and needs user manual intervation
```

```
                                        IInternalEvent event =
createInternalEvent("request_manualLog");
                                        dispatchInternalEvent(event);

                                        firstTime = false;// The windows must appear just once
time
                                        System.out.println("Monitor: Th e parallel
monitoring is still working. Wait 5 minutes...");
                                }
                        } catch (ParserConfigurationException e) {
                                System.out.println("Monitor: The parser was not configured
correctly.");
                                e.printStackTrace();
                        } catch (SAXException e) {
                                System.out.println("Monitor: Some problem occur while
parsing the file.");
                                e.printStackTrace();
                        }

                        String dataHora = new AuxFunctions().getDateTime();
                        System.out.println("Monitor: A log is being analysed at: " + dataHora);
                        waitFor(30000000); // 5 minutes to analyse again
                }
        }
}

-----------------------------------
package jadex.examples.TMMAS.agent.monitorAgent;

import jadex.examples.TMMAS.util.handleClasses.AuxFunctions;
import jadex.runtime.IInternalEvent;
import jadex.runtime.Plan;

import java.util.ArrayList;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

public class ImplementSignPlan extends Plan {

        private static final long serialVersionUID = 1L;

        public void body() {

                System.out.println("Monitor: Starting Sign Definition.");

                ArrayList<String> sign = new ArrayList<String>();
                Document execTraces = (Document) getBeliefbase()
.getBelief("execTraces").getFact(); // Get from beliefset the file

                Element root = execTraces.getDocumentElement();// Get the ROOT element
```

```
                sign.add(root.getNodeName());// Start the identifier using the root name

                NodeList nodes = root.getChildNodes();// Get all the NODES

                for (int i = 0; i < nodes.getLength(); i++)// For each node
                {
                        if (!sign.contains(nodes.item(i).getNodeName()) &&
(nodes.item(i).getNodeName() != "#text"))// If the node name is not inside the array
                        {
                                sign.add(nodes.item(i).getNodeName());// Insert it
                        }

                        NodeList elements = nodes.item(i).getChildNodes(); // Get the
elements inside the node
                        for (int p = 0; p < elements.getLength(); p++) {
                                if ((!sign.contains(elements.item(p).getNodeName())) &&
(elements.item(p).getNodeName() != "#text"))// If the node name is not inside the array
                                {
                                        sign.add(elements.item(p).getNodeName());
                                }
                        }
                }

                System.out.println("This is the sign content:");
                for (int p = 0; p < sign.size(); p++) {
                        System.out.print(sign.get(p) + " / ");
                }
                System.out.println();
                getBeliefbase().getBelief("sign").setFact(sign);// Save it in the belief

                // --> CALL -"Internal event that calls the sign verification
                IInternalEvent event = createInternalEvent("request_signVerification");
                dispatchInternalEvent(event);
                String dataHora = new AuxFunctions().getDateTime();
                System.out.println("Monitor: Sign created at: " + dataHora);
        }
}
```

```
package jadex.examples.TMMAS.agent.monitorAgent;

import jadex.examples.TMMAS.util.handleClasses.ExtensionFileFilter;
import jadex.runtime.IInternalEvent;
import jadex.runtime.Plan;

import java.io.File;
import java.io.IOException;

import javax.swing.JFileChooser;
import javax.swing.filechooser.FileFilter;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
```

```
import org.w3c.dom.Document;
import org.xml.sax.SAXException;

public class ManualGetFilePlan extends Plan {

	private static final long serialVersionUID = 1L;

	@Override
	public void body() {

		System.out.println("Monitor: Starting ManualGetFilePlan.");

		JFileChooser fc = new JFileChooser(); // FileChooser instance
		fc.setDialogType(JFileChooser.OPEN_DIALOG);// Set open file dialog
		fc.setDialogTitle("Open File");// Set the title to the open file

		FileFilter filter = new ExtensionFileFilter("XML or LOG", new String[] {
				"xml", "log" });// Instantiate a filter of kinds of file
								// extensions
		fc.setFileFilter(filter);// Setting the filter in the FileChooser
		File file = null;

		int returnVal = fc.showDialog(fc, null);

		if (returnVal == JFileChooser.APPROVE_OPTION) {
			file = fc.getSelectedFile();// Get the file selected
		}

		try {
			DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
			DocumentBuilder builder = dbf.newDocumentBuilder();
			Document execTraces = builder.parse(file);// Execution Traces

		// Document on the hand

			getBeliefbase().getBelief("execTraces").setFact(execTraces);// Save

					// it in
					// the
					// belief

			System.out.println("Monitor: Calling the file sign creation...");
			// "Internal event that means the agent can make the sign of the XML
			IInternalEvent event =
createInternalEvent("request_signImplementation");
			dispatchInternalEvent(event);

		} catch (IOException e) {
			System.out.println("Monitor: Problem while opening the file!");
		} catch (ParserConfigurationException e) {
```

```
                        System.out.println("Monitor: The parser was not configured
correctly.");
                        e.printStackTrace();
                } catch (SAXException e) {
                        System.out.println("Monitor: Some problem occur while doing the
parse of the file.");
                        e.printStackTrace();
                }
        }
}
----------------------------------
package jadex.examples.TMMAS.agent.monitorAgent;

import jadex.examples.TMMAS.util.handleClasses.AuxFunctions;
import jadex.runtime.IInternalEvent;
import jadex.runtime.IMessageEvent;
import jadex.runtime.Plan;
import java.io.File;
import java.io.IOException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.xml.sax.SAXException;

public class ReceiveExtFilePlan extends Plan {
        private static final long serialVersionUID = 1L;
        @Override
        public void body() {

                System.out.println("Monitor: Starting ReceiveExtFilePlan.");

                IMessageEvent msg = waitForMessageEvent("receiveTEMSG"); //Wait for the
message from Monitor agent
                System.out.println("Monitor: Execution Traces file received from outside.");

                Boolean firstTime = true;
                while (true) {
                        try {

                                File xmlFile = (File) msg.getContent();// Get the content of the
message

                                DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
                                DocumentBuilder db = dbf.newDocumentBuilder();
                                Document execTraces = db.parse(xmlFile);// Execution Traces
Document

                                getBeliefbase().getBelief("execTraces").setFact(execTraces);//
Save it in the belief
```

```
                                System.out.println("Monitor: Calling the file sign creation...");
                                // "Internal event that means the agent can make the sign of
the XML
                                IInternalEvent event =
createInternalEvent("request_signImplementation");
                                dispatchInternalEvent(event);
                        } catch (IOException e) {
                                System.out.println("Monitor: The log is not present in the
standard path or is an invalid XML file!");
                                System.out.println("Monitor: Set the log file or wait the next
monitor agent act!");

                                if (firstTime) {
                                        System.out.println("Monitor: Preparing to open file
manually...");

                                        // "Internal event that means the agent doesn't find
the log
                                        // and needs user manual intervation
                                        IInternalEvent event =
createInternalEvent("request_manualLog");
                                        dispatchInternalEvent(event);

                                        firstTime = false;// The windows must appear just once
time
                                        System.out.println("Monitor: The paralel monitoring
still works. Wait 5 minutes...");
                                }
                        } catch (ParserConfigurationException e) {
                                System.out.println("Monitor: The parser was not configured
correctly.");
                                e.printStackTrace();
                        } catch (SAXException e) {
                                System.out.println("Monitor: Some problem occur while doing
the parse of the file.");
                                e.printStackTrace();
                        }

                        String dataHora = new AuxFunctions().getDateTime();
                        System.out.println("Monitor: Revisao de log realizada em: " +
dataHora);
                        waitFor(60000000); // 10 minutes
                }
        }
}
---------------------------------
package jadex.examples.TMMAS.agent.monitorAgent;

import jadex.adapter.fipa.AgentIdentifier;
import jadex.adapter.fipa.SFipa;
import jadex.examples.TMMAS.util.handleClasses.AuxFunctions;
import jadex.examples.TMMAS.util.handleClasses.KnownStructuresBase;
```

```
import jadex.examples.TMMAS.util.handleClasses.UnknownStructureBase;
import jadex.runtime.IMessageEvent;
import jadex.runtime.Plan;
import java.util.ArrayList;
import org.w3c.dom.Document;

public class VerifySignPlan extends Plan {

	private static final long serialVersionUID = 1L;
	@Override
	@SuppressWarnings("unchecked")
	public void body() {
		System.out.println("Monitor: Starting Sign Verification.");

		ArrayList<String> sign = new ArrayList<String>();
		sign = (ArrayList<String>) getBeliefbase().getBelief("sign").getFact();// Get from
beliefset the sign

		KnownStructuresBase knownStructuresBase = (KnownStructuresBase)
getBeliefbase().getBelief("KSBase").getFact();// Get from beliefset the signBase

		UnknownStructureBase unknownStructureBase = (UnknownStructureBase)
getBeliefbase().getBelief("USBase").getFact();// Get from beliefset the signBase

		boolean recognized = true;
		if (knownStructuresBase.getSize() != 0)// if exist any known structure in the
base
		{for (int i = 0; i < knownStructuresBase.getSize(); i++)// For each known
structure
			{for (int p = 0; p < sign.size(); p++)// For each tag in the sign
				{if
(!knownStructuresBase.existInKnownStructure(sign.get(p)))// If the structure is unknown
					{
	unknownStructureBase.insertUnknownStructure(sign.get(p));// Register it in the
Unknown Structures Base
	recognized = false; // Mark that the sign was not fully recognized
					}
				}
			}
		}
		else // if does not exist any known structure in the base
		{
			recognized = false; // Mark that the sign was not fully recognized
			for (int p = 0; p < sign.size(); p++)// For each tag in the sign
			{
				unknownStructureBase.insertUnknownStructure(sign.get(p));//
Register it in the Unknown Structures Base
			}
		}

		Document execTraces = (Document)
getBeliefbase().getBelief("execTraces").getFact();// Get from beliefset the traces of execution
```

```
                // To send it to the proper agent
                if (recognized)
                {
                        IMessageEvent me =
createMessageEvent("sendExecutionTracesMSG");
                        me.getParameterSet(SFipa.RECEIVERS).addValue(new
AgentIdentifier("AnalyzerAgent", true));
                        me.getParameterSet(SFipa.RECEIVERS).addValue(new
AgentIdentifier("AnalyzerAgent0", true));
                        me.getParameterSet(SFipa.RECEIVERS).addValue(new
AgentIdentifier("AnalyzerAgent1", true));
                        me.setContent(execTraces);
                        // The Java object is directly used as content.
                        sendMessage(me);
                        System.out.println("Monitor: Execution Traces sent to Analyzer
Agent");
                }
                else
                {
                        IMessageEvent me =
createMessageEvent("sendExecutionTracesMSG");
                        me.getParameterSet(SFipa.RECEIVERS).addValue(new
AgentIdentifier("CanonizerAgent", true));
                        me.getParameterSet(SFipa.RECEIVERS).addValue(new
AgentIdentifier("CanonizerAgent0", true));
                        me.getParameterSet(SFipa.RECEIVERS).addValue(new
AgentIdentifier("CanonizerAgent1", true));
                        me.getParameterSet(SFipa.RECEIVERS).addValue(new
AgentIdentifier("CanonizerAgent2", true));
                        me.setContent(execTraces);
                        // The Java object is directly used as content.
                        sendMessage(me);
                        System.out.println("Monitor: Execution Traces sent to Canonizer
Agent");
                        // Update the unknownStructuresBase
                        unknownStructureBase.saveUknownStructures();
                }
                String dataHora = new AuxFunctions().getDateTime();
                System.out.println("Monitor: Sign verified at: " + dataHora);
        }
}
```