Método Sistêmico com Suporte em GORE para Análise de Conformidade de Requisitos não Funcionais Implementados em Software

Systemic Method Supported by GORE for Analysis of Software-Implemented Non-Functional Requirements Compliance

André Luiz de Castro Leal [11]

Henrique Prado Sousa [22]

Julio Cesar Sampaio do Prado Leite [33]

Data de submissão: 11/06/2015, Data de aceite: 01-03-2016

Resumo: A análise de requisitos não funcionais (RNF) é um desafio e vem sendo explorado na literatura científica há muito tempo. Tal iniciativa deve-se ao fato da existência do problema de se verificar o uso das operacionalizações desse tipo de requisito no software construído. Nesse trabalho apresentamos um método, com técnicas e ferramentas de apoio, que verificam se um software está em conformidade com padrões de RNF estabelecidos em catálogo como alternativa para o problema de verificação de RNF. A estratégia adotada utiliza agentes autônomos para verificação 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 que é eficaz na identificação de conformidades. Um diferencial na proposta é que, de maneira geral, os trabalhos com foco em verificação são fortemente voltados para a visão funcional, enquanto a solução aqui apresentada é inovadora na ligação dos RNF a sua efetiva implantação. Como prova de conceito aplicou-se e customizou-se uma técnica de padrões de RNFs baseados em orientação a objetivos em estudos de caso de exemplos do cotidiano prático de software. Como 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: conformidade, requisito não funcional, orientado a objetivos, NFR-framework

1 André Luiz de Castro Leal, D.Sc. - Universidade Federal Rural do Rio de Janeiro – UFRRJ - Seropédica – RJ {andrecastr@gmail.com}
2 Henrique Prado de Sá Sousa, M.Sc. - Universidade Federal Rural do Rio de Janeiro – UFRRJ - Seropédica – RJ. {hsousa@inf.puc-rio.br}
3 Julio César Sampaio do Prado Leite, D.Sc. - Pontifícia Universidade Católica do Rio de Janeiro – PUC-Rio - Rio de Janeiro – RJ. {jcspl@inf.puc-rio.br}

Abstract. The analysis of non-functional requirements (NFR) is a challenge and has been explored in the literature for a long time. This initiative is due to the fact of the existence of the problem of verifying the use of the NFR's operationalization in software. In this paper WE present a method, with supporting tools and techniques that check if a software complies with standards of non-functional requirements as described in catalog, as an alternative to the NFR verification problem. The strategy adopted in this thesis uses autonomous agents to check software compliance regarding THE operationalization of nfr, by using a knowledge base of *patterns* persisted in catalog. partial 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. our approach adds strongly over existing approaches in that it innovates in linking NFR to its effective implementation, while existing approaches focus on a functional view. For demonstration, it was applied and customized it a technique based on *patterns* of NFRs oriented goals on case studies examples of practical everyday software. As well as the construction of a framework for agents that operate in XML notation to identify compliance of software with respect to a catalog of RNF.

Keywords: compliance, non-functional requirements, goal oriented, NFR-framework

1 Introdução

Requisitos de restrição e qualidade em software propõem características em alto nível de abstração e sua avaliação possui grau de julgamento que depende do ponto de vista do interessado [1]. Por exemplo, dizer se é transparente ou não, tomado Transparência como um Requisito não Funcional (RNF) [2], pode não ser adequado, pois irá depender do grau de subjetividade de quem está na expectativa da informação. Além disso, conforme apresentado no Catálogo de Transparência de Software (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 [3]. Essas decomposições são tratadas de metas-flexíveis a partir do framework de Chung et al. (2000) [4].

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 [5]. A verificação dessas condições torna-se mais efetiva para os RFs, uma vez que dada a condição, a verificação 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 analisa. 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 [5].

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 requisitos [6]; certas restrições, são altamente subjetivas e só podem ser determinadas através de avaliações empíricas complexas [7]; RNFs podem se relacionar a um ou mais RFs e possuem dessa maneira uma característica de ortogonalidade [2]; RNFs tendem a conflitos e contradições, uma vez que são suscetíveis à percepção de quem o avalia [8]; não há regras "universais" e diretrizes para determinar quando RNFs são plenamente atendidos [9].

Analisar a implementação em software de RNF isoladamente a partir de sua natureza semântica ou conceitual é uma tarefa difícil diante das características apresentadas, portanto faz-se necessário refinar esses requisitos até que se tornem elementos mais concretos de julgamento [4]. Fenton e Pfleeger (1997) [1] sugerem 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. Lamsweerde (2001) [10] e Lamsweerde e Letier (2000) [11] sugerem especificações hierárquicas de decomposição de metas produzindo estados a serem alcançados e do comportamento do sistema necessário para atingir esses estados. Lamsweerde (2001) [10] argumenta que de modo considerável o refinamento de RNF 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 define a especialização em camadas de padrões (*patterns*) que envolvem desde requisitos de alto nível de abstração como qualidades RFs de mais baixo nível ou operacionalizações [4] [12]. O objetivo de sua criação em *patterns* é possibilitar reuso de seu conhecimento. Supakkul et al. (2010) [12] definem 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) [4] atuam no desafio quando propõem um framework para representar RNF e suas relações; em [12], que definem padrões de tratam desde o conceitual até o operacional para que RNFs fossem representados e avaliados com relação a seu grau de satisfação; e em [13] sugerem a adaptação da proposta de [12] quando exploram no modelo *Goal-Question-Metrics* (GQO) o tratamento do último nível do *Goal-Question-Metrics* (GQM) como operacionalizações baseadas em práticas da Engenharia de Software (ES).

Dentro do contexto de pesquisas de análise de conformidade de implementação de RNFs em software, alguns pontos de investigação se destacam:

- Proposta de uma arquitetura única: um ponto pouco explorado de pesquisa é tratar *patterns* em uma única arquitetura que possa servir de base para automação da análise de RNFs. Tal abordagem mostra-se importante, uma vez que analisar os desmembramentos dos RNFs e suas relações, bem como as alternativas de operacionalizações podem não ser triviais necessitando assim de uma estrutura que

viabilize a analise de satisfação de RNFs. A proposta é importante para que se dê foco ao reunso de uma arquitetura na análise de diferentes RNFs.

- Estabelecimento de um método: no sentido de se tornar possível a abordagem sugerida para análise de RNF, o objetivo primeiro da pesquisa é estabelecer um método a partir de uma abordagem *top-down* ao especificar RNF desde modelos conceituais até seu detalhamento em direção às operacionalizações, até que possam ser analisadas por agentes autônomos.

- Uso de catálogo de RNFs: O trabalho toma como insumo o catálogo definido em [47] que possui um desdobramento de vários RNFs a partir do requisito de Transparência, tais como: Acessibilidade, Usabilidade, Informativo, Entendimento, Auditabilidade e suas decomposições. Dessa forma, o catálogo proposto em CTS (2013) disponibiliza uma série de RNFs caracterizados conceitual e operacionalmente que podem permitir uma análise da sua implementação em software.

Até esse ponto de elicitação, especificação e refinamento das qualidades, o catálogo é concebido a partir da proposta do NFR *Framework*, tratado no trabalho como RNF *framework* [4]. O *framework* propõe a representação das qualidades e seus elos de contribuição positivos ou negativos, além de seu nível de satisfação (satisfeito a contento). Além disso, o *framework* permite a representação da concepção e raciocínio dos processos em grafos, chamados *Softgoal Interdependency Graph* (SIG) e oferece um catálogo de conhecimento, incluindo técnicas de desenvolvimento. O modelo ainda permite explicitar as operacionalizações que podem tornar os RNFs, tratados no *framework* como *softgoals* (metas-flexíveis), como satisfeitos a contento. Já no ponto de operacionalizações há a adoção do GQO [13]. Assim como o GQM [14], 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 ES em sua camada Operacionalization (O). Dessa forma, a camada da criação de operacionalizações procura tornar elementos plausíveis de análise ou julgamento, como por exemplo, utilizar de alternativas de RFs que deem suporte aos RNFs estabelecidos nas camadas de *Goal* (G).

- Adoção da abordagem de *Goal Oriented Requirements Engineering* (GORE): o trabalho constitui-se a partir de uma abordagem orientada a objetivos GORE [10], uma vez que abordagens tradicionais [15], [16] e [17] focam o desenvolvimento apenas na análise dos processos e dados e não capturam o conhecimento (*rationale*) para o software a ser desenvolvido. Isso em muitos casos não contempla toda a complexidade do domínio do problema onde o software está envolvido [18]. Além disso, GORE foca em atividades que precedem a formulação de requisitos do software. As principais atividades na abordagem de GORE perpassam por elicitação e refinamento de metas, suas análises e a associação de metas a agentes [18], tais agentes trabalham de forma conjunta, apesar de sua autonomia, para atingir as metas definidas. Portanto, GORE é uma abordagem que dá suporte ao método a ser proposto.

- Analisar se o uso de SMA é aplicável ao problema de análise de RNFs pré-concebidos em catálogo: A presente pesquisa aproxima-se da 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 um potencial para a autonomia das decisões dos agentes em relação a ações para atingir as metas definidas, contribuição entre vários agentes para atingir uma meta comum, estabelecer bases de conhecimento sobre suas ações. Nesse ponto, a pesquisa propõe o uso de agentes com orientação baseada em intencionalidade, ou agentes intencionais [19]. 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) [20] [21].

O presente artigo está estruturado da seguinte forma: na seção 2, está a contextualização geral da pesquisa; na seção 3, é apresentado o método; na seção 4, são apresentadas as arquiteturas utilizadas como solução para camada conceitual e de estruturas XML utilizadas no SMA; na seção 5, é apresentada a arquitetura de implementação do SMA; na seção 6, são apresentadas provas de conceito da aplicação do método em análise de RNF em software; na seção 7, estão descritas as conclusões finais.

2 Contextualização

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. A versão desenvolvida não contemplava uma sistematização de política de monitoração e sim uma modelagem intencional da arquitetura da política de funcionamento dos agentes, isso percebido após a evolução da pesquisa. Seus resultados foram publicados em [22] e [23].

Foi construído um SMA que efetuava coleta de dados dos traços de execução gerados a partir de agentes do software *Lattesscholar* (LS) [24]. O LS é um sistema multi-agentes (SMA) independente para contagem de citações científicas, que combina os serviços do sítio do *Lattes* [25] e do *Google Scholar* [26]. Sua execução é baseada na pesquisa inicial pelo nome do autor no sítio do 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* [27], do *Microsoft Academic Search* [28] e do *Google Scholar Citation* [29], 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) [30] 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:

a) MONITOR: o agente monitora constantemente os traços de execução gerados pelos agentes do LS, com o objetivo de captar os seus registros;

b) CANONIZADOR: tem por objetivo receber os registros dos traços de execução e transformar as informações em um padrão que possa ser consolidado.

c) 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;

d) 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;

e) 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.

A segunda versão dos agentes veio para corrigir algumas deficiências da primeira. Essas alterações perpassaram por: 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; leitura dos traços de execução: na primeira versão eram lidos traços de execução gerados a partir de formatos ASCii[4] e passaram a ser lidos apenas por conteúdos gerados em estruturas XML; outra importante alteração foi o uso de tags lidas das estruturas de comunicação dos agentes

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

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.

3 Modelo central do método sistêmico para análise de RNF

Um modelo central é exposto para que se dê entendimento da completude e complexidade do trabalho, 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 1 apresenta a partir de *Structured Analysis and Design Technique* (SADT)[5] [17] 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 [4]; (A2) Definir *patterns* (proposto a partir de [12] e [13]; (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, os 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 representados por 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 propotos o método ou técnica utilizada em cada passo. A construção do método proposto é portanto uma conjunção de métodos e técnicas propostas por autores da literatura científica 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.

5 Na notação SADT: caixas representam atividades, setas a esquerda representam dados de entrada, setas para baixo representam controles, setas para cima representam mecanismos, e setas para direta representa saídas das atividades.

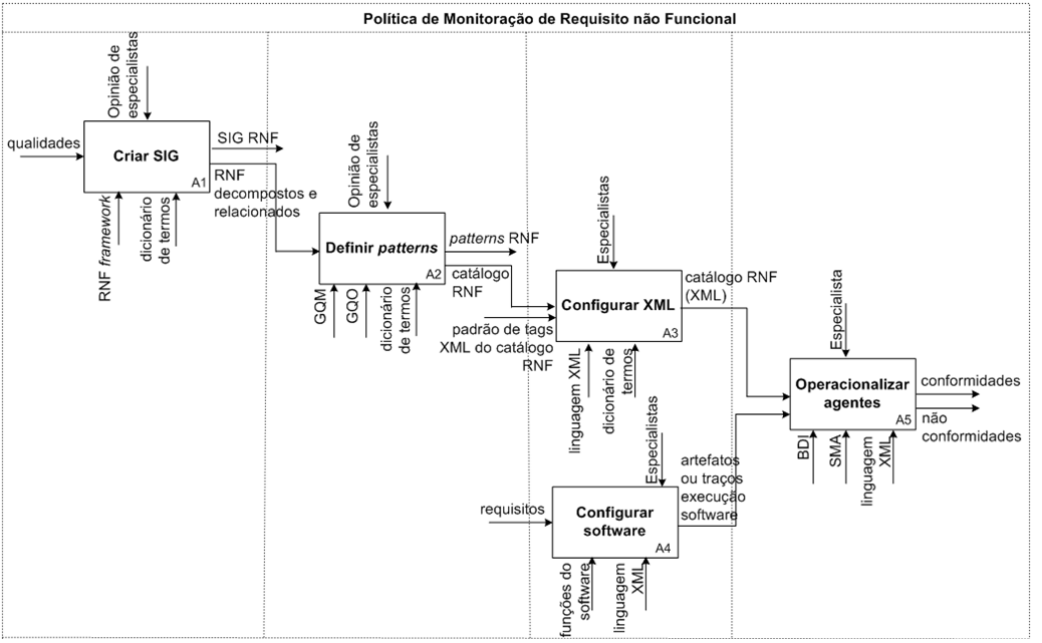

Figura 1. Visão Macro do Método.

1. Criar SIG (A1): consiste em definir de forma sistemática a decomposição de uma qualidade em outras qualidades que a caracterizem, sua hierarquização (com 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. Além do nível de satisfação (como satisfeito a contento [31], devido as características de subjetividade do atributo). Tal estrutura é baseada na proposta do NFR *framework* de [4]. 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. Em [4] é estabelecido que tais qualidades passam a ser metas a serem cumpridas, ou metas flexíveis. 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 [32]. 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) [33] para solucionar um problema, e essa satisfação (*satisfice*) 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: sugere 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 com 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 [13]. 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 [31]. A partir desses desdobramentos há a criação de um catálogo do *softgoal* a partir de uma coleção de padrões denominado RNF *patterns* (*Objetive patterns, Alternative patterns, Selection patterns, Problem Patterns* [12] e *Question Patterns* [13]). 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 [45]. 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 [2] há uma customização que envolveria o Universo de Informação (UdI) [34] 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 [35]. 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 [31]. A atividade é sub-dividida em: a) Capturar artefato: objetiva a captura o artefato do software a ser analisado; b) Analisar: tem por finalidade analisar os artefato 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-definidas no catálogo XML de RNF. O detalhamento de cada atividade do método pode ser encontrado em [36].

4 Arquiteturas Utilizadas

4.1 Modelo conceitual dos *patterns*

O modelo conceitual dos *patterns* de RNF elaborado nessa seção surge a partir da proposta dos autores: - [12] no que se refere a camadas de "Padrões Objetivos" (*Objective Patterns*), subdividido em *patterns* Identificação e *patterns* Decomposição. O primeiro para identificar o RNF objeto da análise e o segundo elemento suas decomposições. Além dos Padrões Seleção (*Selection Patterns*); - o proposto em CTS (2013) e [13], 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, adaptado de [37], 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;

A Figura 2 apresenta o relacionamento entre os elementos do *pattern* (identificação, decomposições, categorias das questões, quetões e alternativas) e também o elemento variáveis essenciais, onde estão os marcadores que serão objetos de comparação. A figura apresenta sua estrutura e também as cardinalidades de relações.

É importante perceber na estrutura que o *pattern* Identificação é o elemento principal de onde todos os outros passam a ser estruturados. Nele estará o *softgoal* principal pelo qual há a intenção de avaliar seu grau de compatibilidade com as regras configuradas a partir de *patterns*. 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 várias outras decomposições.

Essas decomposições são categorizadas para que alternativas das questões possam ser identificadas e relacionadas a essa categorias para que possam servir para avaliação dos *softgoals*. As questões por sua vez devem possuir alternativas (baseadas em boas práticas de ES, conforme tratado em seções anteriores desse trabalho) para que possam servir como opções de respostas para avaliação 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 padrão que dependerâ do domínio da aplicação.

As variáveis essenciais por sua vez devem representar marcadores passíveis de comparação com os artefatos do software. Dessa forma, a comparação entre a heurística e o artefato, bem como sua relação com aos *patterns* alternativas possibilitam a avaliação desses artefato com todas as regras definidas no método.

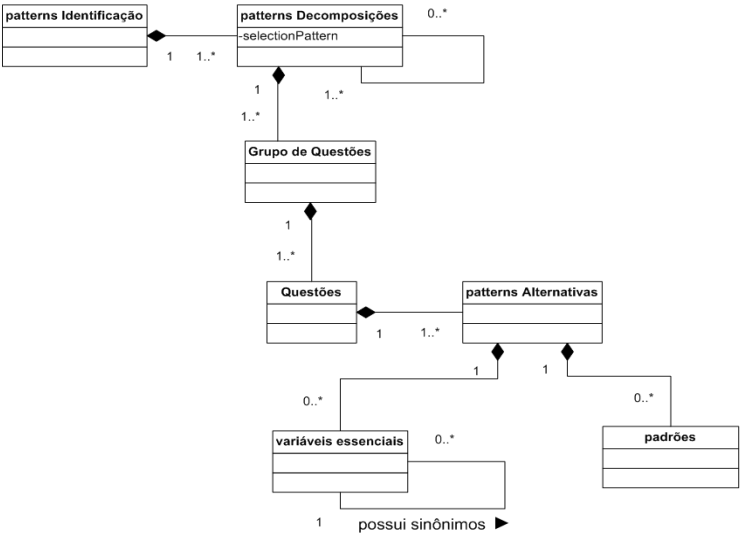

Figura 2. Modelo conceitual dos *patterns* de RNF.

4.2 XML catálogo

Após a criação do modelo conceitual foi possível iniciar a criação de uma estrutura XML do catálogo de *patterns* de RNF, uma contribuição desse trabalho. 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 estruturas criadas podem ser vista com suas descrições apresentadas na Tabela 1. Na Tabela 1 as tags tags finais são utilizadas conforme padrão XML iniciadas por "/" como por exemplo: </catalogo>, </versão>, </patternObjetivos>, mas não foram apresentadas na tabela.

Tabela 1. 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 XML catálogo. 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.

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 XML catálogo. As três tags *<idVariaveisEssenciais>*, *<nocaoVariaveisEssenciais>* e *<sinonimosVariaveisEssenciais>* tiveram como fundamento o que é utilizado em Léxico Ampliado da Linguagem (LAL) [34] 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 [34]. 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 [34]. 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 o objetivo de se obter um 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 [37] 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 2 apresenta os detalhes das classificações de noção e impacto dos símbolos de acordo com as classes.

Tabela 2. Noção e Impacto em Léxico [34].

	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 XML catálogo.

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 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. 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. Essa avaliação se dá a partir da identificação de conformidades e não conformidades. As conformidades sugerem que o artefato analisado possui algum tipo de compatibilidade com o catálogo definido, enquanto as não conformidades, indicam alguma incompatibilidade.

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 [38] [39]. 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 sugere 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.

A interação dos agentes é determinada pelo modelo intencional conforme Figura 3:

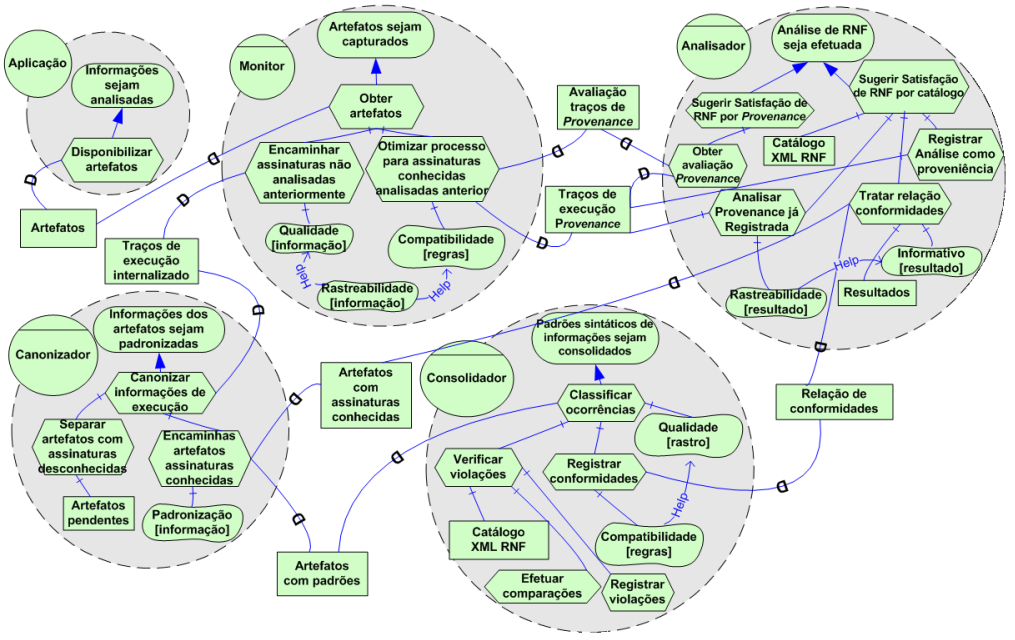

Figura 3. Modelo intencional da interação dos agentes – *framework* i*.

4.5 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.

4.6 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 verificação da qualidade do processo que o produziu [38] [39]. O dicionário Inglês Oxford (2013) [40] 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 [41] and Vasquez [42] sugerem 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 4 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 provenance para que seja avaliada e sugerida a conformidade com o RNF, uma vez que o agente MONITOR utiliza os artefatos de provenance 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.

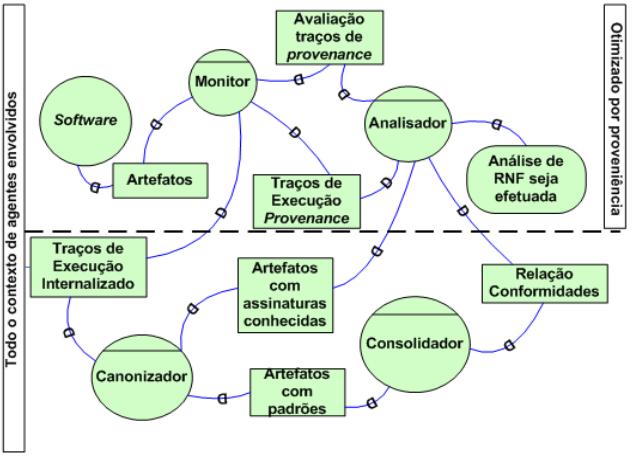

Figura 4. Visão da interação dos agentes com recorte da análise com suporte de proveniência.

O fluxo baseado em registros de provenance 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 provenance 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>estruturasAlgoritmo</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 e verificação 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.

5 Arquitetura de implementação dos agentes do SMA

A implementação dos agentes foi baseada em um mapeamento entre o modelo intencional, a arquitetura BDI e a codificação em Jadex [43], conforme proposto em [44]. A Figura 5 apresenta a proposta do mapeamento entre as diferentes camadas.

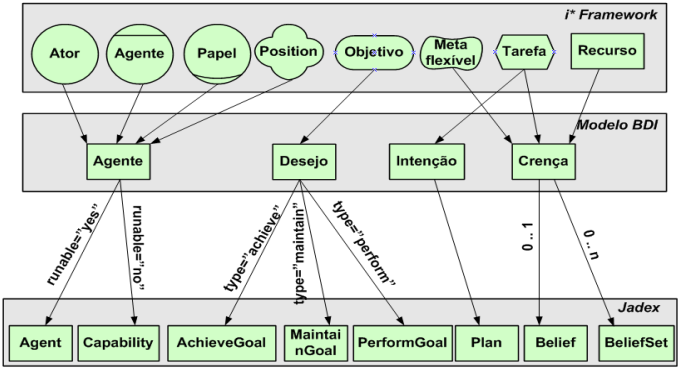

Figura 5. Mapeamento i* x BDI x Jadex [44]

A descrição textual da Figura 5 indica a estratégia de implementação dos agentes. Os agentes não executáveis originados de papéis ou posições deveriam ser implementados como capacidades; os desejos foram traduzidos como objetivos e desenvolvidos ou mantidos de acordo com a tag type do modelo BDI; as intenções foram implementadas com planos, ou seja, classes Java que estenderam os planos da plataforma Jadex [43]; 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 [44]. 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 [36] baseado na proposta de [44] 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, tal código fonte poderá ser revisto em outro momento para atender a melhores condições de arquitetura e qualidade.

A implementação doa 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 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); }    }  }
```

6 Provas de conceito

As provas de conceito do método foi elaborada a partir de sua aplicação em pelo menos quatro realidades de software. O método foi aplicado em software em execução, em código fonte, traços de execução e também em artefatos de arquitetura. Os software escolhidos foram aqueles que os pesquisadores tinham maior confiança em sua execução e na corretude de seus modelos ou artefatos gerados. Os software escolhidos foram:

- Código fonte do Software C&L: O C&L (Cenários e Léxico) [37] 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 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.

- Traços de execução do servidor Apache: Servidor Web livre compatível 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, sugere-se que o servidor tenha uma representação de cerca de 47% das plataformas instaladas em 2007 com crescimento para 55%, aproximadamente, em 2010.

- Traços de execução do LS: O LS [24] é um SMA independente para contagem de citações científicas, que combina os serviços do sítio do *Lattes* [25] e do *Google Scholar* [26]. 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.

- Artefato de arquitetura do LS: A especificação da arquitetura do LS é feito com base no iStarJade[6], que muito se assemelha ao trabalho de Baia et al. (2012) no que diz respeito ao mapeamento entre diagramas i* e a linguagem iStarML [46]. Os agentes tem sua arquitetura implementada a partir de modelos i* com apoio de um padrão de representação textual iStarML [46]. Tal representação é compatível com XML e a partir desse são criadas as estruturas da arquitetura dos agentes.

[6] www.les.inf.puc-rio.br/wiki/images/e/eb/Eduardo_2010_2.ppt

Os estudos foram feitos 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 sugere 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 (*satisfice*) [31].

Por motivo da extensão dos estudos será apresentado aqui apenas um detalhamento da prova de conceito realizado sobre os traços de execução do software LS. Maiores detalhes sobre outros estudos podem ser visto em [36].

6.1 Aplicação do Método Sobre Traços de Execução do 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 [3]. 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" [47].

A aplicação do método pretende avaliar se há no software, objeto da análise, características que satisfazem o requisito de Auditabilidade e uma das possibilidades é no atendimento de um de seus *softgoals* de hierarquia inferior, ou seja, em camada inferior no modelo que é criado. Essa possibilidade pode ser feita para o *softgoal* de Rastreabilidade.

Os dois primeiros estudos basearam-se em artefatos de software em formato ASCii e para o LS serão feitos estudos sobre o seu traço de execução a partir do padrão XML.

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 6.

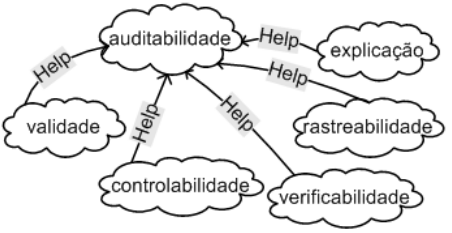

Figura 6. SIG Auditabilidade [3].

O detalhamento de Auditabilidade sugere 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 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 7 no quadro Resultado. Tais detalhamentos são propostos no CTS [47] e são definidos a partir da aplicação da A2.1 Descrever questões (descrever questões para *softgoals* folha, no caso Rastreabilidade).

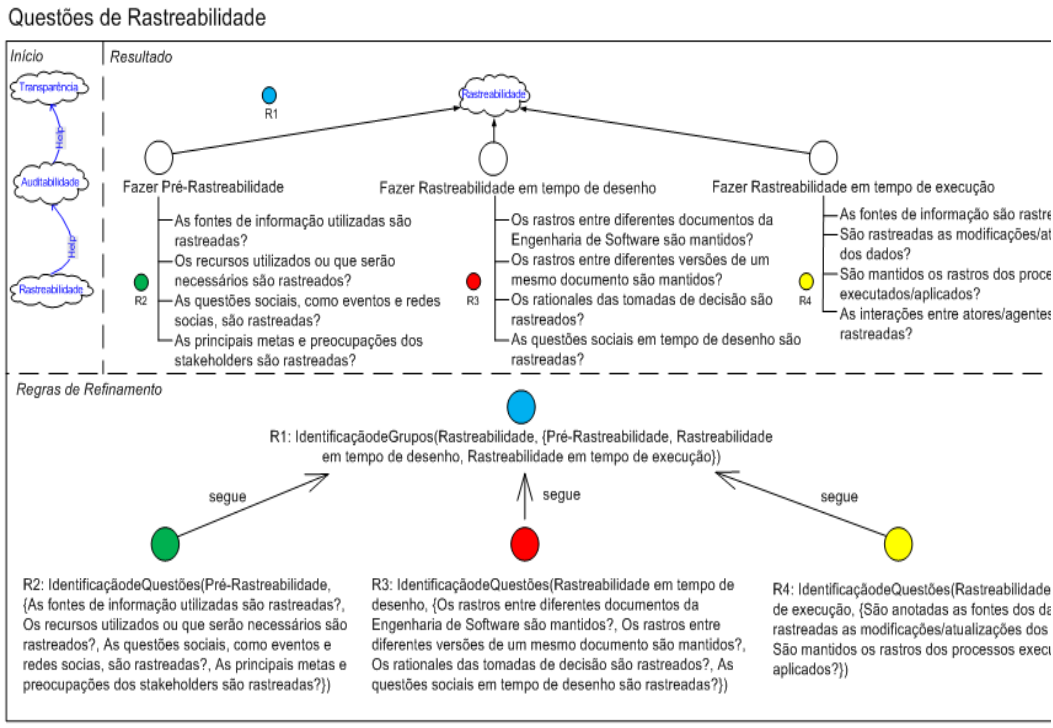

Figura 7. *Patterns* do *softgoal* Rastreabilidade [47].

A decomposição das questões a partir A partir da definição das questões sugere-se 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. As questão e operacionalização escolhidas aleatoriamente podem ser vistas na Tabela 3 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 3. 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 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 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 4. 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:	Contente
Padrão:	não se aplica

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

```
... (trecho suprimido propositadamente)
<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</idSinonimosVariaveisEssenciai
s>

<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 propositadamente)
```

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 LS são em formato ASCii e divididos em seções. Por exemplo, abaixo é apresentado uma primeira seção gerado pelo Agente1 do LS. 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
```

A seguir, ao final da primeira seção, são registrados traços de execução da ação do agente na pesquisa por obras, publicações científicas. Nesse ponto, o Agente1 inicia uma preparação de uma base de conhecimento de obras de autores para que possam ser pesquisadas em seguida o número de citações.

```
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.

Nosso foco de estudo para análise de compatibilidade com os padrões estabelecidos em catálogo de RNF está nessa segunda seção, apesar do LS também gerar outras seções com conteúdos diversos da execução de seus agentes. Para isso faz-se necessário a geração dos traços de execução do LS a partir de um 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: 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 é apresentado em arquivo XML como apresentado a seguir:

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>
```

O estudo 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 há também a necessidade da avaliação por interação humana para indicar se os conteúdos das tags correspondem com o que sugere 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.

7 Conclusão

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 teorias de GORE, RNF *framework*, GQM, GQO e RNF *patterns* que dão suporte teórico ao que é construído enquanto método sistêmico proposto no trabalho; enquanto que para a construção da aplicação e sua as arquiteturas físicas suportadas pelo SMA estão fundamentos sobre modelagem intencional, *framework* i*, painel de intencionalidades, SMA e BDI. Dessa forma, entende-se que o trabalho possuim uma prática de construção TOP-DOWN uma vez que inicia a partir de definição de modos conceituais de determinado RNF, bem como a partir de intencionalidade de agentes envolvidos no SMA, até às definições de camadas físicas de mais baixo nível da aplicação computacional. Tal abordagem

também é utilizada pela definição do catálogo utilizado, em [47], para a definição de mais alto nível de *softgoals* que satisfação a um determinado RNF analisado até o mais baixo nível, onde estão operacionalizações funcionais que satisfazem aos níveis conceituais acima. Tratar RNFs a partir de modelos GORE não é uma nova frente de pesquisa, porém a aplicação de GORE aliada a *patterns* operacionalizados por SMA para análise de satisfação é uma novidade na literatura, principalmente quando esses *patterns* compõem um catálogo que alia a modelagem conceitual às alternativas (operacionalizações) de respostas funcionais que possam satisfazer à RNFs.

Essas abordagens unidas em um método sistematizado, ou seja, que deixe claro seus passos metodológicos baseados em critérios técnicos, compõem um desafio, uma vez que é necessária o estudo e criação de uma sistemática que leve em consideração, desde a concepção de modelos conceituais, até arquiteturas físicas para implementação de uma aplicação que possa lidar com a verificação de RNF que tem por característica um elevado grau de subjetividade dependente do ponto de vista de quem o avalia. Isso foi viabilizado a partir da criação de uma estrutura (catálogo) que associasse, a partir da modelagem proposta no NFR *framework* [4] e GQO [13], metas flexíveis a operacionalizações a partir de notações XML que pudessem ser interpretadas e verificadas condições de satisfação por agentes autônomos.

Alguns trabalhos futuros são vislumbrados:

- O foco inicial da análise foi a verificação de conformidades dos registros em relação aos patterns de RNF, apesar disso o SMA permite o armazenamento de não conformidades o que pode ser explorados com a proposta do uso de sistemas de recomendação [48] [49]. 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.

- Extender o catálogo XML para que incorpore estruturas formais 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, MPS.BR ou ISO 12207. 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 [50] [51]. O catálogo XML de RNF pode compor tais normas para que regras de análise sejam postas a fim de que os agentes operem baseado em contexto pré-estabelecido a partir de uma abordagem NBDI [50]. Apesar de ter uma semelhança do que é proposto no trabalho de tese, o mesmo não formaliza tal pesquisa baseada nos pressupostos teóricos de NBDI e não trata de questões de restrição, como por exemplo poderia ser aplicado numa camada de Padrões Alternativas/Problemas, em que problemas poderiam ser tratados como regras de restrição.

Contribuição dos autores:

- André Castro: O autor participou na coleta de dados, análise bibliográfica, elaboração de modelos, definição de arquiteturas, definição do método e na aplicação prática das provas de conceito.

- Henrique Sousa: O autor participou das reuniões de pesquisa para entendimento da elaboração de algoritimos necessários para verificação automática de modelos.

- Julio Leite: O autor participou diretamente na orientação do trabalho de pesquisa.

Referências

1. Fenton, N. E.; Pfleeger, S.L.. Software Metrics: a Rigorous and Practical Approach, 2nd Ed., PWS Publishing Company. 1997.
2. Leite, J.C.S.P.; Cappelli, C.. Software Transparency. Business & Information Systems Engineering, Springer, 2010, pp 127-139.
3. 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.
4. Chung, L.; Nixon, B. A.; YU, E., and Mylopoulos, J. Non-Functional Requirements in Software Engineering. Springer, 2000.
5. Sommerville, I.. Engenharia de Software, 9ª Edição. Pearson Education, 2011.
6. Fidge, C.; Lister, A.. The Challenges of Non-Functional Computing Requirements. Seventh Australian Software Engineering Conference (ASWEC93), 1993.
7. Matoussi, A.; Laleau, R.. A survey of Non-Functional Requirements in Software Development Process, Universita Paris, Faculté des Science et Technologie, Paris 2008
8. 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.
9. Glinz, M. On Non-Functional Requirements. In 15th IEEE International Volume , Issue , 15-19 Oct, pages 21–26, 2007.

10. van Lamsweerde, A.. Goal-Oriented Requirements Engineering: A Guided Tour. 5th IEEE International Symposium on RE'01, pp. 249-262, August 2001.

11. van Lamsweerde , A.; Letier, E.. Handling Obstacles in Goal-Oriented Requirements Engineering. IEEE Trans. Software Eng., vol. 26, pp. 978-1005, 2000.

12. 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.

13. 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.

14. 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.

15. Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, W.. Object-Oriented Modeling and Design. Prentice Hall. 1991.

16. Demarco. T., Structured Analysis and System Specification. Yourdon Press. 1978.

17. Ross, D.. Structured Analysis (AS): A Language for Communicating Ideas (SADT). IEEE Transactions on Software Engineering, vol. 3, no.1, 1977, pp. 16-34.

18. Lapouchnian, A.. Goal-Oriented Requirements Engineering: An Overview of the Current Research. Technical report, Universidade de Toronto, Canadá, 2005.

19. Bratman, M. E. Intention, Plans, and Practical Reason. University of Chicago, ISBN: 1575861925, 208 pages, 1999.

20. 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.

21. 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.

22. 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.

23. 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.

24. Lattesscholar: Requirements Engineering Group at PUC-Rio. <Disponível em: http://www.er.les.inf.puc-rio.br/~wiki/index.php/Lattesscholar> <Acessado em: 12/05/2015>

25. Lattes. Plataforma Lattes: Currículo Lattes. <Disponível em: lattes.cnpq.br > <Acessado em: 12/05/2015>

26. Google Scholar. <Disponível em: www.scholar.google.com > <Acessado em: 12/05/2015>
27. Publish and Perish. <Disponível em: http://www.harzing.com/pop.htm> <Acessado em: 12/05/2015>
28. Microsoft Academic Search. <Disponível em: http://academic.research.microsoft.com> <Acessado em: 12/05/2015>
29. Google Scholar Citation <Disponível em: http://scholar.google.com/intl/en/scholar/citations.html> <Acessado em: 12/05/2015>
30. Yu, E.S.K.. Modelling Strategic Relationships For Process Reengineering. Ph.D. dissertation. Dept. of Computer Science, University of Toronto, 1995.
31. 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.
32. 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.
33. Simon H. A.. The Sciences of the Artificial. MIT Press, Cambridge, MA, USA, 3rd ed., 1996.
34. 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.
35. Wooldridge, M.. An Introduction to Multi-Agent Systems. John Wiley and Sons. 2002.
36. Leal, A. L. de C.. Análise de Conformidade de Software com Base em Catálogos de Requisitos não Funcionais: Uma Abordagem Baseada em Sistemas Multi-Agentes. Rio de Janeiro. 196 p. Tese de Doutorado – Departamento de Informática, PUC-Rio, 2014.
37. 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.
38. 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.
39. 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.
40. Oxford English Dictionary, "provenance, n". Oxford University Press.
41. Pinheiro da S.; MCGUINNESS, P.; MCCOOL, D.. Knowledge Provenance Infrastructure. IEEE Data Engineering Bulletin 26(4), 2003, pp. 26-32.
42. Vasquez I.; Gomadam K.; Patterson S.. Framework for representing provenance for web services and processes. Technical Report, LSDIS Lab, 2005.
43. 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.

44. 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.

45. 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.

46. Cares, C.; Franch, X.; Perini, A.; Susi, A.. iStarML: The i* Mark-up Language: References Guide. Barcelona, Spain, August 2007.

47. 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: 17/05/2015>.

48. 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.

49. Burke, R.. Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction, Vol. 12(4), 2002, pp. 331-370.

50. 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.

51. Camino, A. G.; Normative regulation of open multi-agent systems, Ph.D. dissertation, Artificial Intelligence Research Institute (IIIA), Spain, 2009.