NDR-Tool: Uma Ferramenta de Apoio ao Reuso de Conhecimento em Requisitos Não Funcionais

Alex Lins de Araújo, Luiz Marcio Cysneiros, Vera Maria B. Werneck

Universidade do Estado do Rio de Janeiro (UERJ), Rio de Janeiro, Brasil

alex@karyon.com.br, cysneiros@yorku.ca,vera@ime.uerj.br

Abstract. Non-functional requirements (NFR) are fundamental for the software development. The NFR- Framework allows the elicitation to deeply cover necessary trade-offs involving synergetic and conflicting solutions. It also favor the capture of design decisions involving the reasons that lead one to choose between one alternative and another to implement a NFR. This work proposes and describes the NDR-Tool, a tool that supports the software engineer in the requirements elicitation and modelling process. This tool proposes the use of ontologies and web semantics techniques to facilitate storing and retrieving knowledge related to possible alternatives to implement NFR. The tool NDR-Tool was developed and integrated with a tool for modeling NFR diagrams. As proof of concept we used the tool in the medical field.

Keywords: NFR-Framework, Ontology, NDR, NDR-Tool, NFR.

1 Introdução

Apesar da importância dos requisitos não funcionais (NFR-*Non-functional requirements*), muitos métodos de desenvolvimento de software [2, 3, 4] não tratam nem têm uma análise detalhada dos NFRs incorporadas aos modelos na fase de requisitos.

Para o acompanhamento dos NFRs, duas abordagens distintas podem ser utilizadas: orientada ao produto e orientada ao processo. Na abordagem orientada ao processo, as decisões de projeto (*Design Rationale*) são fundamentadas durante o processo de desenvolvimento do produto. Estas decisões podem afetar positivamente ou negativamente outros NFRs ou mesmo outros requisitos funcionais.

Alguns frameworks considerando NFRs já foram propostos, mas segundo CHUNG e LEITE [3], a abordagem orientada a metas tem sido a que trata de NFRs de forma mais abrangente. Assim, uma alternativa na modelagem e análise específica de requisitos não funcionais é a utilização do NFR Framework [2].

CHUNG et al [2] e posteriormente YU e CYSNEIROS [22], definem catálogos de conhecimentos armazenando condições de satisfação de NFRs, sendo que em YU e CYSNEIROS [22] existe a preocupação em registrar o conhecimento derivado de projetos passados. Esses catálogos são disponibilizados usando uma notação gráfica

baseada em SIGs (*Softgoal Interdependency Graphs*) conforme proposto por CHUNG [2].

Para ser possível o reuso desses catálogos, LÓPEZ et al [10] definiu a ontologia NDR estruturando as informações do framework de NFR representados por SIGs com o objetivo de permitir consultas ao modelo através de ferramentas automatizadas. Desta forma é possível, a partir de um SIG, efetuar uma transformação, manual ou automatizada, gerando a representação NDR do modelo. Assim cada SIG pode ser definido como uma instância na ontologia NDR, e é possível a realização de pesquisas e comparações entre diferentes projetos sistematizando o processo de *design rationale* de SIGs [11].

No entanto, não existe um processo automático de transformação de um modelo SIG em uma instância NDR, e nesse processo de consulta é necessário um conhecimento das ferramentas de especificação e utilização de ontologias.

Além disso, há uma necessidade quanto à reutilização dos catálogos, tanto de projetos como de domínio, o que é de suma importância no processo de levantamento dos requisitos não funcionais e por consequência na avaliação dos impactos, que eventualmente levem a satisfação desses NFRs, sobre os requisitos funcionais. É bastante desejável que o desenvolvedor tenha acesso a ferramentas que possibilitem o acesso simples e rápido a estas informações para que torne efetivo o uso da ontologia durante todo o processo de desenvolvimento.

Dentro deste contexto, este trabalho teve como objetivo apoiar a definição e análise de requisitos não funcionais através do desenvolvimento da ferramenta NDR-Tool, proposta em ARAÚJO [1]. Esta através de um processo automático transforma catálogos NFRs de projetos e de domínios em instâncias da ontologia NDR. A ferramenta permite também realizar consultas de forma relativamente simples sobre a semântica da ontologia que retrata o conhecimento sobre os vários NFRs.

Este trabalho está dividido em cinco seções. Na seção 2 é apresentada a ontologia NDR, uma ontologia de NFRs para SIGs na qual foi fundamentada a proposta da ferramenta NDR-Tool. Na seção 3 é descrita a proposta de ferramenta NDR-Tool, sua modelagem e o protótipo inicial. A seção 4 mostra a utilização da NDR-Tool através de um exemplo de uso da ferramenta em uma modelagem de software da área médica. Finalmente na seção 5 são apresentadas as considerações finais do trabalho e as sugestões de trabalhos futuros.

2 Ontologia NDR

LÓPEZ et al [10] apresentam uma ontologia para descrever NFRs e *design rationale* chamada NDR. Através dela é possível representar a semântica de um SIG proveniente dos catálogos ou de soluções específicas para um domínio, formatando-o para um padrão legível por dispositivos de forma automática, permitindo a pesquisa, compartilhamento e reuso tanto de catálogos quanto de *design rationale*.

A ontologia NDR importa a ontologia NFR de SANCHO et al [15], tratando os conceitos de SIG, de forma semelhante, mas implementando alguns conceitos como as interdependências do tipo *correlation* e definindo propriedades para as classes

filhas de *refinement*. Cada classe da ontologia representa um conceito presente no SIG e as propriedades relacionadas a estas classes representam as interdependências entre as *softgoals*, *claims* e *argumentations,* os quais representam o *design rationale*, também são representados por classes.

A figura 1 mostra a ontologia NDR [11] referente aos conceitos relacionados a *softgoals*. A classe nfr:NFR_Type foi importada da ontologia NFR de SANCHO et al [15].

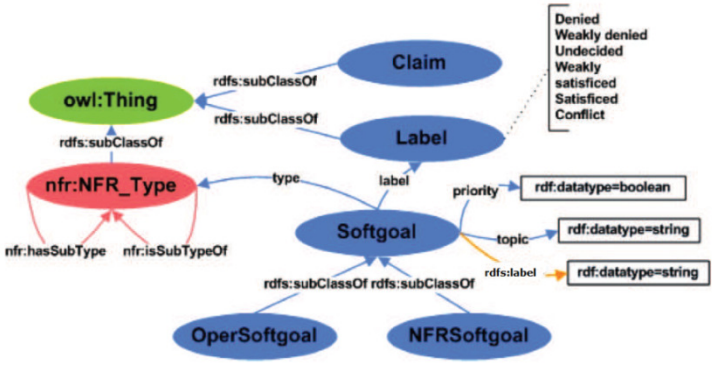

Fig. 1. *Softgoal-related Concepts* [10]

A classe nfr:Softgoal, foi especializada a partir de nfr:NFR_Type e definida três propriedades: priority, topic e rdfs:label. Priority informa o grau de relevância da softgoal, topic descreve o contexto no qual a softgoal se apresenta. rdfs:label contém o nome da *softgoal*. Se o nível de satisfação da *softgoal* já foi definido, associa-se a *softgoal* a uma classe *Label*. Esta classe nada mais é do que uma lista enumerada das possibilidades de satisfação da softgoal: *Denied, Weakly_denied, Weakly, Weakly_satisficed, satisficed, Undecided* e *Conflict*. Na ontologia, os tipos de *softgoal* (NFR, *Operationalizing* e *Claim*) são representadas respectivamente pelas classes NFRSofgoal, OperSofgoal e Claim. Para realçar a diferença entre uma claim softgoal e as demais, LÓPEZ et al [11] optaram por não relacionar esta classe à classe *softgoal*.

A figura 2 mostra como as interdependências são representadas na ontologia. A representação da relação entre duas *softgoals* foi feita através do conceito *head-tail* (cabeça-cauda). Desta forma, pode-se representar a relação pai-filho entre as *softgoals*. Assim, foi definida a classe Interdependency que tem duas subclasses: ImplicitInterdependency e ExplicitInterdependency.

Se a interdependência é explícita, ela pode ser uma decomposição, representada pela classe Decomposition ou uma operacionalização, representada pela classe Operationalization. A decomposição pode ainda ser dividida em uma decomposição operacional, representada pela classe OperDecomposition, ou uma decomposição de NFR através da classe NFRDecomposition. Ambas se relacionam através de uma dupla relação com suas respectivas softgoals, OperSoftgoal e NFRSoftgoal.

Para representar o grau de contribuição entre duas *softgoals*, é utilizada a classe Contribution. Essa classe é uma enumeração com as seguintes opções: *Break, Hurt, Unknown, Help, Make, Some-, Some+, And, Or* e *Equal*.

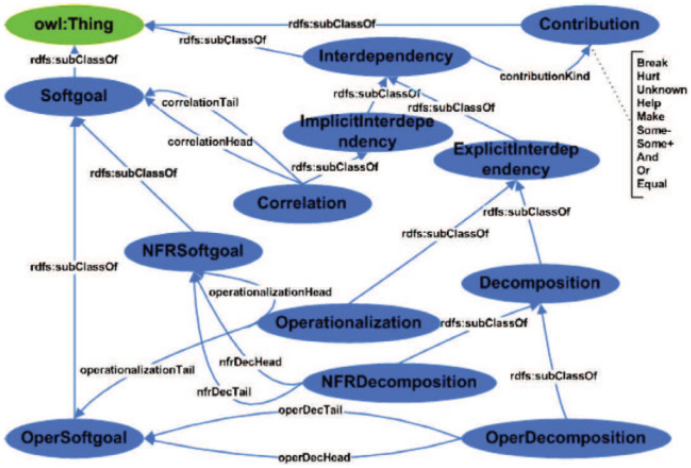

Fig. 2. *Interdependency e decomposition Concepts* [11]

A parte da ontologia referente às decisões de projeto é mostrada na figura 3. Para distinguir as interdependências que envolvem *claims softgoals* das demais a classe *Argumentation* foi criada sem uma relação com a classe Interdependency. As classes ArgSoftgoal e Arginterdependency representam a argumentação sobre uma *softgoal* ou sobre uma interdependência. Neste caso a classe Contribution irá informar o grau de contribuição da argumentação.

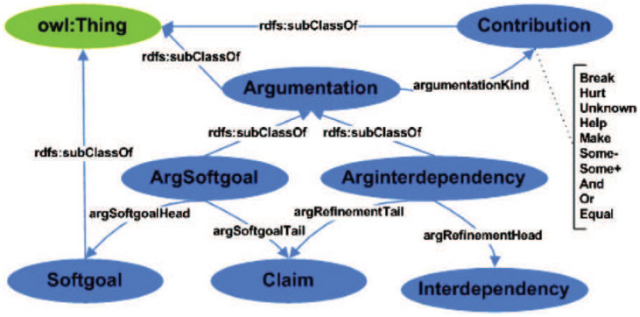

Fig. 3. - Argumentation-related Concepts [11]

A ontologia NDR permite, a partir de qualquer modelo SIG, executar uma transformação manual, caso exista apenas o desenho do modelo, ou automatizada, a partir do XML que representa o modelo, e armazenar as informações em um formato semântico, permitindo a construção de consultas através de ferramentas de forma automatizada.

Named Graph (NG) é uma extensão baseada em RDF que permite identificar cada RDF como um URI. Estes possibilitam a inclusão de metadados em gráficos RDF, garantindo a procedência e confiabilidade destes. Por outro lado, os Catálogos NFR gerenciam muitos SIGs que lidam com *softgoals* e interdependências semelhantes. Quando o desenvolvedor utiliza algo do catálogo, mesmo após a edição do modelo, não é possível diferenciar um *softgoal* editado de um do catálogo. Dentro do modelo NDR, a utilização de NGs permitem gerenciar instâncias diferentes através de diferentes URIs, fazendo com que cada instância evolua de forma independente, proporcionando um uso semântico do modelo. Por outro lado se a informação deve ser compartilhada entre os diversos modelos, ela é armazenada através de um NFR_Type, representado por um único URI para todos os SIGs.

3 Definição da NDR-Tool

O objetivo da NDR-Tool é auxiliar o engenheiro de software na definição de NFRs através do reuso de modelos. Para apoiar esse processo, foi utilizada a ontologia NDR. Para tornar esse processo automático, o requisito principal foi que a consulta e as implicações de agregação de novos NFRs fossem realizadas de forma simples sem necessidade de conhecimento da ontologia NDR.

Partindo dessas premissas o processo de desenvolvimento da ferramenta NDR-Tool foi realizado através de cinco fases distintas: (i) estudo das tecnologias a serem utilizadas na ferramenta, (ii) definição dos requisitos, (iii) modelagem da ferramenta, (iv) construção do software e (v) teste de uso da ferramenta através de modelos em um estudo de caso.

No estudo da tecnologia a ser utilizada na NDR-Tool foram pesquisados softwares e bibliotecas que fornecessem suporte a estruturas XML, RDF, e principalmente ao suporte de ontologias escritas em OWL, tanto na especificação quanto na utilização das mesmas, através de consultas as estruturas criadas. A ontologia NDR proposta por LÓPEZ et al [11] estava disponível apenas a nível conceitual e assim foi necessário reproduzi-la baseando-se nos artigos e especificando-a em OWL. Na fase de definição de requisitos, um catálogo com os léxicos da ferramenta foram registrados e a definição dos requisitos foi realizada a partir de um protótipo da ferramenta. Na terceira fase foi realizada a modelagem do sistema, utilizando-se o framework i-star que é uma abordagem orientada a metas. Na quarta fase foi realizada a construção do software e, na última fase, o teste de uso da ferramenta baseando-se em um exemplo de estudo de caso na área médica e nos catálogos dos NFRs de Usabilidade e Rastreabilidade.

Para construção da ferramenta NDR-Tool foram procuradas tecnologias que suportassem o uso de OWL, RDF [18] e SPARQL. Após o estudo de linguagens e frameworks que permitissem integrar em um único ambiente essas tecnologias com uma linguagem de programação, optou-se pela linguagem Java por estar bastante difundida no meio acadêmico e, principalmente, disponibilizar frameworks gratuitos que permitem o uso nativo de OWL.

Para o ambiente de desenvolvimento, foi utilizado como IDE (Integrated Development Environment) a plataforma Eclipse [8]. Posteriormente, o projeto foi migrado para a plataforma NetBeans 7.2 [12], devido a facilidade com que a IDE trabalha com a biblioteca de interface gráfica Swing, o que facilita o desenvolvedor na construção da interface com o usuário.

Para tratar a questão da manipulação da ontologia, foi utilizado o framework Apache Jena [9]. Ele possui um conjunto de bibliotecas que permite a construção de aplicações que utilizem Web semântica, possibilitando o tratamento de estruturas RDF e em particular ontologias escritas em OWL.

Como foi necessário descrever a ontologia NDR em OWL, foi utilizado o software Protégé [14]. Este é um software de livre distribuição (open source) que disponibiliza um conjunto de ferramentas para especificação de ontologias. Especificamente o Protégé-owl Editor é uma extensão que permite a edição de ontologias especificadas em OWL.

Para a modelagem dos SIGs optou-se inicialmente pela ferramenta de apoio a modelagem OpenOME - Organization Modelling Environment [13] (*Knowledge Management* Lab, Universidade de Toronto). Esta ferramenta suporta a análise e modelagens diversas, tais como i-star, mas a parte de NFR não permite instanciar *Operationalization*. Por isso foi avaliado o conjunto de softwares StarUML e RE-Tools, que juntos, permitem a modelagem de SIGs seguindo os conceitos de CHUNG [2].

O software StarUML [16] é um projeto open source que disponibiliza uma plataforma para o desenvolvimento de modelos UML/MDA e executa sobre o ambiente Windows. Por ser flexível, permite a expansão através de plugins (chamados Add-Ins) que contemplem outras modelagens. RE-Tools [17] é um conjunto de bibliotecas open source que funcionam como plugins que podem ser instalados no software StarUML [16], alterando a funcionalidade do software para permitir o desenho de modelos i*, Kaos, Problem Frames e especificamente o NFR Framework.

No processo de modelagem da NDR-Tool foi adotado o framework orientado a metas i* (i-star) [20, 21]. Após o levantamento dos requisitos e a construção do léxico foi feita a modelagem do sistema. Na figura 4 é apresentado o modelo SD (Strategic Dependency) relativo à ferramenta NDR-Tool. No modelo proposto, o engenheiro de software cria ou escolhe o modelo SIG através da ferramenta RE-Tools. Como os SIGs estão persistidos pelo RE-Tools, um agente *Converter*, foi projetado e desenvolvido para ler o modelo SIG e criando os respectivos indivíduos na ontologia NDR. O engenheiro de software pode utilizar a ferramenta NDR-Tool como apoio a modelagem, solicitando a pesquisa de termos nos catálogos existentes. A NDR-Tool, através do *NDR-Analizer*, localiza referências ao NFR procurado na ontologia NDR, seja ela uma correlação, um refinamento ou uma contribuição.

As estratégias internas da ferramenta NDR-Tool têm como principal meta fornecer apoio à modelagem de SIGs. As tarefas que ajudam a atingir essa meta são: procurar nos modelos os NFRs requisitados pelo engenheiro de software, mostrar as interdependências encontradas na ontologia e possibilitar a importação de elementos para o modelo de trabalho. Além destas, uma meta complementar encontrada foi disponibilizar a NDR-Tool através de um plugin incorporado ao NFR-Editor.Durante a análise, NDR-Tool teve uma segunda meta: ser capaz de realizar pesquisas em tempo real.

Apesar de esta meta contribuir negativamente para a softgoal *performance*, ela ajuda na questão da disponibilidade. Para mitigar a questão do desempenho, a ferramenta pode persistir a ontologia NDR, otimizando o tempo de busca.

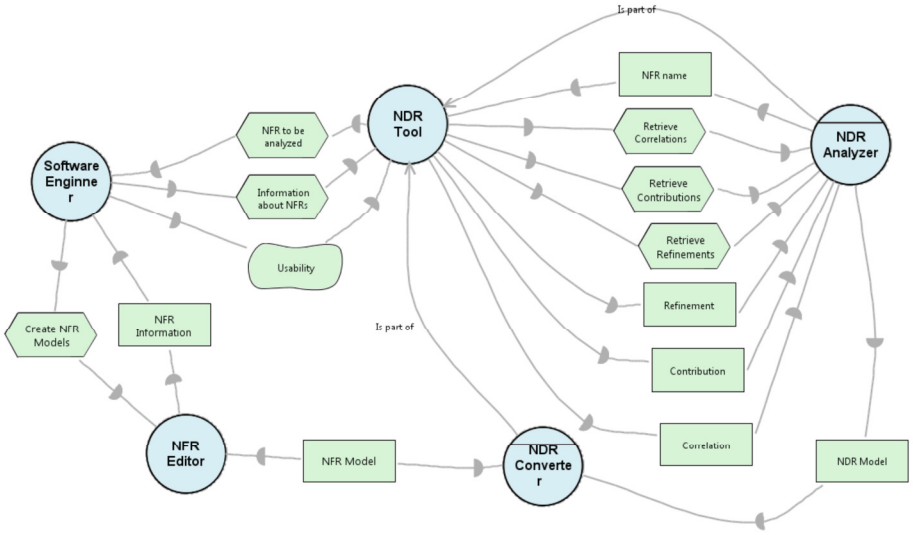

Fig. 4. – Modelo SD, NDR-Tool [1]

4 Construção da NDR-Tool

O funcionamento do ambiente de software pode ser visto na figura 5. Os diagramas SIG são criados pelo engenheiro de software na ferramenta Star UML/RE-Tool Designer [17]. Automaticamente, o módulo RE-Tools Interface carrega os diagramas do repositório, e o NDR-Tool *Preprocessing* cria os Individuos na ontologia NDR.

O engenheiro de software consulta a base de catálogos e projetos, interagindo com o módulo NDR-Tool Window. Cada solicitação do usuário é repassada ao módulo NDR-Tool *Inquirer* que é responsável pela consulta à ontologia, retornando os dados em uma estrutura que permite ao módulo NDR-Tool Window mostrá-los ao usuário.

Apesar de descrita em LÓPEZ et al [15], a ontologia NDR não foi disponibilizada na internet. Desta forma foi necessário reescrever a ontologia baseando-se nas figuras e descrições realizadas por LÓPEZ et al [15]. Para a escrita da mesma, foi utilizado o software editor de ontologias Protégé versão 4.1. Ele é compatível com a versão 2.0 da OWL.

Durante o desenvolvimento do software, o framework JENA, versão 2.7.3, teve um papel bastante importante, pois ele permite o acesso estruturado à ontologia, através de uma série de objetos. Para gerenciamento da carga em memória e acesso a ontologia são utilizadas algumas destas classes. JENA foi concebido inicialmente para aces-

so ao formato RDF e posteriormente estendido para uso com o formato OWL. Essas classes permitem abstrair a leitura do XML relativo a descrição da ontologia.

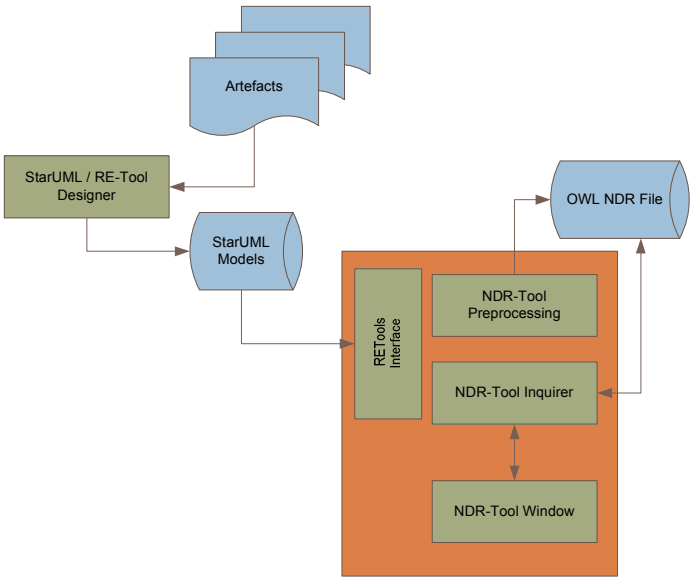

Fig. 5. – Interação entre os módulos do sistema e o ambiente [1]

Para a leitura e carga dos arquivos de catálogos e projetos construídos no Star UML/RE Tools, foi utilizada a API disponibilizada pelo software. O acesso é realizado via COM objects dentro de um padrão MDA (Model Driven Architecture).

5 Estudo de Caso da Ferramenta NDR-Tool

Um estudo de caso utilizando a ferramenta NDR-Tool foi realizado através de uma simulação baseada no caso descrito e construído por CYSNEIROS et al [4], [6], [18] em que é elaborada a definição de requisitos de um sistema de controle de cirurgias.

A priori, o engenheiro de software começa o desenvolvimento do sistema definindo os requisitos não funcionais do sistema de controle de cirurgias através da ferramenta StarUML/RE-Tools [17]. Durante esta fase, o engenheiro de software utiliza a NDR-Tool, para realizar buscas nos catálogos e integrar os requisitos encontrados à modelagem atual e, assim, incluir novos NFRs à modelagem com conhecimento das implicações que estas podem acarretar.

Antes de descrever o estudo de caso, é mostrado como efetuar as configurações iniciais no StarUML e também como realizar a integração entre a ferramenta NDR-Tool e o StarUML/RE-Tools.

5.1 Integrando a NDR-Tool ao StarUML

Para que um novo projeto permita a utilização de diagramas SIG utilizando o Star UML/RE-Tools [17], devem ser realizadas algumas configurações inicialmente no StarUML/RE-Tools, criando um novo projeto e depois deve ser configurada a habilitação do plugin RE-Tool no projeto para utilização de diagramas do NFR Framework.

A o plugin RE-Tools não permite diferenciar um *refinement* de um *correlation*. Desta forma o engenheiro de software deve por padrão sempre que quiser representar um relacionamento do tipo *correlation*, após criar o relacionamento na ferramenta com a contribuição correspondente (hurt, help, some-, some+, etc), adicionar uma *constraint* com nome "*Correlation*".

Para permitir uma melhor integração entre a ferramenta de modelagem e a NDR-Tool, foi disponibilizado dentro do StarUML, a opção de menu Tool>>NDR-Tool (figura 6).

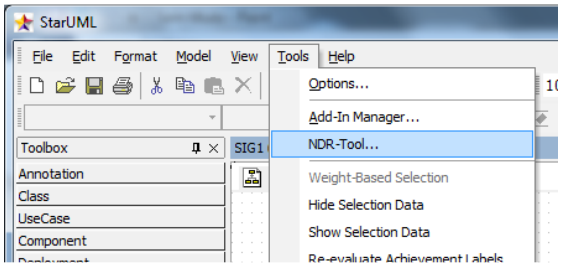

Fig. 6. – Integração entre a NDR-Tool e o StarUML

Ao selecionar esta opção, o StarUML inicia a ferramenta NDR-Tool, com a tela de busca da ferramenta (figura 10). Neste momento a ferramenta NDR-Tool transforma o modelo aberto na instância da ontologia e todos os modelos configurados na pasta de repositórios de catálogos de NFRs. Após isso o usuário pode solicitar um termo a ser encontrado no repositório da ontologia NDR e começar a aprimorar seu modelo.

5.2 Sistema de Controle de Cirurgias

Para demonstrar o uso da NDR-Tool e verificar a validade da ferramenta proposta, foi realizada uma simulação baseada no estudo de caso construído por CYSNEIROS et al [4], [6], 18]. O artigo apresenta um experimento controlado onde é avaliado se a utilização de catálogos pode ajudar ou não a elicitação de NFRs.

CYSNEIROS et al [4], [6], 18] faz uso dos catálogos dentro de uma abordagem sistemática, mostrada na figura 7. Assim, inicialmente é feita a construção de um modelo funcional e a seguir é realizado um aprimoramento do modelo, valendo-se da busca de NFRs necessários nos catálogos. A seguir, as alternativas de operacionalização relevantes são importadas para o modelo e avaliadas pelo engenheiro de requisitos.

Para elucidar essa abordagem, CYSNEIROS et al [4], [6], [18] faz um estudo de caso modelando um sistema de cirurgias no Hospital Universitário Pedro Ernesto

(HUPE) no Rio de Janeiro, Brasil. Este mesmo sistema é remodelado neste trabalho apresentando um Estudo de Caso da Ferramenta NDR-Tool.

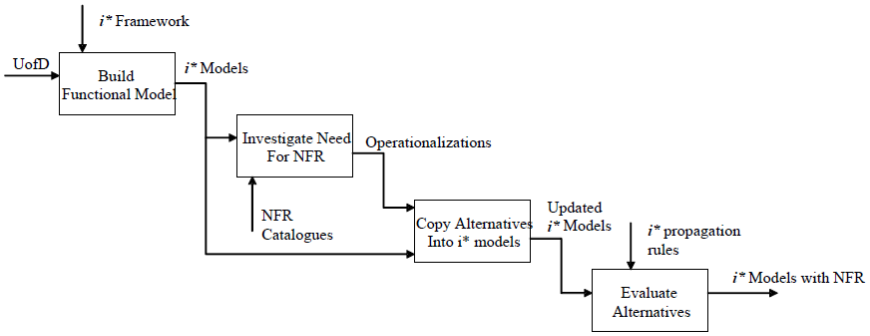

Fig. 7. Abordagem do uso de catálogos para elicitação de NFRs (CYSNEIROS, 2007)

O objetivo do sistema de controle de cirurgias é planejar e agendar as cirurgias no HUPE permitindo aperfeiçoar a administração das cirurgias e viabilizar um melhor tratamento ao paciente. O paciente é enviado ao setor de cirurgia depois de uma avaliação clínica no próprio HUPE ou em outro hospital da rede pública. O chefe da cirurgia deseja que o sistema tenha características como ajudar o cirurgião a gerenciar seu tempo, como por exemplo agendar suas cirurgias (com exceção das cirurgias de alta complexidade). Também poderia permitir o acesso ao sistema de sua casa ou de seu consultório. Em determinadas situações poderia avisá-lo sobre cirurgias de emergência. O sistema deve lidar com os vários perfis do médico: alguns têm dispositivos móveis, outros têm apenas computador em casa, e alguns nunca usaram um computador. O sistema deve permitir que o paciente informe sua preferência quanto a dia da semana e horário, além de acessar o sistema de fora do hospital, confirmando a data da cirurgia. Além do agendamento, o paciente pode ter acesso a prescrições e restrições nutricionais.

Inicialmente os catálogos de Usabilidade (especificamente o de *usefulness*) e de rastreabilidade foram implementados, utilizando o software StarUML/RE-Tools. Estes catálogos (disponibilizados por CYSNEIROS em [5] foram armazenados na pasta de repositório da NDR-Tool para que fossem incorporados na ontologia NDR. Assim, serviram de base de conhecimento para que a NDR-Tool pudesse realizar as consultas.

O catálogo *Rastreability* está em NFR Framework. Já o catálogo Usability/Usefulness foi adaptado para SIG, pois originalmente foi construído utilizando a metodologia i-star [20]. O modelo diagrama SIG em i-star pode ser visto na figura 8 mostrando os modelos construídos com a ferramenta StarUML/RE-Tools.

Após a criação dos catálogos de NFRs, o passo seguinte foi iniciar a modelagem do sistema. A figura 9 mostra parte da modelagem dos requisitos funcionais do Sistema de Controle de Cirurgias. Inicialmente, o foco é a localização de requisitos funcionais, pois são de mais fácil elicitação pelo engenheiro de software. No entanto, o engenheiro de requisitos já mapeia alguns requisitos não funcionais que encontrar durante a análise.

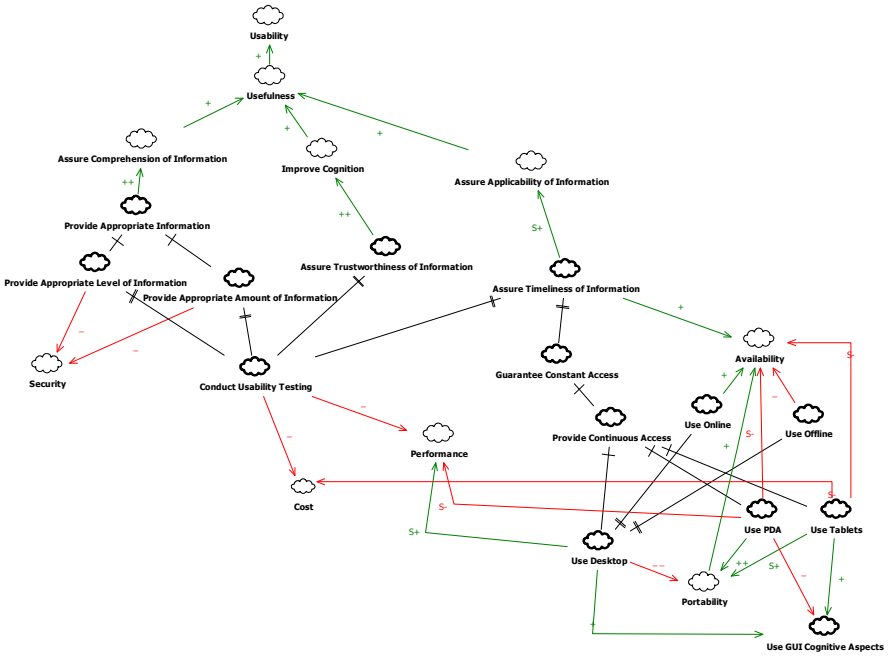

Fig. 8. – O SIG Usability/Usefulness no StarUML

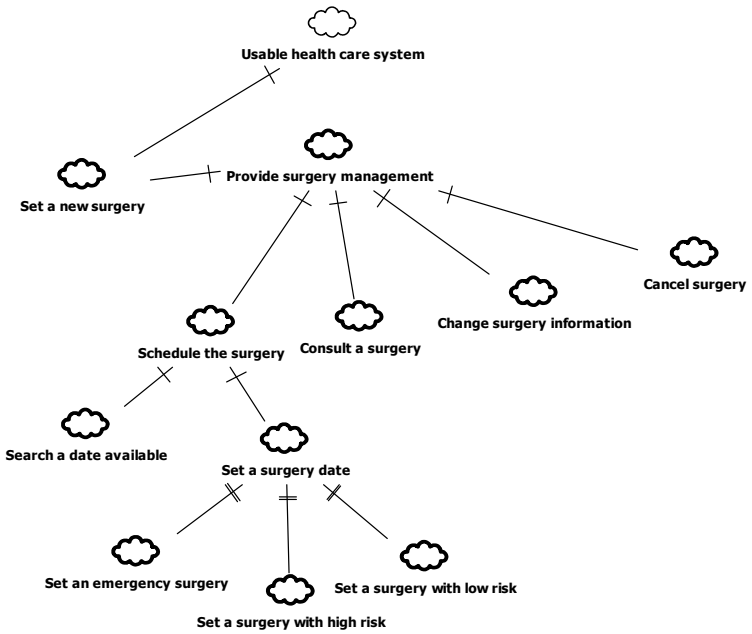

Fig. 9. – Levantamento dos requisitos funcionais

A ferramenta NDR-Tool permite filtrar os resultados, trazendo ou não *correlations, refinaments* ou *contributions*. No caso de filtrar *contributions*, o usuário deverá informar o tipo de *contribution* que está sendo localizada (*hurt, help, some-, some+,* etc).

Seguindo o processo de abordagem, procurou-se investigar possíveis NFRs que pudessem ser incorporados ao diagrama de projeto.

Através da NDR-Tool, é possível localizar requisitos provenientes do repositório de catálogos e projetos. A figura 10 mostra uma busca onde o engenheiro de requisitos procurou por requisitos que iniciam por "Usa". Ao informar o termo a ser encontrado e clicado no botão de pesquisar, do lado esquerdo da tela aparece uma lista com todos os *softgoals* encontrados e, se informado, qual o tópico relativo àquele *softgoal*.

A lista de relacionamentos obedece ao filtro informado junto ao termo de busca. Assim, o usuário pode selecionar se quer visualizar relacionamentos do *tipo refinement* ou *correlation*. Além disso, pode filtrar também pelo tipo de contribuição. Ao clicar no botão de reticências ao lado de "*All Contributions*", pode reduzir a lista de relacionamentos à *contribution* selecionada (*hurt, help, some-, some+,* etc).

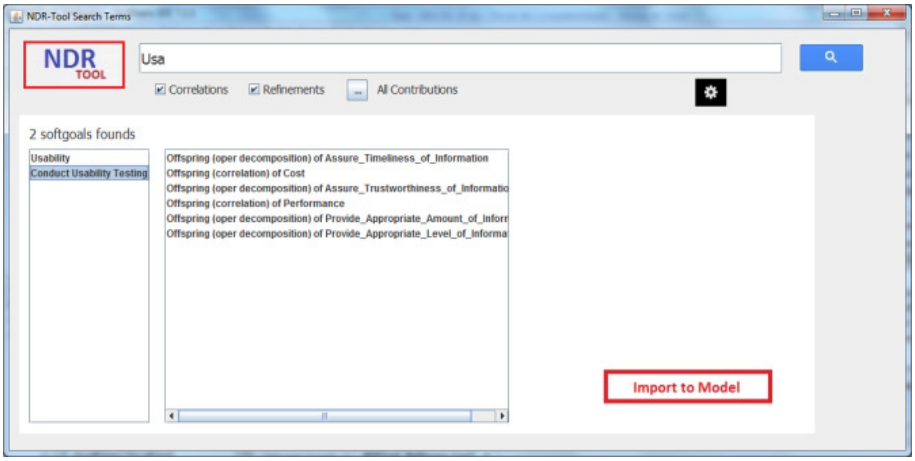

Fig. 10. – NDR-Tool: tela de busca

Selecionando o *softgoal* de interesse, é mostrado na lista do lado direito da janela, todos os relacionamentos que levam a este *softgoal* (*offspring*) ou que iniciem a partir deste *softgoal* (*parent*).

Após visualizar a lista de requisitos, o engenheiro de software pode importar os NFRs que tenha interesse para o SIG do projeto. Assim ele seleciona o conjunto de *softgoals* que tem interesse em importar e, a seguir, clicando no botão "*Import to Model*", os elementos são importados para o diagrama SIG que estiver ativo no StarUML, como mostra a figura 11 em relação ao RFN *Usefulness*.

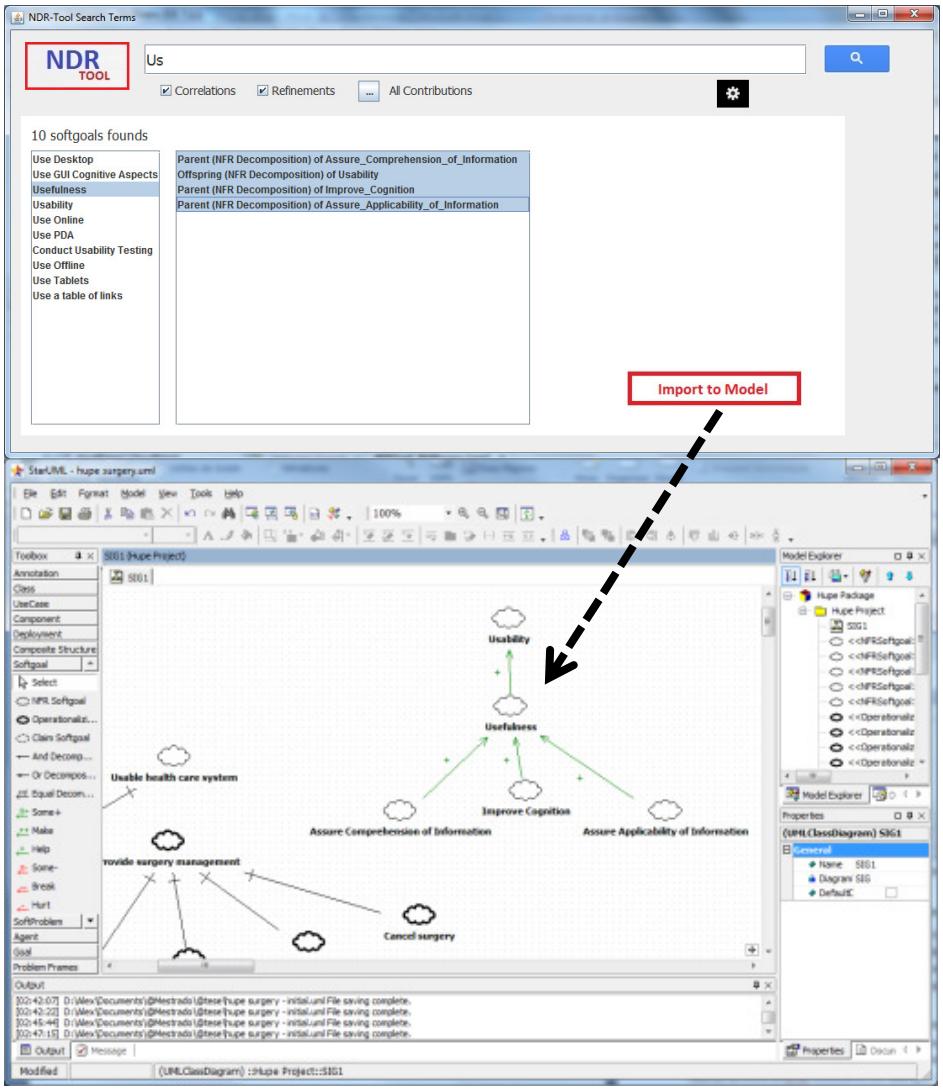

Fig. 11. – Carregando o modelo de trabalho com *softgoals*

6 Considerações Finais e Trabalhos Futuros

Este trabalho propôs uma ferramenta para apoiar a estruturação de NFRs, descrevendo os catálogos de NFRs e de projetos automaticamente na ontologia NDR. Desta forma, é disponibilizado para o engenheiro de software uma consulta amigável à esse conhecimento já na definição dos requisitos.

O engenheiro de software pode a qualquer momento durante o processo de desenvolvimento de software, pesquisar e/ou importar NFRs e seus respectivos relacionamentos para o modelo SIG do sistema que está analisando. Através da execução de um processo de transformação em diagramas SIG salvos, através da ferramenta Star UML/RE-Tools, são criados *individuals* na ontologia NDR, que possibilitam posteriormente a realização de pesquisas.

Este trabalho contribui no sentido de viabilizar a utilização de ontologias na representação de modelos SIG, especificamente a ontologia NDR. Apesar de LÓPEZ [15] realizar a pesquisa à ontologia através de SPARQL, o custo do seu aprendizado e o trabalho para realizá-las, inviabiliza a sua utilização na prática. A ferramenta NDR-Tool, mostra que é possível abstrair ao engenheiro de software a questão da utilização do SPARQL.

A funcionalidade principal da ferramenta foi totalmente desenvolvida, entretanto a necessidade de uma avaliação mais criteriosa quanto ao uso da ferramenta como apoio ao engenheiro de software deve ser avaliada por especialistas. A construção de um estudo de caso poderá aferir mais claramente a utilidade da ferramenta, ou o uso dessas, em disciplinas de Engenharia de Software como apoio a modelagem de NFRs.

Em relação a ontologia NDR, a mesma poderia ser estendida para contemplar algumas informações do SIG presentes no editor de diagramas (no exemplo, o StarUML), importantes na visualização e na incorporação dos modelos.

Referências

1. ARAÚJO, Alex Lins de. NDR-Tool: Uma Ferramenta de Apoio ao Reuso de Conhecimento em Requisitos Não Funcionais.. Dissertação Mestrado em Ciências Computacionais, Universidade do Estado do Rio de Janeiro, 2013.
2. CHUNG, L. et al. Non-Functional requirements in software engineering. Kluwer Academic Publishers. 2000.
3. CHUNG L. and Leite, J.C.S.P. . On Non-Functional Requirements in Software Engineering. A.T. Borgida et al. (Eds.): Mylopoulos Festschrift, LNCS 5600, pp. 363–379, 2009. Springer-Verlag Berlin Heidelberg 2009.
4. CYSNEIROS, L. M. and Leite, J.C.S.P. . “Nonfunctional Requirements: From Elicitation to Conceptual Models”, IEEE Transactions on Software Engineering, Vol. 30, no.5, (Maio 2004) 328-350.
5. CYSNEIROS, L. M. ; WERNECK, Vera M. B. ; KUSHNIRUK, A. . Reusable Knowledge Satisfacing Usability Requirements. In: 13 IEEE International Requirements Engineering Conference, 2005, Paris. Proceedings of 13 IEEE International Requirements Engineering Conference. New York: Printing House, 2005. v. 1. p. 463-464.
6. CYSNEIROS, L. M.. Catalogues on Non-Functional Requirements. Disponível em: http://www.math.yorku.ca/~cysneiro/nfrs/nfrs.htm, acessado em Julho/2012.
7. CYSNEIROS, L. M.. Evaluating the Effectiveness of Using Catalogues to Elicit Non-Functional Requirements. In Proc. Workshop em Engenharia de Requisitos (WER 2007) (pp. 107-115).
8. ECLIPSE, 2011. Disponível em: http://pt.wikipedia.org/wiki/Eclipse_(software), acessado em: agosto/2012

9. JENA, 2012. Apache Jena. Disponível em: http://jena.apache.org/index.html, acessado em julho/2012.
10. LÓPEZ, C.; Cysneiros, L.M.; Astudillo, H. . NDR Ontology: Sharing and Reusing NFR and Design Rationale Knowledge. Managing Requirements Knowledge, 2008. MARK '08. First International Workshop on. Set/2008.
11. LÓPEZ C.; Inostroza, P.; Cysneiros, L.M. and Astudillo, H. . Visualization and comparison of architecture rationale with semantic *Web* technologies, Journal of Systems and Software, v.82 n.8, p.1198-1210, Agosto, 2009.
12. NETBEANS, 2012. Disponível em: http://netbeans.org/community/releases/72/ acessado em: janeiro/2012
13. OPENOME, 2009. OpenOME, an open-source requirements engineering tool. Disponível em: https://se.cs.toronto.edu/trac/ome/wiki, acessado em Julho/2012.
14. PROTÉGÉ, 2012. What is protégé?. Disponível em: http://protege.stanford.edu/overview/index.html, acessado em Agosto/2012.
15. SANCHO, P. P.; Juiz, C.; Puigjaner, R.; Chung, L. and Subramanian, N. . An approach to ontology-aided performance engineering through *NFR Framework*. In WOSP '07: Proceedings of the 6th international workshop on Software and performance, pages 125–128, New York, NY, USA, 2007. ACM.
16. StarUML, 2005. StarUML The Open Source UML/MDA Platform. Disponível em: http://staruml.sourceforge.net/en/index.php, acessado em Outubro/2012.
17. SUPAKKUL, S.; Chung, L.. RE-Tools: A Multi-notational Requirements Modeling Toolkit. Disponível em: http://www.utdallas.edu/~supakkul/tools/RE-Tools/index.html, acessado em Outubro/2012.
18. RDF, 2004. Resource Description Framework (RDF): Concepts and Abstract Syntax. Disponível em http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/. Acessado em julho/2012.
19. WERNECK, Vera M. B. ; CYSNEIROS, L. M. . Early Analysis of Usability Requirements. In: IADIS Virtual Multi Conference on Computer Science and Information Systems (MCCSIS), 2009, Carvoeiro. Proceedings of IADIS Virtual Multi Conference on Computer Science and Information Systems (MCCSIS), 2009. v. 1.
20. YU, E. . Modelling Strategic Relationships for Process Reengineering, Ph.D. thesis, also Tech. Report DKBS-TR-94-6, Dept. of Computer Science, University of Toronto, 1995.
21. YU, E., Giorgini, P., Maiden, N. e Mylopouloseds, J., The Social Modeling for Requirements Engineering Yu, E. The MIT Press, 2011.
22. YU, E. and Cysneiros, L.: Designing for privacy and other competing requirements. In: Proc. of SREIS 2002, North Carolina, Raleigh. 2002.