Utilizando Requisitos Não Funcionais para Análise de Modelos Orientados a Dados

Luiz Marcio Cysneiros, Julio Cesar Sampaio do Prado Leite

PUC-Rio, Rio de Janeiro, Brasil
cysneiro,julio{@inf.puc.rio.br}

Abstract. Recentes pesquisas apontam para a necessidade de modelos conceituais serem capazes de lidar com metas, de forma a poderem retratar situações complexas que existem no mundo real. Este trabalho aborda o uso de requisitos não funcionais (RNF) na análise de modelos orientados a dados. Esta análise é baseada no uso do Léxico Ampliado da Linguagem (LAL) como âncora para a construção de um grafo de RNFs e para a construção do modelo de dados e de heurísticas para a validação do modelo de dados. Nós adotamos o largamente utilizado modelo entidade relacionamento como o modelo de representação de dados. Apesar de apresentarmos apenas o modelo ER, acreditamos que o método apresentado neste artigo possa ser estendido para outros modelos de dados.

Keywords: engenharia de requisitos, requisitos não funcionais, MER.

Utilizando Requisitos Não Funcionais para Análise de Modelos Orientados a Dados

Luiz Marcio Cysneiros [*]
Julio Cesar Sampaio do Prado Leite[*]
Departamento de Informática PUC- Rio
R. Marquês de São Vincente 225
22453-900 - Rio de Janeiro, Brasil
e-mail: {cysneiro, julio}@inf.puc-rio.br

Resumo

Recentes pesquisas apontam para a necessidade de modelos conceituais serem capazes de lidar com metas, de forma a poderem retratar situações complexas que existem no mundo real. Este trabalho aborda o uso de requisitos não funcionais (RNF) na análise de modelos orientados a dados. Esta análise é baseada no uso do Léxico Ampliado da Linguagem (LAL) como âncora para a construção de um grafo de RNFs e para a construção do modelo de dados e de heurísticas para a validação do modelo de dados. Nós adotamos o largamente utilizado modelo entidade relacionamento como o modelo de representação de dados. Apesar de apresentarmos apenas o modelo ER, acreditamos que o método apresentado neste artigo possa ser estendido para outros modelos de dados.

Palavras-chave: engenharia de requisitos, requisitos não funcionais, MER

Abstract

Recent research points out that conceptual model need to deal with goals, in order to model complex situations that happen in the real world. This work proposes to use non-functional requirements to analyze data models. The analysis is anchored in the use of Language Extended Lexicon (LEL) to build both NFR graph and the data models together with the use of heuristics to validate data models. We adopted the widely used ER conceptual model as the data modeling representation schema. Although we present only the ER model, we believe the proposed method may be extended to other data models.

Key-Words : Requirements Engineering, Non-Functional Requirements, ER model

1 – Requisitos Não Funcionais

Recentes trabalhos na engenharia de requisitos [Dardenne 95] [Mylopoulos 92] [Mylopoulos 95] apontam para a crescente necessidade de modelos conceituais serem capazes de lidar com metas, de forma a retratar com mais precisão necessidades do mundo real. Uma das vantagens de termos modelos conceituais que lidam com metas é a capacidade de se representar aspectos não funcionais como, confidencialidade, performance, qualidade e precisão.

Acreditamos, assim como outros autores [Mylopoulos 92] [Loucopoulos 95] [Kirner 96] [Chung 95] [Kotonya 95], que esses aspectos não funcionais devam ser tratados como requisitos não funcionais. Requisitos não funcionais, ao contrário dos funcionais, não expressam nenhuma função (transformação) a ser implementada em um sistema de informações; eles expressam condições de comportamento e restrições que devem prevalecer.

A identificação de impactos, causados por restrições operacionais ao software ou por específicos comportamentos, mostrou-se como um grande desafio a ser contornado para que tenhamos softwares que, satisfazendo seus requisitos não funcionais, sejam desenvolvidos em tempo hábil.

A literatura tem registrado vários casos onde fica evidente que a falta de um tratamento adequado dos requisitos não funcionais pode levar a resultados desastrosos. Um desses é o caso do serviço de ambulâncias da cidade de Londres [Finkelstein 96] que foi alvo de uma rigorosa auditoria. Essa auditoria é, certamente, uma das peças mais significativas sobre os custos da não utilização dos preceitos de engenharia de software na construção de sistemas complexos.

Acreditamos, assim como no projeto EAGLE [EAGLE 95], que requisitos não funcionais estão normalmente ligados a requisitos funcionais.

Este artigo é um aprofundamento da proposta apresentada por Cysneiros [Cysneiros 97], onde os requisitos não funcionais são apresentados e representados inicialmente em um grafo de RNFs [Mylopoulos 92], para em seguida serem representados no modelo entidade relacionamento de forma a possibilitar uma melhor identificação dos impactos que a inclusão de requisitos não funcionais podem trazer ao modelo de dados.

Inicialmente (Seção 2), descreveremos o papel do modelo de dados da modelagem de requisitos.

Posteriormente (Seção 3), descrevemos nossa proposta de um método de análise de modelo de dados com relação à requisitos não funcionais. Esta análise é baseada no uso do Léxico ampliado da linguagem como âncora para a construção de um grafo de RNFs e para a construção do modelo de dados. O LAL será também utilizado nas heurísticas para a validação do modelo de dados. Nós adotamos o largamente utilizado modelo entidade relacionamento como o modelo de representação de dados.

Em seguida (Seção 4), daremos uma visão de alguns trabalhos relacionados à requisitos não funcionais.

Por fim (Seção 5) comentaremos as contribuições trazidas por este trabalho e apontamos futuras pesquisas a serem realizadas.

2 – O Modelo de Dados e seu Papel na Modelagem de Requisitos

A ciência da computação tem feitos consideráveis avanços no que diz respeito a modelagem. Linguagens que podem gerar modelos de considerável nível de abstração já são utilizadas na pratica, e é grande o progresso nos esquemas de formalização dessas linguagens.

Diferentes propostas de linguagens existem e essas são normalmente conhecidas como linguagens de especificação. Os modelos produzidos por estas linguagens são extremamente úteis na tarefa de prover semântica para as informações coletadas.

É importante ressaltar que um modelo (fruto de uma linguagem ou combinação de linguagens de representação) pode ser profundo ou raso. Com isso queremos dizer que um modelo pode ser bastante forte semanticamente ou muito menos forte. Um exemplo e' se compararmos o modelo KAOS [Lamsweerd 91] com o modelo Entidade Relacionamento [Elmasri 89]. Modelos rasos são úteis principalmente nas fases exploratórias da Engenharia de Requisitos, desde de que entenda-se que esses modelos são modelos não totalmente completos.

Das linguagens de representação mais comumente utilizadas pela industria, o modelo Entidade Relacionamento, ou ER, desponta. Esse modelo foi originalmente proposto Chen e apesar de todos os seus problemas tem tido grande aceitação. Essa aceitação, acreditamos, é devido, principalmente, a sua característica de um modelo raso. Utiliza-se apenas de três conceitos (entidade, relacionamento e atributos) ao passo, que por exemplo KAOS utiliza cerca de 12 conceitos.

A utilização do modelo ER na fase de modelagem da engenharia de requisitos tem por objetivo dar ao engenheiro de software uma noção preliminar da visão de dados do futuro sistema. Normalmente esse modelo é utilizado na fase de especificação por profissionais de banco de dados, para servir de base para o modelo de dados do sistema.

Apesar de ser um modelo raso, não consideramos que o mesmo possa ser validado através de consulta a clientes e usuários, porque para estes o modelo não resulta em uma abstração de fácil entendimento. No entanto para profissionais de engenharia de software e banco de dados o modelo ER é facilmente compreendido.

A possibilidade de análise de um modelo ER ainda na fase de definição de requisitos é importante porque adicionará qualidade aos requisitos produzidos. Melhor ainda se essa análise prescindir de atores do Universo de Informações que não estão habituados a modelos. Nossa proposta encaixa-se exatamente neste cenário, isto é, utilizar um meio de analise do modelo de dados que não necessite interação direta com o cliente/usuário.

3 – Método de Análise

Uma vez que acreditamos que requisitos não funcionais devam ser tratados desde o princípio do processo de desenvolvimento de software, precisaremos de uma ferramenta para representar os RNFs.

Para este fim nossa proposta é utilizar o grafo de RNFs proposto por Mylopoulos [Mylopoulos 92] com pequenas modificações. Este grafo conterá os requisitos não funcionais do sistema, que por sua vez serão refinados em submetas que formarão uma árvore que em suas raízes conterá os atributos necessários para satisfazer esse RNF. A seção 3.1 detalha a nossa proposta para esta ferramenta.

O método de análise do modelo entidade relacionamento proposto neste trabalho, é baseado na idéia de se usar símbolos do LAL para construir tanto o grafo de RNFs quanto o modelo entidade relacionamento. Isto ajudará a localizar em que entidade ou relacionamento um RNF irá impactar.

Uma vez identificado a entidade ou relacionamento impactado pelo RNF, deverá ser realizada uma validação dos atributos da entidade ou relacionamento, confrontando-os com os do RNF.

Algumas heurísticas foram criadas para auxiliar a identificação de que entidades ou relacionamentos são impactados pela satisfação de um RNF e para a identificação de possíveis problemas no modelo de dados oriundos da decisão de se implementar esse RNF.

3.1 – Léxico Ampliado da Linguagem (LAL)

O objetivo do LAL [Leite 93], é conhecer o vocabulário usado no UdI. O LAL é âncorado numa idéia simples: Entender a linguagem do problema sem se preocupar em entender o problema. O principal objetivo do LAL é registrar símbolos (palavras e frases) peculiar a um domínio de aplicação.

Leite [Leite 95] define UdI da seguinte maneira:

"Universo de Informações é o contexto geral no qual o software deverá ser desenvolvido e operado. O UdI inclui todas as fontes de informação e todas as pessoas relacionadas ao software. Essas pessoas são também conhecidas como os atores desse universo. O UdI é a realidade circunstanciada pelo conjunto de objetivos definidos pelos que demandam o software"

O LAL [Leite 93] é baseado em um sistema de códigos composto por símbolos, noções e impactos. Este é construído utilizando-se um conjunto de descrições, com o objetivo de se obter a semântica dos símbolos encontrados. Essa descrição utiliza um sistema de representação onde cada símbolo é descrito por noções e impactos. Cada símbolo é uma entrada e as noções e impactos são itens dessa entrada, nos quais toda referência a outras entradas deve ser sublinhada (princípio da circularidade).

A descrição da noção deve procurar identificar o seu significado e as relações fundamentais de existência com outras entradas.

A descrição do impacto deve espelhar os efeitos do uso do símbolo no UdI, ou, qualquer efeito que alguma ocorrência no UdI possa acarretar no símbolo.

Essas descrições devem ser orientadas pelos princípios da circularidade e do vocabulário mínimo. O princípio da circularidade dita que na descrição de noções e impactos empregue-se os próprios símbolos da linguagem. O princípio do vocabulário mínimo dita que a descrição de um símbolo deve utilizar um vocabulário restrito da linguagem natural.

3.2 – Adaptação do Grafo de Requisitos Não Funcionais

Nossa proposta é utilizar o grafo de RNFs proposto por Mylopoulos [Mylopoulos 92] com pequenas modificações em sua idéia original. Utilizaremos aqui, a mesma idéia de RNFs sendo representados como metas de RNFs, metas para satisfação de RNFs, metas argumentativas e os métodos de decomposição apresentados por Mylopoulos [Mylopoulos 92].

A primeira modificação proposta é identificar acima da árvore de RNFs o ator do UdI, ou um determinado setor da empresa, onde foi identificado o RNF, e consequentemente, ao qual ele estará mais diretamente relacionado. Esta modificação tem por objetivo possibilitar algum nível de rastreabilidade de onde foi originado o RNF. Desta forma uma futura necessidade de melhor detalhar um RNF ou possíveis conflitos identificados, será facilitada.

A utilização do LAL na construção do grafo de RNFs irá auxiliar na identificação de qual entidade ou relacionamento, existente no modelo entidade relacionamento, irá ser impactada pela decisão de satisfazer este RNF.

Esta exigência pode ser expressa da seguinte forma :

a) Dado o conjunto R de Requisitos não funcionais e o conjunto S de Símbolos do LAL onde :

$$R=r_1, r_2, ..., r_n$$

$$S= s_1, s_2, ..., s_n$$

Existe uma relação $\{(r_i,s_j) \mid \forall\ r_i \in R\ \exists\ s_j \in S\ \}$.

A segunda modificação proposta é escrever, ao lado do nó, o RNF seguido do símbolo do LAL que identifica a classe de dados afetada por este RNF. Este símbolo do LAL estará referenciado entre colchetes. A figura 1, por exemplo, contém o RNF **Qualidade[Laudo]**, que irá apontar todos os atributos necessários para satisfazer o RNF **Qualidade** aplicada à classe de dados **Laudo**, classe esta que necessariamente deverá ser uma entrada do LAL. Este RNF será decomposto em submetas que serão necessárias para que ele seja satisfeito e assim por diante, até que o engenheiro de software considere que o detalhamento das necessidades são suficientes.

As submetas que decompõe outra meta/submeta serão separadas por um ponto. As submetas poderão ou não serem entradas do LAL.

Os nós folha de cada árvore de RNF irão espelhar os necessários atributos para se satisfazer este RNF. Podemos então dizer que o conjunto de atributos de um RNF será o conjunto $A=\{ a_1, a_2, ..., a_{n,} \}$ de todos as submetas que estiverem no nível folha da árvore de RNF.

Estes atributos serão classificados em dois diferentes tipos : Atributos Gerais e Atributos de Dados.

Atributos gerais irão expressar restrições e comportamentos específicos que devem ser satisfeitos pelo sistema. Submetas como "O sistema deve responder em no máximo 5 segundos" ou "Os laudos devem ser impressos em impressoras laser" são exemplos de atributos gerais.

Atributos de dados irão expressar dados que deverão ser também atributos de uma classe de dados de forma a permitir que este RNF seja satisfeito.

3.2.1 – Exemplo

A figura 1 mostra um exemplo de parte de um grafo de RNF aplicado a um sistema de informações para laboratório (SIL). Decompondo o RNF **Qualidade[Laudo]**, encontraremos o que o ator do UdI entende por qualidade do laudo como sendo "O laudo deve ser impresso somente em impressoras laser e os exames devem estar em uma determinada ordem".

Isto foi representado pelas submetas **Qualidade[Laudo.Impressão a Laser]** e **Qualidade[Laudo.Ordem de Exames]**.

A submeta **Qualidade[Laudo.Impressão a Laser]** é um atributo geral que já é bastante claro, não demandando mais refinamentos.

A submeta **Qualidade[Laudo.Ordem de Exames]** teve que ser decomposta, já que era necessário saber que tipo de ordem deve ser respeitada na impressão do laudo.

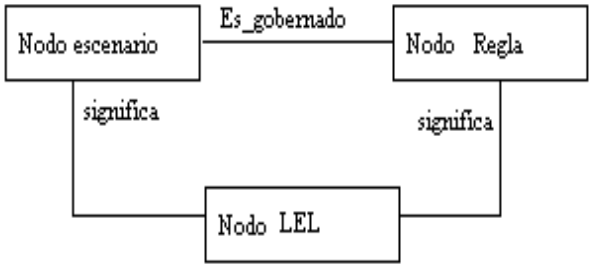

Figura 1 – Exemplo de Grafo de RNFs

Decompondo esta submeta, foi possível identificar que para o ator do UdI era necessário que o laudo fosse impresso seguindo uma determinada ordem de setores (O setor A antes do B) e uma determinada ordem de exames (O exame X antes do Y). Isto foi representado como **Qualidade[Laudo.Ordem de Exames.Ordem de Setor]** e **Qualidade[Laudo.Ordem de Exames.Ordem específica]**.

Neste exemplo o conjunto de atributos A necessários para satisfazer o RNF **Qualidade[Laudo]** seria descrito por : A= {Impressão a Laser, Ordem de Setor, Ordem específica}. Para efeitos de análise do modelo de dados será utilizado apenas o conjunto de atributos de dados. Neste caso o conjunto de atributos AD a ser utilizado para a análise do modelo de dados seria representado por : AD={Ordem de Setor, Ordem específica}.

3.3 – Adaptação do Modelo Entidade Relacionamento

A adaptação do modelo entidade relacionamento [Navathe 92] por nós proposta consiste apenas na utilização do LAL para a construção do modelo entidade relacionamento. Desta forma toda entidade constante do modelo entidade relacionamento, que expressa os requisitos funcionais do sistema, deverá ser expressa utilizando-se um símbolo do LAL.

Esta propriedade pode ser descrita da seguinte forma :

b) Dado o conjunto de entidades E composto das entidades existentes no modelo Entidade relacionamento e o conjunto S de Símbolos do LAL então :

$$E= e_1, e_2, ..., e_n,$$

Existe uma relação $\{ (e_k,s_j) \mid \forall e_k \in E \exists s_j \in S \}$.

3.4 – Heurísticas para a Detecção de Possíveis Inconsistências no Modelo de Dados

Uma vez definidas as adaptações ao uso do grafo de RNF e do modelo entidade relacionamento, descrevemos a seguir heurísticas a serem aplicadas para a detecção de possíveis inconsistências no modelo entidade relacionamento no que tange à satisfação dos requisitos não funcionais elicitados.

1. Uma vez que a e b constantes das Seções 3.3.2 e 3.3.3 é satisfeita, podemos dizer que deve existir uma função L que determina a ligação de um RNF com uma entidade do modelo entidade relacionamento, conforme especificado a seguir.

$$L(r_j,e_k) \mid \forall r_i \in R \exists e_k \in E \Rightarrow \{(r_i,s_j) \wedge (e_k,s_j) \mid \forall r_i \in R \exists s_j \in S \exists e_k \in E \wedge e_k \equiv s_j\}$$

Se esta função não for satisfeita, isto é, se existir um RNF associado a um símbolo do LAL e este símbolo não for uma entidade do modelo, teremos uma indicação de que, possivelmente o modelo entidade relacionamento não contém todas as entidades que seriam necessárias para satisfazer este RNF. Caberá então ao engenheiro de software buscar no grafo de RNF o ator associado ao RNF e pesquisar possíveis falhas do modelo entidade relacionamento associadas a este ator do UdI.

2. Uma vez que a heurística 1 foi aplicada e uma entidade foi identificada no modelo entidade relacionamento, deve ser feito o balanceamento dos atributos entre RNF e entidade ou relacionamento. Note-se que um RNF pode ter diversos atributos e alguns podem estar associados não com a entidade identificada em 1, mas sim com algum relacionamento desta entidade. O engenheiro de software deverá então verificar que atributos do RNF estarão ligados à entidade e quais estarão ligados a um relacionamento desta entidade.

   Para os atributos ligados à entidade a regra a seguir deve ser verificada:

   Dado o conjunto A de atributos de um RNF e o conjunto AE que representa o conjunto de atributos de uma entidade $e_k \in$ E então :

$$A=a_1, a_2, ..., a_n,$$
$$AE= ae_1, ae_2, ..., ae_n,$$
$$\text{Tal que } \forall\ a_i \in A\ \exists\ ae_j \in AE \wedge e_k \equiv s_j.$$

   Se a regra a seguir não for verdadeira existem atributos no RNF que não existem na entidade. Consequentemente, isto apontará para possíveis adições de atributos à entidade ou mesmo para a criação de uma nova entidade.

Para os atributos ligados ao relacionamento a regra a seguir deve ser verificada :

Dado o conjunto A de atributos de um RNF e o conjunto AR que representa o conjunto de atributos de um relacionamento $re_k \in$ RE então :

$$RE=re_1, re_2, ..., re_n$$
$$A=a_1, a_2, ..., a_n,$$
$$AR= ar_1, ar_2, ..., ar_n,$$
$$\text{Tal que } \forall\ a_i \in A\ \exists\ ar_j \in AR \wedge re_k \equiv s_j.$$

Analogamente à entidade, se a regra acima não for satisfeita existirão atributos que pertencem ao RNF que não estão presentes no relacionamento ao qual a entidade se liga. Consequentemente, isto apontará para possíveis adições de atributos ao relacionamento.

A figura 2 apresenta um exemplo de um Modelo ER onde aplicamos as heurísticas acima para o exemplo mostrado na figura 1. No exemplo mostrado na figura 2 a entidade representada em negrito foi acrescentada após a aplicação das heurísticas. Neste caso primeiramente identificamos no modelo onde aparece a classe de dados Laudo. Uma vez identificada a entidade Laudo, foi aplicada a heurística que ímplica que a entidade Laudo contenha todos os atributos do requisito não funcional **Qualidade[Laudo]** conforme visto na seção 3.2.

Uma vez que identificamos que os atributos encontrados no grafo para o requisito não funcional **Qualidade[Laudo]** não estão presentes na entidade Laudo do Modelo ER, decidimos então que era necessário incluir a entidade Ordem Exames no Modelo ER.

Figura 2 – Exemplo de Aplicação das Heurísticas

4 – Trabalhos Relacionados

4.1 - Nonfunctional Requirements of Real-Time Systems

Neste trabalho [Kirner 96] são discutidas as propriedades de seis dos mais importantes tipos de requisitos não funcionais para o domínio de sistemas de tempo real. Para cada requisito não funcional é descrito como medi-los, técnicas de como assegurar a presença dos requisitos não funcionais no projeto, e como especificar o requisito numa especificação de requisitos.

Os seis requisitos não funcionais abordados são descritos a seguir conforme em [Kirner 96] : Performance, Confiabilidade, Seguro, Segurança, Facilidade de uso, e Manutenabilidade

Para cada um dos requisitos não funcionais acima descritos, o trabalho trata em linhas gerais de como realizar métricas de ocorrência destes requisitos não funcionais, técnicas para assegurar que o projeto satisfaça a este, requisitos não funcionais e processos de especificação de requisitos.

O trabalho descreve ainda alguns inter-relacionamentos entre requisitos não funcionais específicos, como seguro e confiável, que são mostrados como conceitos ortogonais, pois, um sistema pode ser seguro mas não confiável (um automóvel que não liga), assim como confiável mas não seguro (uma pistola carregada num centro comercial lotado). Isto ressalta a importância de se definir precisamente o que cada um dos requisitos não funcionais significa no contexto da aplicação e seus inter-relacionamentos.

4.2 Representing and using Non-Functional Requirements.

Os requisitos não funcionais são enfocados de uma maneira genérica em [Mylopoulos 92] e [Chung 95], os RNFs são tratados como metas possivelmente conflitantes onde estes RNFs devem ser identificados em sua forma mais geral e refinados até que chegue-se a um conjunto de requisitos, que satisfaçam ao requisito geral.

Este trabalho relaciona ainda, decisões de desenho junto aos RNFs e busca justificar decisões relativas ao domínio da aplicação e auxiliar na detecção de erros. Durante o processo de desenho são propostas as metas a serem atingidas que são então organizadas numa estrutura de grafo, no espirito das árvores E/OU usadas na resolução de problemas.

Entretanto, essas metas que representam requisitos não funcionais raramente podem ser consideradas como "satisfeitas" num senso estrito. Na verdade diferentes decisões de desenho irão contribuir positiva ou negativamente no cumprimento de uma determinada meta. O trabalho fornece alguns exemplos de como identificar e representar tanto os requisitos não funcionais encontrados, quanto eventuais conflitos detectados, e ainda representa alegações feita por atores do UdI que o engenheiro de software julgue relevantes.

Conforme dito na seção 3, baseamo-nos na proposta contida neste artigo para a representação dos requisitos não funcionais.

4.3 - Requirements Engineering With Viewpoints

Em [Kotonya 95] podemos encontrar uma proposta de utilização de um método de definição de requisitos orientado a pontos de vista (VORD), que tenta integrar os requisitos funcionais, não funcionais, e de controle numa mesma especificação de requisitos. Os pontos de vista são divididos em dois diferentes tipos :

- **Diretos : c**orrespondem aos clientes que recebem serviços do sistema e enviam informações de controle e dados para os sistema. Podem ser ainda operadores e usuários de sistemas ou mesmo outros subsistemas ligados ao sistema em estudo.
- **Indiretos :** possuem interesse em alguns ou todos os serviços disponíveis no sistema, mas não interagem diretamente com o sistema. Podem gerar requisitos com restrições aos serviços disponíveis para os pontos de vista diretos.

O trabalho se concentra nos três primeiros passos do método VORD :

1. Identificação e estruturação de pontos de vista
2. Documentação de pontos de vista
3. Análise e especificação de pontos de vista

A identificação e estruturação dos pontos de vista se baseiam em que o processo de entender e identificar um sistema, seu ambiente, requisitos, e restrições residem consideravelmente em "autoridades do sistema". Estas autoridades são pessoas ou documentos com particular interesse, ou grande conhecimento do domínio da aplicação. Podemos aqui incluir usuários finais, engenheiros de software, documentação de sistemas existentes e outros.

Estas autoridades de sistema foram generalizadas em um diagrama em árvore de classes abstratas de pontos de vista, que pode ser utilizada como ponto de partida para a determinação de pontos de vista específicos. Esta árvore é subdividida em dois ramos de pontos de vista, diretos e indiretos.

Sob o ramo dos pontos de vista diretos, podemos encontrar por exemplo, o operador e o sistema existente.

Sob o ramo de pontos de vista indiretos, podemos encontrar: engenharia, regulamentações, organização e ambiente. Ainda caminhando na árvore, sob Organização poderíamos encontrar, consumidor e políticas. Já sob engenharia poderíamos encontrar manutenção e padrões.

Este trabalho, efetivamente, traz uma interessante abordagem dos requisitos não funcionais, utilizando pontos de vista. Apenas o ponto de identificar impactos de requisitos não funcionais em outros requisitos não funcionais identificados, ou de requisitos não funcionais com requisitos funcionais não é ainda explorado o suficiente.

5 – Conclusão

Em uma recente palestra no IFIP G 2.9 [Mylopoulos 95], Mylopoulos coloca que estamos migrando de modelos "2x2' para modelos "4x4" . Os modelos "2x2" caracterizam-se por dois conceitos (entidade e atividade) e por dois mecanismos de estruturação (generalização e agregação). Modelos "4x4" devem cobrir 4 conceitos (entidade, atividade, meta e dependência) bem como 4 mecanismos de estruturação (generalização, agregação, classificação e pontos de vista).

Este trabalho propõe um método para validar modelos de dados em relação aos requisitos não funcionais do sistema. O método é centrado na utilização do LAL para a construção do grafo de RNF e do modelo de dados e apresenta heurísticas para validar o modelo de dados a partir do grafo de RNFs.

Neste trabalho utilizamos o modelo entidade relacionamento [Navathe 92], acreditamos porém, que este método possa ser estendido para outros modelos de dados.

Trabalhos futuros incluem o aprofundamento das heurísticas apresentadas e a extensão do método para outros modelos de dados.

6 – Bibliografia

[Chung 95] Chung, L., Nixon, B.A. "Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach" *Porch. 17th Int. Con. on Software Eng*. Seatle, Washington, April 24-28, 1995.

[Cysneiros 97] Cysneiros, L.M. and Leite, J.C.S.P. "Definindo Requisitos Não Funcionais" *Anais do XI SBES,* Out 1997, pp:49-54.

[Dardenne 95] Dardenne, A.. van Lamsweerde, Fickas, S.. "Goal Directed Requirements Acquisition*". Science of Computer Programming, Vol. 20* Apr. 1993, pp. 3-50.

[Finkelstein 96] Finkelstein, A. and Dowell J. "A Comedy of Errors: The London Ambulance Service Case Study" *Proceedings of the Eighth International Workshop on Software Specification and Design*, IEEE Computer Society Press 1996, pp. 2-5.

[EAGLE 95] Evaluation of Natural Language Processing Systems - http://www.issco.unige.ch/ewg95 - 1995.

[Kirner 96] Kirner T.G. , Davis A .M. , "Nonfunctional Requirements of Real-Time Systems", *Advances in Computers, Vol 42* pp 1-37.

[Kotonya 95] Kotonya G. , Sommerville I. , "Requirements Engineering With Viewpoints", *Cooperative Systems Engineering Group Technical Report* CSEG/10/1995.

[Lamsweerde 91] A. Lamsweerde, A. Dardenne and F. Dubisy, KAOS Knowledge Representations as Initial Support for Formal Specification Processes, *Universite Catholique de Louvain, Unite d'Informatique, Research Report* RR-91-8, 1991.]

[Leite 93] Leite J.C.S.P. and Franco, A.P.M. "A Strategy for Conceptual Model Acquisition" *in Proceedings of the First IEEE International Symposium on Requirements Engineering*, SanDiego, Ca, IEEE Computer Society Press, 1993 pp. 243-246.

[Leite 95] Leite J.C.S.P., Oliveira, A.P.A., "A Client Oriented Requirements Baseline", *in Proceedings of the Second IEEE International Symposium on Requirements Engineering*, York, UK, IEEE Computer Society Press, 1995 pp. 108-115.

[Loucopoulos 95] Loucopoulos, Pericles and Karakostas, *" System Requirements Engineering",* McGraw Hill International Series in Software Engineering, 1995.

[Mylopoulos 92] Mylopoulos,J. Chung, L., Yu, E. and Nixon, B., "Representing and Using Non-functional Requirements: A Process-Oriented Approach." *IEEE Trans. on Software Eng, 18(6),* June 1992, pp.483-497.

[Mylopoulos 95] Mylopoulos, J. "Capturing Intentions in Requirements engineering: A modeling Perspective" *Invited Talk at the IFIP WG 2.9 meeting*, Bramshill, UK, March 1995.

[Navathe 92] Navathe, S.B., "Evolution of Data Modeling for Databases" *Communications of the ACM, Vol 35* No 9 1992 pp 112-123.