Definindo Requisitos Não Funcionais

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

Requisitos não funcionais expressam qualidades de cunho geral, bem como, restrições específicas de um determinado problema. Esse tipo de requisito sempre existiu mas não vinha sendo tratado de forma sistematizada quando se pensava na definição de um software. Esse trabalho aborda diretamente o aspecto de requisitos não funcionais durante as fases iniciais do desenvolvimento de software e propõe uma representação que integra requisitos não funcionais com uma representação de modelagem de dados. Nossa estratégia foi validada com um estudo de caso real. Acreditamos que esse trabalho preenche uma importante lacuna no tratamento de requisitos que antes tinham um viés apenas de ordem funcional.

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

Abstract

Non-Functional requirements express both quality properties and constraints for specific problems. This kind of requirement has always been present, but not treated in a systematic way during software definition. This work deals with non-functional aspects during the initial phases of software development and proposes a representation that integrates non-functional requirements with a data modeling representation. Our proposal has been validated using a case study. We believe that this work fills a gap in the requirements definition process.

Keywords : requirements engineering, non-functional requirements, elicitation, ER model

1 - Introdução

Cada vez mais o software é encarado como produto e, como produto, precisa ter qualidade e preço. Para que os aspectos de qualidade sejam considerados no processo de produção é indispensável que se leve em consideração os requisitos não funcionais. Pensar que um software de qualidade era aquele que estava o mais próximo possível de 100% de conformidade com os requisitos funcionais esperados dele deixa de ser uma realidade. Requisitos não funcionais (RNFs) passaram a ser exigidos pelos clientes para que um software seja considerado de qualidade; tais como : facilidade de uso, desempenho, clareza de informações e outros.

Quanto ao preço, esse é sempre dependente do custo de desenvolvimento do software e a não observância da necessidades de RNFs durante o processo de desenvolvimento do

* This work was suported by CNPq
* This work is supported in part by CNPq grant n. 510845/93-2

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

software, pode levar a uma recodificação custosa e demorada o que, consequentemente, eleva o preço do software.

Os requisitos não funcionais, ao contrário dos funcionais, não expressam nenhuma função a ser realizada pelo software, e sim comportamentos e restrições que este software deve satisfazer.

Um exemplo da diferença entre os dois pode ser visto a seguir :

Requisito funcional :

- calcular saldo a pagar de imposto de renda.

Requisito não funcional :

- a declaração deve ser simples o suficiente para que o cálculo do imposto seja feito sem a necessidade do usuário pedir ajuda a um contador.

Para satisfazer um requisito não funcional é possível que sejam criados conflitos, tanto com outros requisitos não funcionais, como com requisitos funcionais. Consequentemente, o não tratamento dos requisitos não funcionais durante o desenvolvimento do software pode levar a que esses conflitos só apareçam quando da implementação do software. Essa ocorrência, comum em vários projetos, demonstra a fragilidade das práticas atuais.

A identificação de impactos, causados por restrições operacionais ao software, 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 uma falta de tratamento adequada dos requisitos não funcionais pode levar a resultados desastrosos. Um desses é o caso do serviço de ambulâncias da cidade de Londres 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 [Finkelstein 96].

Nosso trabalho procurou justamente atacar esse problema, isso é, de que forma tratar requisitos não funcionais com a devida importância. Parte de nossa proposta foi fortemente influenciada pelos trabalhos de Mylopoulos [Mylopoulos 92] [Chung 95] que modelam RNFs como uma rede de metas. Utilizamos o trabalho de Mylopoulos com uma pequena modificação para servir de primeiro modelo não funcional. Para tal, criamos uma estratégia de como elicitar RNFs, apresentada na Seção 2. Escolhemos um modelo de dados, o MER, como a representação de especificação na qual integramos os RNFs, essa integração está descrita na Seção 3. Na Seção 4 descrevemos de que forma foi criada uma nova linguagem no meta-ambiente Talisman para tratar de RNFs associadas ao MER. A Seção 5 reporta os resultados do estudo de caso conduzido em um grande laboratório de análises clínicas. Concluímos, ressaltando nossa contribuição, comparando com trabalhos já publicados e delineando pesquisas futuras.

2 - Estratégia para Elicitar Requisitos Não Funcionais

Nesse trabalho utilizamos a idéia de que requisitos, de uma maneira geral, não são estáticos e portanto sofrem mudanças durante o ciclo de vida do software [Leite 95]. Isso significa que os requisitos identificados, durante a fase de elicitação de requisitos, continuam evoluindo por todo o ciclo de vida do software, devendo o engenheiro de software ficar atento para estas possíveis evoluções e representá-las devidamente.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

O modelo SADT da Figura 1 ilustra a estratégia proposta nesse trabalho. Cada atividade desse modelo será uma subseção, na qual detalharemos o seu comportamento.

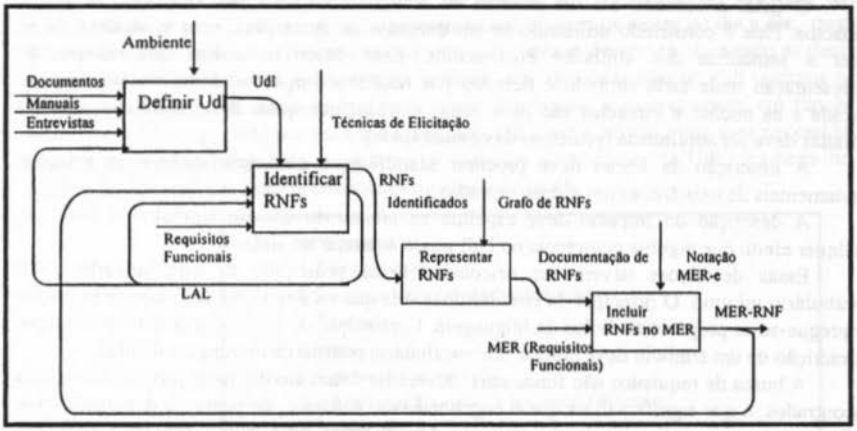

Figura 1 : SADT da estratégia de elicitação de requisitos não funcionais

2.1 - Definir UdI

Nessa fase o engenheiro de software deverá, através da leitura de manuais, livros, legislação, normas e observação do ambiente, definir o Universo de Informações (UdI) relativo ao software a ser desenvolvido. Leite [Leite 91] 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"* .

Nessa fase, já é possível que o engenheiro de software se depare com alguns requisitos não funcionais, principalmente no estudo de normas e legislações aplicadas ao UdI.

2.2 - Identificar Requisitos Não Funcionais

A elicitação de requisitos não funcionais deve ser feita separadamente dos requisitos funcionais, de forma a manter o engenheiro de software centrado no problema que ele está investigando. Isso significa que, mesmo se durante a elicitação de requisitos funcionais o engenheiro de software identificar algum requisito não funcional, ele não deve se aprofundar nesse requisito não funcional, e sim, anotá-lo à parte para posterior investigação.

Essa regra, é claro, vale também para o caso inverso que seria a identificação de um requisito funcional ainda não identificado durante a elicitação dos requisitos não funcionais

Nessa etapa, o engenheiro de software deverá utilizar uma ou mais técnicas de elicitação para identificar os requisitos não funcionais inerentes ao UdI. É importante que o engenheiro de software se familiarize com os termos usados pelos atores do UdI. Com esta

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

finalidade, utilizamos o **Léxico Ampliado da Linguagem (LAL)** [Leite 90] [Franco 92], que tem por objetivo conhecer o vocabulário usado no UdI.

O LAL 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.

A busca de requisitos não funcionais deverá ter como auxilio os requisitos funcionais encontrados, o que significa dizer que o engenheiro de software, de posse do documento que define os requisitos funcionais do sistema, deverá fazer uma busca em cada requisito funcional descrito no documento, procurando identificar eventuais RNFs a eles associados. Entretanto, alguns requisitos não funcionais, como por exemplo segurança de acesso a dados do sistema, podem ser globais ao sistema não estando relacionado a um determinado requisito funcional, e sim, presente em diversas partes do sistema, ou mesmo no sistema como um todo.

Um instrumento importante no auxilio à identificação de RNFs pode ser o uso de um *checklist* de RNFs como guia. A Figura 2 mostra um *checklist* que poderá ser usado como ponto de partida. Essa lista não pretende ser completa e sim um apanhado dos RNFs mais comuns, sendo portanto aconselhável que a medida que se identifique um RNF não constante da lista que esta seja atualizada.

Este *checklist* pode ser utilizado de duas maneiras distintas:

- No questionamento direto do cliente sobre cada um desses RNFs que ele desejaria ou não ter como por exemplo : **O que em termos de desempenho é importante para você ?** ou **Qual a precisão necessária para esta ação ?** E assim por diante, tomando uma atitude ativa em relação ao cliente.
- Na utilização de itens da lista como pontos a serem lembrados quando o engenheiro de software estiver observando o cliente em seu ambiente, durante reuniões, ou percorrendo o documento de requisitos funcionais do sistema. Serve então, como um guia de palavras e qualidades para as quais o engenheiro de software deve estar bastante atento.

É importante ressaltar que apesar da qualidade do processo de desenvolvimento de software ser certamente um RNF desejável, ele não está presente no checklist abaixo, uma vez que, esse trabalho é voltado para a identificação de RNFs relacionados à aplicação.

Além do *checklist*, o engenheiro de software deverá empregar as técnicas de elicitação que mais se adaptarem ao ambiente. As técnicas de Enfoque Antropológico, Recuperação do Desenho de Software, Análise de Protocolos e Prototipação se mostraram bastante proveitosas.

O enfoque antropológico [Goguen 93] mostra-se de grande valia para a integração do engenheiro de software ao UdI, pois este passa a sentir mais as necessidades pertinentes ao sistema de informação, tendo visões semelhantes de quais são os RNFs desejáveis pelos atores

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

do UdI em geral. Entretanto, devido ao seu conhecimento, o engenheiro de software possivelmente saberá identificá-los com mais facilidade que os outros atores em geral. Integrando-se por completo no ambiente do cliente, o engenheiro de software terá a possibilidade de sentir bem de perto os problemas que afligem os atores no dia a dia, quais as necessidades são realmente fundamentais e quais são apenas desejáveis, tornando as decisões de projetos mais criteriosas. É importante ressaltar aqui que cada ator do UdI costuma achar que seus problemas são maiores que os de quaisquer outros, e muitas vezes, em função de saber que está competindo por recursos e atenção, acaba por sobrevalorizar suas necessidades. Nesse momento a vivência do engenheiro de software no ambiente do UdI, irá permitir um melhor julgamento das reais necessidades do ator em questão

1. Desempenho
    1.1 tempo real
    1.2 velocidade de processamento
    1.3 outros
2. Precisão
    2.1 precisão numérica
    2.2 informação correta no tempo correto
    2.3 Outros
3. Compreensibilidade da informação (clareza)
4. Confiabilidade
    4.1 disponibilidade de equipamentos / informação
    4.2 Integridade
    4.3 Outros
5. Segurança
    5.1 Permissão de consulta/atualização de dados
    5.2 Confidencialidade dos dados
    5.3 Outros
6. Restrições Operacionais
7. Restrições físicas
8. Amigabilidade
9. Interface
10. Manutenabilidade
11. Portabilidade
12. Interoperabilidade
13. Custo

Figura 2- Checklist de RNFs

É conveniente que durante todo o processo o engenheiro de software mantenha-se sempre atento a expressões como as descritas na Figura 3, bem como, a adjetivos e advérbios de modo geral.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

Seria interessante/bom
É importante
ajuda muito
Ganharíamos muito se

Figura 3 - Expressões

Ao identificar um RNF, o engenheiro de software deve realizar a anotação do ator do UdI responsável pelo RNF. Isso é importante para facilitar uma futura identificação da origem desse requisito não funcional na eventualidade de ter-se que fazer alguma alteração relacionada a ele, algum julgamento de prioridades, ou mesmo um maior detalhamento desse RNF. Não é incomum, que, mais a frente o engenheiro de software tenha de tomar decisões de desenho do tipo *" Implementar A em vez de B"* por restrições impostas por requisitos não funcionais conflitantes, e, nesse momento, é bastante útil saber a origem do RNF, para que se possa ir verificar com o ator responsável se não há outro motivo ou outro RNF associado que justifique a manutenção de *B* em vez de *A* e que não tenha sido identificado anteriormente.

É importante que não sejam identificados apenas os RNFs de forma genérica, como por exemplo, especificar **desempenho** como um RNF existente e nada mais. É necessário que o requisito não funcional genérico seja instanciado (detalhado) até termos identificado o que é necessário o sistema possuir/fazer para que esse RNF seja satisfeito. No caso acima citado, dever-se-ia detalhar o que **desempenho** significa para aquele determinado ator do UdI de forma a não haver dúvidas, como por exemplo exemplificar que o sistema deve responder a um equipamento de análises clínicas a ele conectado num tempo máximo de 15 segundos.

2.3 - Representar Requisitos Não Funcionais

A proposta aqui é a utilização de uma adaptação do Grafo de Requisitos proposto por Mylopoulos [Mylopoulos 92]. Em nossa versão iremos identificar os RNFs ao lado do círculo que representa o nó do grafo e, internamente ao círculo, uma letra que identifique o Ator do UdI responsável pelo RNF. Representa-se primeiramente o RNF mais global (**Amigabilidade** por exemplo), passando então a instanciar-se esses requisitos em nós adjacentes ao primeiro, tentando detalhar o RNF identificado em submetas que satisfaçam ao RNF ao qual estiverem ligadas, e assim sucessivamente até que o nó folha contenha uma especificação clara suficiente do RNF.

O ponto de parada das instanciações é objeto de decisão individual do engenheiro de software. Uma abordagem para isso pode ser encontrada em [Dardene 91] onde é sugerido que quando um requisito não funcional for ligado a apenas um ator, então, ele estará detalhado o suficiente. Não concordamos com essa posição, uma vez que, determinados RNFs podem ser associados a apenas um ator do UdI e não estarem detalhados suficientemente, o que dependerá bastante de quanto sintética tenha sido sua definição.

Utilizaremos linhas cheias para representar as instanciações de RNFs em submetas e linhas pontilhadas para representar os possíveis RNFs conflitantes entre si.

A Figura 4 ilustra um exemplo de uso do grafo de RNFs. Nota-se que os 3 requisitos globais presentes são impactantes mutuamente. Para atingir-se um desempenho conforme especificado para a quantidade de dados a serem obtidos na admissão de pacientes e com um software que utilizasse interface gráfica, teríamos que ter estações bastante poderosas.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

Portanto, essas estações levariam a que a restrição orçamentária não fosse atendida, principalmente, levando-se em conta que é necessário que se compre equipamentos de primeira linha para satisfazer ao requisito de baixo índice de manutenção.

Observa-se ainda, que paramos a instanciação ao chegarmos a expressão de que para satisfazer ao requisito não funcional de **baixo índice de manutenção** é necessário adquirir equipamentos de primeira linha, mas seria possível argüir a necessidade de instanciar o que são equipamentos de primeira linha. A decisão de continuar ou não vai depender tanto do bom senso do engenheiro de software como da disponibilidade de dados para definir o requisito não funcional em questão.

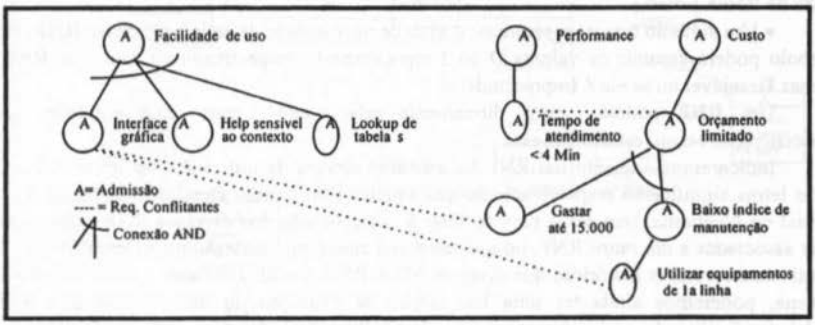

Figura 4 - Grafo de requisitos não funcionais

2.4 - Representar os Requisitos Não Funcionais no MER

Nessa etapa iremos integrar os RNFs representados no grafo anterior ao MER obtido. A representação do ator da UdI no grafo irá auxiliar-nos a identificar o ponto no MER onde o requisito não funcional está ligado. Essa fase ficará explicitada nas Seções que se seguem.

Ao final do processo de representação dos RNFs no MER é interessante realizar-se um processo de verificação fazendo uma comparação dos requisitos modelados no grafo de requisitos com os presentes no MER. Todos que existirem em um têm que existir no outro.

3 - Extensão ao MER Para Representar RNFs

A representação dos RNFs no MER [Navathe 92] surgiu pela necessidade encontrada de sermos capazes de, não apenas identificar os RNFs do sistema, mas também de identificar os impactos desses no seu desenho, bem como possibilitar decisões de desenho menos intuitivas.

O fato do MER ser uma ferramenta largamente utilizada por engenheiros de software e projetistas de bancos de dados foi fator decisivo para que ela fosse escolhida [Cysneiros 97] como o modelo de integração. Denominaremos esta nova forma de representação de **MER-RNF**.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

Com a representação do RNF no local onde ele ocorre (junto a uma entidade ou um relacionamento), temos uma melhor visualização de que impactos esse RNF poderá acarretar e com que requisito ele pode vir a conflitar. A extensão por nós proposta ao MER conforme definido por Navathe [Navathe 92] é composta de duas representações distintas:

- Uma Classe de objeto que representará o RNF propriamente dito, ao qual chamaremos de **RNF**. O **RNF** será representado pelo retângulo que simboliza entidades com uma linha horizontal em seu topo e o Nome (descrição) desse na parte central do retângulo. No topo permanece um espaço para que seja colocada a identificação do ator do UdI ligado a esse RNF. Esta identificação deverá ser a mesma utilizada na parte interior dos círculos do grafo de RNFs.
- Um símbolo que irá representar o grau de necessidade de satisfação desse **RNF.** Este símbolo poderá assumir os valores **D** ou **I** representando respectivamente que esse **RNF** é apenas **D**esejável ou se ele é **I**mprescindível.

Um **RNF** poderá estar diretamente relacionado com: uma entidade, uma especialização ou um relacionamento .

Indicaremos o quanto um RNF foi atendido através da utilização das letras **T,P** e **N.** Estas letras significarão respectivamente que aqueles RNFs estão atendidos de forma Total, Parcial ou Nenhuma. Isso serve para facilitar a compreensão dos diversos RNFs que podem estar associados a um outro RNF, no que tangem a sua implementação no sistema ou não. Se mantivermos versões históricas dos diversos MER-RNF, gerados durante o ciclo de vida do sistema, poderemos ainda ter uma boa noção da evolução do atendimento dos RNFs detectados facilitando, assim, o rastreamento de possíveis tomadas de decisão de desenho.

A Figura 5 a seguir ilustra as possibilidades.

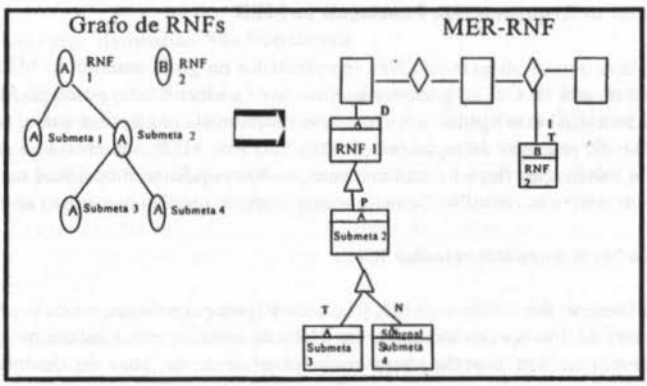

Figura 5: Exemplo do MER-RNF

Para representar os RNFs no MER o engenheiro de software se baseará no grafo de RNFs levantado anteriormente. O primeiro passo é identificar quais os atores presentes no grafo e localizá-los no MER. No caso do exemplo da Figura 4 o único ator presente é a **Admissão**, que no MER tem uma entidade com o mesmo nome. Uma vez identificado, é apenas uma questão de representar no MER os RNFs constantes do grafo de RNFs e estudar o

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

modelo em busca de eventuais implicações de desenho que a inclusão destes RNFs possam acarretar.

A Figura 6 mostra um exemplo de como ficaria o grafo de RNF mostrado na Figura 4 quando representado na extensão por nós proposta.

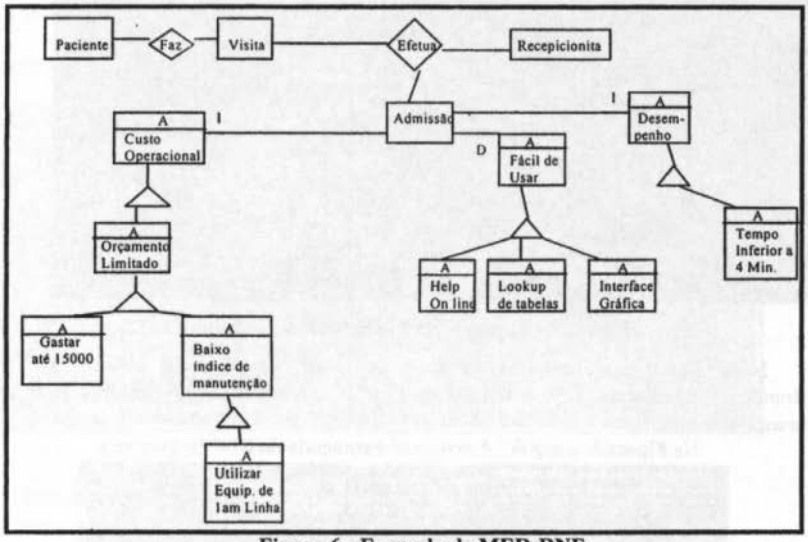

Figura 6 - Exemplo de MER-RNF

4 - Implementando o MER-RNF no Talisman

Talisman [Staa 92] é um meta-ambiente de engenharia de software assistido por computador. Talisman é uma coletânea integrada de ferramentas mecanizadas utilizadas para desenvolver, controlar a qualidade e manter planos e especificações de projetos, códigos e documentação de sistemas quaisquer.

Uma base de conhecimento prove a definição das linguagens de representação e o relacionamento entre os elementos da linguagem é administrado pelo administrador de ambiente, que pode, via utilização de linguagem de programação (formulários), adaptar o Talisman às necessidades inerentes ao seu ambiente de desenvolvimento de software.

Um dicionário de dados estendido associado a cada representação usada em um projeto constitui a base de software (repositório de projetos). Durante o desenvolvimento o engenheiro de software coloca fatos do sistema que está sendo modelado na base de software correspondente.

Para implementar o MER-RNF no Talisman foi necessária a criação da Classe de Objetos, requisitos não funcionais, e a inclusão nos formulários de especificação, validação e impressão de código próprio para realizar estas funções [Cysneiros 97].

A Figura 7, mostra um exemplo do MER-RNF implementado no Talisman

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

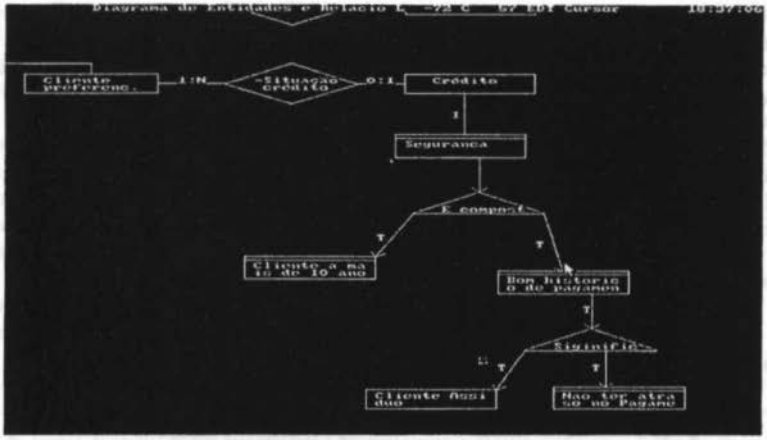

Figura 7 - Exemplo do MER-RNF no Talisman

Neste exemplo podemos ver o RNF **Segurança** associado à entidade **Crédito,** mostrando a necessidade de que o fornecimento de crédito a um cliente seja feito com segurança absoluta.

Na Figura 8, a seguir, é mostrado o conteúdo do RNF **Segurança**

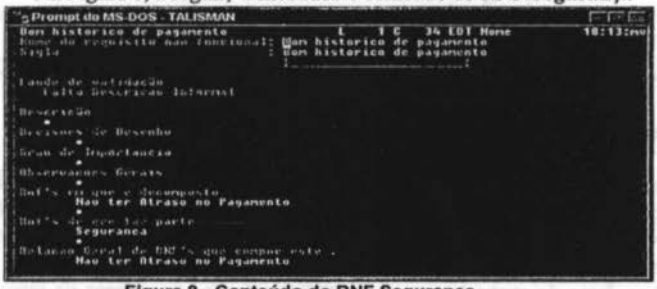

Figura 8 - Conteúdo do RNF Segurança

Podemos notar na figura que na implementação do MER-RNF no Talisman , a linguagem de descrição de um RNF possui os seguintes descritores:

Descrição : Contém a descrição de qual a razão daquele RNF.
Decisões de Desenho : Descreve as eventuais decisões de desenho que sejam necessárias para satisfazer esse RNF.
Grau de importância : Contém um valor de 0 a 10 determinando a importância e prioridade de atendimento desse RNF.
Observações : Utilizado para realização de observações em geral.
RNFs em que é decomposto : Exibe os RNFs que são especializações (submetas) diretamente ligadas ao RNF em questão.
RNFs de que faz parte : Exibe os RNFs do qual este faz parte.
Relação geral de RNFs que

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

compõe este : Relação de todas as submetas relacionadas a esse RNF.

O preenchimento dos três últimos descritores é realizado automaticamente pelo Talisman quando é feita a validação dos RNFs.

Esta validação faz ainda críticas como: falta descrição, RNF tem entidade como especialização, existem mais de uma especialização associada ao mesmo RNF.

Um exemplo desta validação pode ser visto na Figura 9, a seguir, que mostra o conteúdo do RNF **Bom histórico de pagamento**, mostrado na Figura 7.

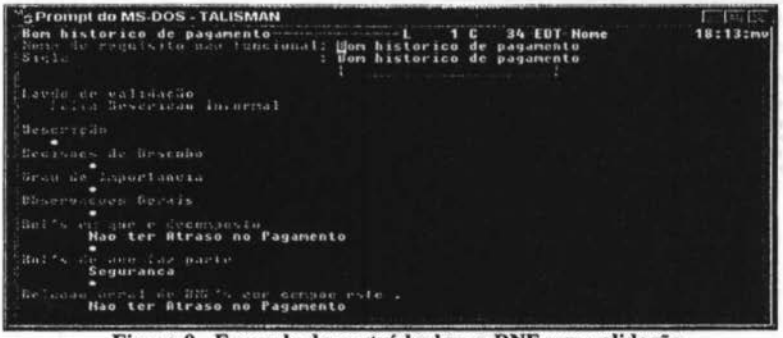

Figura 9 - Exemplo do conteúdo de um RNF com validação

5 - Estudo de Caso

A estratégia proposta nesse trabalho foi utilizada em um estudo de caso realizado em um laboratório de análises clínicas de grande porte durante o desenvolvimento de um sistema de informatização de laboratórios. Esse software foi desenvolvido por três equipes diferentes, relacionadas com a área de atuação de cada equipe, a saber: área administrativa, área de atendimento e área técnica.

A estratégia proposta foi utilizada apenas pela equipe responsável pelo desenvolvimento do software para atender a área técnica. Aqui, o software deveria compreender não apenas as necessidades de manipulação de informações como: dados do paciente, resultados anteriores e outros, como também, a integração do software com os analisadores que realizam exames e possuem capacidade de comunicação bidirecional.

A utilização do LAL [Leite 90] e do enfoque antropológico [Goguen 93] foram fatores importantes no sentido de que fomos capazes de entender a linguagem dos atores do UdI e sentir de perto suas necessidades. Aqui, vários requisitos não funcionais foram encontrados, muitos dos quais o ator do UdI não nos teria solicitado se não estivéssemos tão próximo dele. Nesse aspecto o enfoque antropológico trouxe a grande vantagem de não apenas termos a possibilidade de vivênciar o trabalho do ator do UdI e com isso ter mais sensibilidade às suas necessidades, como também fazer com que a comunicação entre o ator do UdI e o engenheiro de software fosse feita em bases mais sólidas.

A Figura 10 mostra parte do grafo de RNF obtido ao final do estudo de caso.

Em função desse grafo e através do processo descrito na Seção 2, os RNFs presentes no grafo foram incorporados ao MER. Em alguns casos, durante a incorporação de RNFs ao MER, descobríamos a necessidade de existirem novas entidades, ou, de aumentar os atributos

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

pertencentes a algumas entidades e/ou relacionamentos. Esta visualização não era simples pelo grafo de RNFs. Ao mesmo tempo, a representação de conflitos torna-se mais fácil no grafo do que no MER-RNF pois representar esses conflitos iria poluir demais o MER.

Portanto conclui-se que, a cada alteração que se visualize durante a inclusão de RNFs no MER que implique em novos RNFs ou em conflitos entre RNFs já representados, deve-se voltar a representar essas alterações no grafo, e ainda voltar ao UdI para certificar-se que nada mais ficou escondido. É claro que, ao final, as versões do grafo de RNFs e do MER-RNF devem conter os mesmos RNFs (incluindo todas as suas submetas).

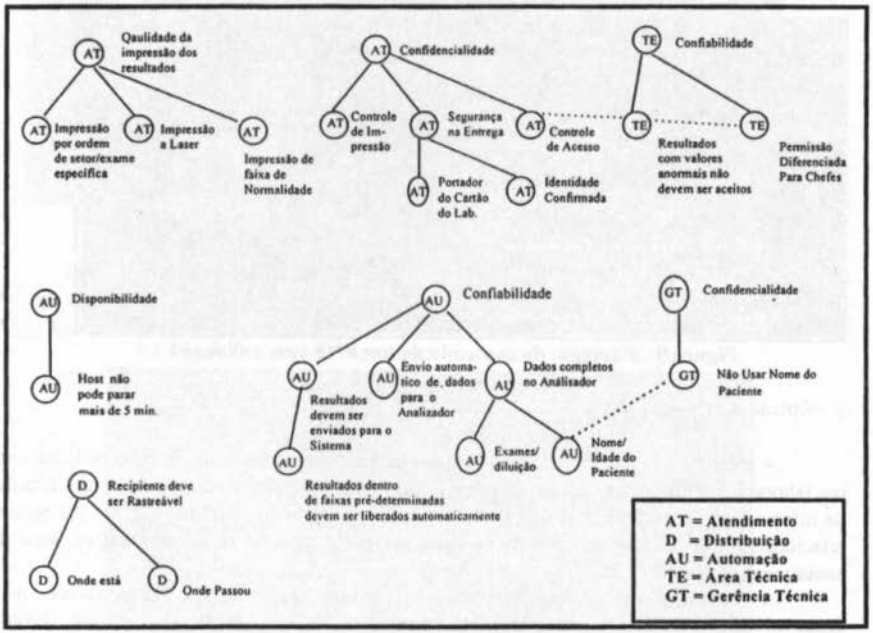

Figura 10 - Grafo de RNFs do estudo de caso

A figura 11 mostra parte do MER-RNF resultante de nosso estudo de caso. Nessa figura, as entidades que se destacam em negrito foram criadas ou alteradas devido a RNFs encontrados.

Quando da realização desse estudo de caso, ainda não estava operacional a extensão ao MER implementada no Talisman. Muitas das idéias utilizadas na linguagem MER-RNF vieram de necessidades observadas durante a confecção do MER de maneira manual como por exemplo, registrar as implicações de desenho que aquele RNF poderá trazer para o desenho do sistema e o grau de importância do mesmo.

A seguir são detalhados alguns pontos relevantes observados durante a construção do MER-RNF.

A entidade **Resultados**, onde encontramos um RNF **Confiabilidade** cujos refinamentos estão relacionados com um requisito funcional (possibilitar a passagem de resultados), indica que a passagem de resultado não deve permitir que um usuário digite

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompress

resultados que tenham sido considerados como anormais ou impossíveis. Por exemplo, uma dosagem de glicose de 16000 não pode ser aceita pois só em casos raros isso iria acontecer. Para esses casos, teremos um nível de acesso diferente para os chefes de setores, esses sim, sem limitação alguma na passagem de resultados. Nesse ponto o MER-RNF apresenta a entidade **Permissão de acesso** relacionada com **Funcionário**, que irá conter em seus atributos além do funcionário, as permissões necessárias para atender ao RNF **Confiabilidade** relacionado à ***Resultado***, que deverá contemplar tanto os programas aos quais o funcionário terá acesso, como também, que subexames ele pode ou não passar e os valores máximos e mínimos para esse funcionário em relação ao subexame em questão. Isso permitirá que um determinado funcionário possa entrar no programa de passagem de resultados, mas eventualmente não poderá alterar determinados exames daquele paciente, e outros exames ele poderá alterar desde que restrito a uma determinada faixa.

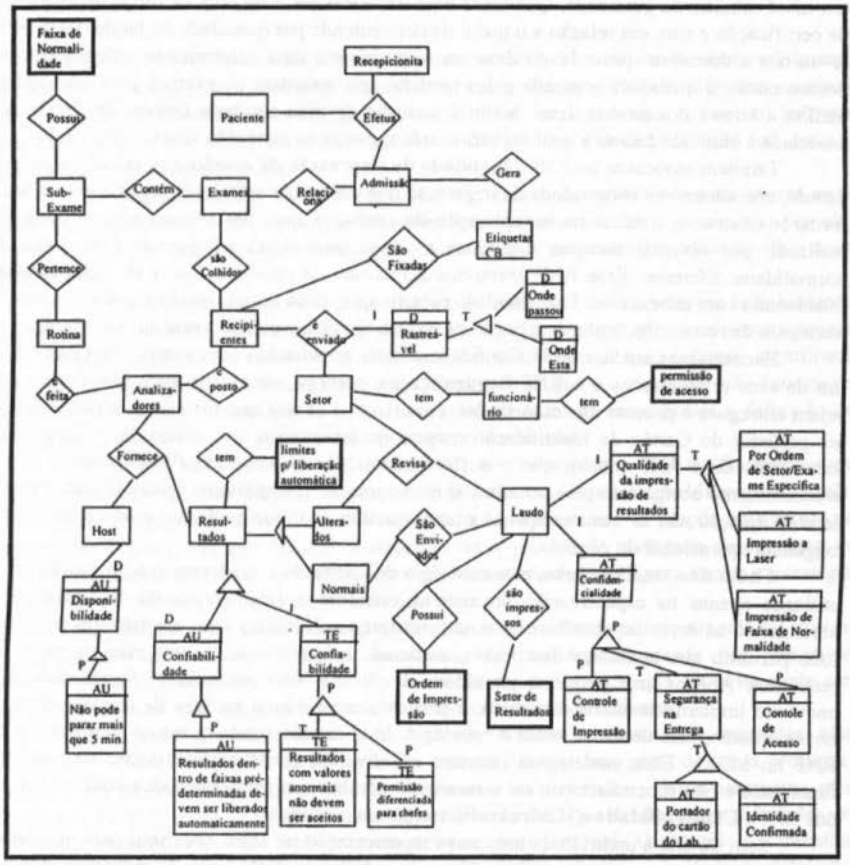

Figura 11 - MER-RNF do estudo de caso

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

Outro RNF que implicou em mudanças no modelo de dados é o **Rastreável,** que está associado com um relacionamento que denota o envio de recipientes aos setores, mostrando que esse envio deve ocorrer de tal forma que permita dizer por onde o recipiente passou e onde ele está no momento. Esse RNF irá certamente trazer conseqüências de desenho, pois teremos que especificar o relacionamento **"é enviado"** de forma a permitir localizar o recipiente. Isto implicará que a tabela que conterá os atributos desse relacionamento deverá ter campos como: Setor Origem, Setor Destino, Data, Hora entre outros.

A entidade ***Laudo*** está associada a dois RNFs, **Qualidade de impressão de resultados** e **Confidencialidade**. Aqui, mais uma vez, apesar de termos 2 requisitos não funcionais associados à uma só entidade, eles não são conflitantes e além disso são imprescindíveis para a satisfação do usuário. O RNF **Qualidade de impressão de resultados** refere-se à necessidade que o usuário possui, por questões de mercado, de que o laudo tenha um nível mínimo de qualidade. Qualidade aqui não deve ser vista com os olhos de processos de certificação e sim, em relação a o que o usuário entende por qualidade do laudo. Esse RNF levou-nos a descobrir que o laudo deve ser impresso em uma determinada ordem para que possua então, a qualidade esperada pelos médicos que solicitam os exames pois essa ordem facilita a leitura dos laudos. Isso levou à inclusão de uma entidade ***Ordem de Impressão*** associada à entidade ***Laudo*** a qual registra a ordem que os exames irão aparecer no laudo.

Também associado ao RNF **Qualidade de impressão de resultados**, relacionado com ***Laudo***, encontramos a necessidade da impressão dos valores de referência para cada exame de forma a orientar o médico na interpretação do laudo, já que, um mesmo exame pode ser realizado por diversas técnicas diferentes e cada uma delas irá possuir uma faixa de normalidade diferente. Esse RNF levou-nos à inclusão da entidade ***Faixa de Normalidade*** relacionada com subexames. Essa entidade permite que, tanto na impressão do laudo como na passagem de resultados, tenha-se a noção de quanto alterado (ou não) está aquele exame.

Encontramos ainda, o RNF Confidencialidade relacionado com a entidade ***Laudo*** cujo um de seus refinamentos é o RNF **Segurança na entrega**, necessário para que laudos não sejam entregues à pessoas não autorizadas. Para tanto, a pessoa que for buscar o laudo deverá ser portador do Cartão de Identificação próprio do laboratório, ou, apresentar a carteira de identidade. Esse RNF mostra que o atributo identidade na entidade ***Paciente*** deve ser de preenchimento obrigatório para possibilitar esse controle. É importante ressaltar que, o cartão de identificação não se resume apenas a uma questão de implementação, é na verdade, uma exigência operacional do negócio.

Como dito anteriormente, esta estratégia de elicitação e modelagem aqui proposta, foi utilizada apenas na especificação do sistema referente a área técnica do laboratório não abrangendo as áreas de atendimento e administrativas. Notamos que, a estratégia como um todo permitiu que tenhamos descoberto, modelado e implementado uma enorme parte dos requisitos. Apenas uma pequena quantidade de RNFs não conhecidos foram detectados enquanto implementávamos o sistema, e praticamente nenhum na fase de implantação. Em nossa opinião, isso deveu-se tanto à estratégia de elicitação usada quanto à modelagem dos RNF no MER. Essa modelagem permitiu em diversos casos a visualização de requisitos funcionais e não funcionais que até o momento não haviam sido detectados como nos casos dos RNFs **Confiabilidade** e **Rastreável** anteriormente citados.

Sem dúvida a inclusão de uma nova representação no MER contribui para dificultar a leitura do mesmo para sistemas de médio e grande porte. Entretanto, acreditamos que isso pode ser contornado particionando-se o MER conforme a necessidade de cada sistema. Além

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompr

disso; acreditamos que as contribuições positivas da inclusão dos RNFs no MER superaram amplamente as dificuldades provocadas pelo aumento do número de símbolos no MER.

Outra dificuldade encontrada foi a de relacionar os RNFs encontrados e representados no grafo de RNF, com o ponto no MER (entidade ou relacionamento) ao qual estaria associado. Utilizamos um processo de inspeção do grafo de RNF tentando relacionar o ator do UdI relacionado com o RNF em questão com as entidade e relacionamentos que por ventura se relacionassem a este ator. O RNF **Qualidade de impressão de resultados** por exemplo, facilmente pôde ser identificado com a impressão de laudos. Já o RNF **Confiabilidade** relacionado à automação não foi trivial. Nesse caso tivemos que procurar no MER qual a entidade que estava mais diretamente relacionada a este RNF, e encontramos a entidade resultados, que esta relacionado com a entidade analisadores.

6 - Conclusão

Requisitos não funcionais a muito são mencionados em métodos de desenvolvimento de software, nomes como: restrições de software, condições de contorno, objetivos e outros, são algumas das denominações utilizadas. Apesar disso, raramente encontra-se um software em que os requisitos não funcionais tenham sido levados em consideração ou mesmo elicitados de maneira adequada.

Essa negligência em tratar os requisitos não funcionais pode ter várias causas onde pode-se destacar o fato de, apesar de presentes em diversos métodos eles não são por eles tratados. Estão lá presentes mais como comentário do que propriamente como um fato a ser considerado.

Entretanto, com a evolução do hardware e do software e com o cliente se habituando ao uso de computadores, passamos a ter clientes mais exigentes, que não mais querem apenas correção funcional de um software, querem também qualidade em todo o amplo significado que esta palavra pode ter. Para isso, parece fundamental que os requisitos não funcionais sejam incorporados ao processo de desenvolvimento de software em especial à definição de requisitos. Justamente aqui reside nossa principal contribuição.

O trabalho aqui resumido [Cysneiros 97] contribui para a área de Engenharia de Requisitos nos seguintes pontos: a) propõe uma estratégia de elicitação de requisitos não funcionais acrescentando a entidade ator ao modelo de Mylopoulos; b) propõe uma forma de representação integrada que facilita os impactos de requisitos de qualidade no conjunto de uma especificação funcional; c) implementa uma ferramenta de CASE para apoiar a estratégia e d) valida a estratégia proposta com um estudo de caso real.

Os resultados até aqui alcançados são bastante positivos, principalmente no que se refere ao estudo de caso. A baixa quantidade de manutenção gerada por requisitos incompatíveis ou inexistentes, é um forte indício de que a estratégia proposta pode ser utilizada com bons resultados no processo de desenvolvimento de software.

Outros autores recentemente têm dedicado estudos ao tema dos requisitos não funcionais. Além do trabalho de Mylopoulos, que utilizamos como parte de nossa proposta, vale a pena destacar dois outros mais. O primeiro é o trabalho sobre requisitos não funcionais para sistemas de tempo real de Kinner [Kirner 96], que detalha as propriedades de 6 RNFs considerados os mais relevantes em sistemas de tempo real. O segundo é o trabalho de Sommerville e Kotonya [Kotonya 95], que propõe a utilização de um método de definição de requisitos orientado a pontos de vista, que tenta integrar os requisitos funcionais, requisitos

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

não funcionais e requisitos de controle numa mesma especificação, finalizando na geração de "Templates" de requisitos.

Nossa agenda de pesquisa inclui o estudo da aplicação dessa estratégia para ambientes de desenvolvimento que utilizam orientação a objetos, a aplicação da estratégia a outros casos e o estudo de um assistente inteligente para guiar a identificação de pontos onde se faz necessária a adição de requisitos não funcionais. Esta estratégia procurará mesclar a utilização de uma base de conhecimento de requisitos não funcionais de características gerais, com conhecimento de domínios

7 - Bibliografia

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

[Cysneiros 97] Cysneiros, L.M. "Integrando Requisitos Não Funcionais ao Processo de Desenvolvimento de Software" Dissertação de Mestrado PUC/Rio, 1997

[Dardenne 93] 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

[Franco 92] Franco Ana Paula M, "Métodos e representações de suporte à aquisição de linguagens da aplicação." Dissertação de Mestrado da PUC-Rio, abril 1992

[Goguen 93] Goguen, J. A., Linde, C., "Techiniques for Requirements Elicitation", First IEEE Int. Symposium on Requirements Engineering, IEEE Computer Society 1993, pp. 152-164

[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.

[Leite 91] Leite J.C.S.P. and Freeman, P. A. "Requirements Validation Through Viewpoint Resolution", IEEE Trans on Soft. Eng., Vol 17, No 12, Dec 1991, pp 1253-1269 .

[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.

[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.

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

[Staa 92] Staa Informatica "Meta ambiente de desenvolvimento assistido de software TALISMAN", Manual de referência. 1992.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompress