Processo de Engenharia de Requisitos Aplicado a Requisitos Não-Funcionais de Desempenho – Um Estudo de Caso

Denise Lazzeri Gastaldo, Edson Toshimi Midorikawa

Departamento de Engenharia da Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, Trav. 3, nº 158 - CEP 05580-900 – São Paulo – Brasil
{denise.gastaldo, edson.midorikawa}@poli.usp.br

Abstract. This paper presents a requirement engineering process applied to performance non-functional requirements, which are an important factor to get quality results in software projects. The process includes the elicitation, analysis, documentation and validation phases. Along with the process, methods like PLanguage (a specification language), non-functional requirements graphs (to represent knowledge) and an approach using UML use case diagrams (to join functional and non-functional views) are being included in this work.

Resumo. Este artigo apresenta um processo de engenharia de requisitos aplicado a requisitos não-funcionais de desempenho, que são um fator importante para obter resultados de qualidade nos projetos de software. O processo inclui as fases de elicitação, análise, documentação e validação. Juntamente com o processo, métodos como a PLanguage (uma linguagem de especificação), grafos de requisitos não-funcionais (para representar o conhecimento) e uma abordagem utilizando diagramas de casos de uso UML (para unir visões funcional e não-funcional) serão incluídos neste trabalho.

1 Introdução

A Engenharia de Requisitos tem se tornado cada vez mais necessária para resolver os problemas encontrados nas organizações com relação à definição de sistemas. De acordo com [6] requisitos deficientes são a maior causa de falhas nos projetos de software. Além disso, podemos melhorar o processo convencional utilizando práticas de Engenharia de Requisitos. Como resolver estes problemas e aplicar as práticas de Engenharia de Requisitos? Primeiramente devemos entender muito bem o domínio da aplicação e a interface que fica entre o mundo real e o mundo dos sistemas. Melhor será o sistema que conseguir se aproximar e reproduzir o mundo real [5].

Com base em estudos preliminares de casos reais em que os autores estão trabalhando, cerca de 50% do retrabalho[1] referente ao processo de desenvolvimento

[1] De acordo com [11] o retrabalho é definido como o custo das alterações, que inclui o esforço de analisar, resolver e retestar as mudanças aplicadas as *baselines* do software.

de software ocorre nas fases iniciais de elicitação, análise e documentação de requisitos, como mostra a Figura 1.

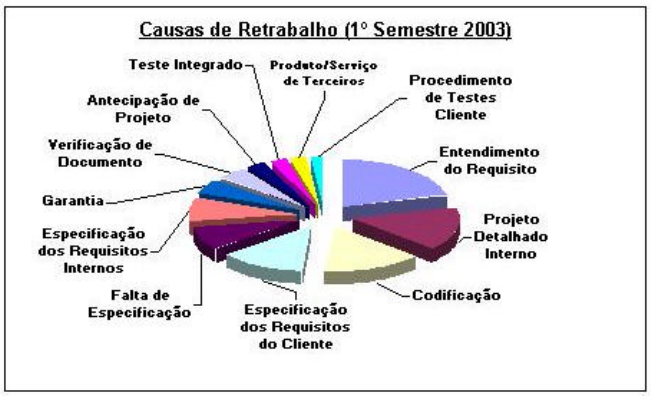

Fig. 1. Causas de retrabalho

O estudo realizado no levantamento do retrabalho mostrou que grande parte das causas serão relacionadas aos requisitos não-funcionais de desempenho, que não puderam ser atendidos por não terem sido incorporados desde o início do projeto. A razão desta afirmação é refletida no aumento de custo e prazos de entrega em conseqüência de mudanças de estratégia a fim de inserir um requisito não-funcional de desempenho não projetado. Muitas vezes, é necessário mudar a arquitetura de software, de hardware e até mesmo a concepção da solução do sistema para atender um requisito não-funcional de desempenho.

Uma das razões pelas quais os requisitos de desempenho não são definidos e analisados logo no início do projeto é o fato de que os requisitos não-funcionais não são cobertos pela maioria dos métodos de Engenharia de Requisitos [14]. Os métodos existentes normalmente estão baseados em análises funcionais ou orientados a objetos. A explicação para isso é a dificuldade em definir os requisitos não-funcionais, algumas das causas são apontadas como:

- Certas restrições somente são descobertas na fase de projeto;
- Outras restrições são altamente subjetivas somente podendo ser determinadas por avaliações empíricas;
- Os requisitos não-funcionais tendem a estar relacionados com mais de um requisito funcional, sendo assim é extremamente complicada expressar esta ligação;
- Não existem regras determinadas para expressar os requisitos não-funcionais.

Com o objetivo preencher a lacuna existente nas fases de elicitação, análise, documentação e validação dos requisitos de desempenho está sendo realizado o presente trabalho, que define um processo de Engenharia de Requisitos aplicado a requisitos não-funcionais de desempenho. Este processo visa dar os subsídios necessários para que estes requisitos sejam incorporados ao software nas fases iniciais, direcionando a solução e a forma de condução e gerenciamento dos projetos.

Métodos de representação gráfica de requisitos não-funcionais [1], uma linguagem de especificação de requisitos denominada *PLanguage* [12], casos de uso UML [3], [8], metodologias gerais aplicadas a requisitos não-funcionais [2], metodologias específicas aplicadas a requisitos de desempenho [10], [13] e conceitos de engenharia de requisitos [14] serão utilizados como suporte ao processo que está sendo definido.

O trabalho será estruturado em seções: a próxima seção trata dos principais conceitos utilizados. A seção 3 apresenta o processo de engenharia de requisitos aplicado a requisitos não-funcionais de desempenho. O estudo de caso em que o processo foi aplicado e validado é descrito na seção 4. E finalmente, a seção 5 apresenta a conclusão.

2 Requisitos Não-Funcionais de Desempenho

Requisitos não-funcionais de desempenho são aqueles que se referem à velocidade de operação do sistema. De acordo com [14] diferentes tipos de requisitos podem ser especificados, são eles:

- Tempo de Resposta, que especifica o tempo de resposta aceitável do ponto de vista do cliente para que alguma operação seja concluída;
- *Throughput*, que especifica a quantidade de dados que precisam ser processados em um intervalo pré-definido de tempo;
- Temporização, que especifica quão rápido o sistema precisa coletar uma entrada proveniente de sensores antes que sejam sobrescritos por outras entradas da mesma natureza, ou quão rápido o sistema precisa realizar o processamento para que seja a entrada de outro subsistema.

De acordo com [7] existem dois tipos de requisitos de desempenho: estáticos e dinâmicos. Os estáticos são aqueles que impõe restrições para que o sistema seja executado, incluindo por exemplo o número de terminais suportados, o número de usuários simultâneos, o número de arquivos que o sistema deve processar e seu tamanho em bytes. Estes requisitos também podem ser chamados de requisitos de capacidade ou espaço.

Os requisitos de desempenho dinâmicos especificam as restrições do comportamento do sistema, que tipicamente inclui o tempo de resposta e o *throughput*.

A abordagem utilizada no presente trabalho foi proposta por [9], que alerta que o desempenho afeta a usabilidade de um sistema. Se o sistema de software é lento, ele reduz a produtividade a ponto de não atender às suas necessidades. Além disso, se o sistema requer muito espaço em disco, pode ser oneroso utilizá-lo. Em contrapartida, se o sistema exige muita memória, este pode afetar outras aplicações que estão sendo executadas no mesmo sistema, ou pode se tornar tão lento que o sistema operacional precise utilizar técnicas de escalonamento de tarefas. Desta forma, [9] propõe uma classificação de requisitos não-funcionais de desempenho em requisitos de tempo e espaço. O requisito de tempo de resposta se subdivide em requisitos de tempo de resposta e de processamento (*throughput*) e o requisito de espaço se subdivide em memória principal e memória secundária.

Os requisitos de desempenho estão intimamente associados à arquitetura de software, assim os mecanismos de comunicação têm influência direta sobre o desempenho obtido.

Portanto, este tipo de requisito deve ser especificado em termos quantitativos para que possam ser testados e validados. Não é desejável que se tenha requisitos do tipo "o tempo de resposta deve ser bom" e "todas as transações devem ser feitas rapidamente". Ao invés destes requisitos o desejável seria escrevê-los da seguinte forma: "o tempo de resposta ao comando x deve ser menor que um segundo em 90% dos casos" e "transações devem ser processadas em menos de um segundo em 98% dos casos", desta forma as especificações ficam muito mais claras.

3 Processo de Gerenciamento de Requisitos Não-Funcionais de Desempenho

Um dos grandes desafios encontrados no desenvolvimento de software atualmente é o gerenciamento dos requisitos de desempenho [10]. Muitas organizações almejam obter melhores desempenhos de suas aplicações, mas não possuem métodos incorporados ao seu ciclo de vida de desenvolvimento que auxiliem a atingir este objetivo.

O gerenciamento de requisitos não-funcionais de desempenho não é trivial, uma vez que mal dimensionado pode impactar no resultado final do sistema como um todo. O atendimento a um requisito não-funcional de desempenho pode resultar na alteração de diversos componentes o que pode sair dispendioso, pois não é possível simplesmente incluir um "*componente de desempenho*" no modelo do sistema.

O desempenho do software além de estar relacionado com a arquitetura, está relacionado com o método de implementação escolhido pelo desenvolvedor. Para auxiliar na fase de projeto de software, um conjunto de princípios pode ser encontrado no SPE ou S*oftware Performance Engineering* [13]. Estes princípios são utilizados para que depois de especificados, os requisitos de desempenho possam ser incorporados ao software para que sejam atendidas as necessidades do usuário final.

Nesta seção será apresentado o processo que engloba todas as fases da engenharia de requisitos aplicada a requisitos não-funcionais de desempenho a fim de estabelecer um padrão, para que a partir da saída deste processo os requisitos de desempenho possam ser gerenciados.

3.1 Processo de Engenharia de Requisitos para Requisitos Não-Funcionais de Desempenho

De acordo com [2] existem duas visões de desenvolvimento de projeto, a visão funcional e a visão não-funcional. Cada visão possui seu ciclo independente com pontos de convergência. Como mostra a Figura 2, estes pontos de convergência refletem, na visão funcional, todas as ações e os dados necessários para que os requisitos não-funcionais sejam atendidos. A definição do método de engenharia de requisitos aplicado a requisitos não-funcionais de desempenho é uma adaptação da

proposta de [2] como pode ser verificado na Figura 2. A proposta de [2] estabelece um processo de elicitação e definição de modelos conceituais de projeto baseados em requisitos não-funcionais, integrando diagramas UML de casos de uso, classes, seqüência e colaboração. Inclui também uma ferramenta de auxílio para registro do léxico da linguagem utilizada para registrar o domínio da aplicação. Mas, em contrapartida, não define um processo completo (com entradas e saídas) de engenharia de requisitos incluindo as fases de análise, documentação e validação.

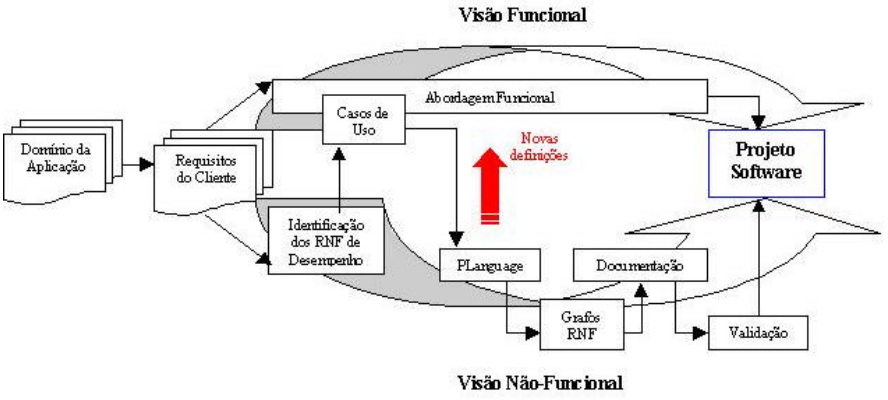

Fig. 2. Processo de engenharia de requisitos aplicado a requisitos não-funcionais de desempenho

No processo da Figura 2, o ciclo para ambas as visões, não-funcional e funcional, inicia-se com o entendimento *domínio da aplicação*, fase que caracteriza o processo de ***elicitação de requisitos*** da engenharia de requisitos. De acordo com [5] as máquinas foram criadas para prover funcionalidades similares às do mundo real, sendo que estes aspectos funcionais são relevantes para que seja possível conhecer o domínio da aplicação. Desta forma, a máquina e o mundo real possuem uma interface em comum. A interface entre o mundo real e a máquina é formada pelos requisitos que os sistemas devem atender; por sua vez, as especificações de requisitos são as descrições desta interface. Assim, o domínio da aplicação é a propriedade do mundo real que conhecemos e que consideramos verdadeiras, e os requisitos de um sistema são as propriedades do mundo que se deseja atingir através de um sistema computacional. O domínio da aplicação é a conexão entre o que a máquina deve fazer e os requisitos da aplicação.

Somente com o entendimento do domínio da aplicação será possível definir os requisitos de um sistema. De acordo com [5] muitos erros encontrados nos requisitos são provenientes da falta de entendimento do domínio da aplicação.

O passo seguinte do ciclo é a definição dos requisitos do cliente que retratam o entendimento do domínio da aplicação aplicado à visão sistêmica. Este passo também é comum para as visões funcional e não-funcional e também está caracterizado dentro da fase de **elicitação de requisitos** do processo de engenharia de requisitos.

Com o entendimento do domínio da aplicação e a definição dos requisitos do cliente é possível fazer a identificação dos requisitos não-funcionais de desempenho, que consiste na identificação de características de desempenho dentro do conjunto de requisitos definidos.

Após a identificação dos requisitos não-funcionais de desempenho ocorre a definição dos casos de uso, iniciando a fase de **análise** do processo de engenharia de requisitos. Os casos de uso capturam as funcionalidades básicas do sistema de uma maneira facilmente entendida pelos usuários não técnicos. Pode ser utilizado para representar todo o sistema e/ou partes do sistema. Os casos de uso são representados por atores, que interagem com o sistema e por funcionalidades realizadas pelos atores [3] e podem ser aplicados nas fases de análise, projeto, implementação e testes.

Desta forma, é possível caracterizar uma das atividades da fase de ***análise*** do processo de engenharia de requisitos, pois com a utilização dos casos de uso é possível identificar além das funcionalidades do sistema os pontos de negociação e novos cenários a visão funcional.

A fase de ***análise*** se estende nas atividades de utilização da linguagem de especificação *PLanguage* e utilização dos *grafos de requisitos não-funcionais*. A utilização destes métodos auxilia o entendimento mais aprofundado do sistema permitindo a definição da solução computacional nos projetos.

A utilização da *PLanguage* e dos *grafos de requisitos não-funcionais* pertence à fases de ***análise*** *e* serão descritas nas seções seguintes. A ***documentação***, próxima fase do processo de engenharia de requisitos, será composta pela saída da fase de análise que irá gerar diagramas de caso de uso, descrição dos casos de uso, descrição dos requisitos em *PLanguage* e representação dos requisitos não-funcionais através dos grafos. A fase de ***validação*** será composta pela aceitação do documento gerado na fase de ***documentação*** pelo cliente interno e externo, liberando as especificações dos requisitos não-funcionais de desempenho para a fase de ***projeto*** detalhado.

3.2 PLanguage

A *PLanguage* ou *"Planning Language"* [12] permite a medida e o teste da qualidade dos requisitos não-funcionais especificados e traz benefícios como facilidade de aprendizado, flexibilidade, além de ser compacta e prevenir omissões através de um conjunto consistente de parâmetros de qualidade que podem ser utilizados nas especificações.

Sua forma de apresentação é composta por um conjunto de palavras chave nas quais os requisitos devem ser especificados. Seu formato, atributos e conceitos são apresentados na Tabela 1.

Table 1. Definição das palavras-chave da linguagem de especificação PLanguage

Palavras Chave	Descrição
TIPO	Etiqueta, rótulo, identificador persistente e único do requisito.
DESCRIÇÃO	Descrição simples e breve do conceito principal ou significado geral do requisito.
STAKEHOLDER	Envolvidos/Afetados pelo requisito
ESCALA	Escala usada para quantificar o requisito.
MÉTRICA	Processo ou método para medir escalas dos requisitos.
MÉTODO	Método para medir a escala.

Palavras Chave	Descrição
FREQUENCIA	Freqüência para medição.
RESPONSÁVEL	Pessoas/Departamento responsável por fazer as medições.
REGISTRO	Onde/Quando as medidas devem ser reportadas.
NÍVEL MÍNIMO	O nível mínimo requerido para evitar falhas.
PLANO	Nível para obter sucesso exigido.
NÍVEL SUCESSO	Como prolongar, aumentar, alongar o sucesso.
NÍVEL DESEJADO	Nível desejável de sucesso que não pode ser atingido através dos métodos atuais.
HISTÓRICO	Resultados anteriores para comparação (histórico).
TENDÊNCIA	Tendência histórica.
HISTÓRICO DE SUCESSO	O melhor resultado obtido.
DEFINIÇÃO	Definição oficial do termo.
AUTORIDADE	Pessoa, grupo ou nível de autorização.

Com a utilização deste *template* de especificação será possível resolver os problemas de falta de método para o levantamento de requisitos, falta de clareza na especificação, falta de detalhamento, confusão entre requisitos funcionais e não-funcionais, falta de formatação e apresentação. O formato da *PLanguage* é claro para qualquer integrante do projeto e, além dos benefícios citados, os testes podem ser feitos com base na própria especificação, eliminando o trabalho da elaboração de um documento de especificação e outro de testes.

A *PLanguage* pode ser uma das entradas para a visão funcional, uma vez que pode esclarecer características fundamentais dos requisitos não-funcionais fornecendo desta forma novas definições e informações.

3.3 Grafos de Requisitos Não-Funcionais

A utilização de grafos de requisitos não-funcionais foi uma alternativa encontrada por [1] para resolver problemas de projeto em conseqüência da pouca importância dada aos requisitos não-funcionais nas fases iniciais do projeto.

A proposta trata os requisitos não-funcionais como *objetivos* a serem atingidos e representando alternativas para tomada de decisões de projeto e arquitetura. Os grafos, em forma de árvores de derivações, utilizam ligações entre os *objetivos* principais que retratam as necessidades dos clientes e os *sub-objetivos* que retratam as funcionalidades que devem ser implementadas a fim de atender aos *objetivos* não-funcionais do projeto.

O primeiro passo na utilização deste método é a obtenção de uma representação organizada do conhecimento de um requisito não-funcional específico. No caso do presente trabalho, será utilizada para a representação de requisitos não-funcionais de desempenho.

De acordo com [1] existem três tipos de métodos que podem ser representados nos grafos: os métodos de *decomposição* que refinam ou esclarecem aspectos dos requisitos não-funcionais em questão, os métodos de *satisfação* que visam atender aos requisitos não-funcionais e os métodos de *operacionalização* que são os meios práticos pelos quais os *objetivos* serão atendidos.

Outros tipos de representações foram propostas a fim de estender o modelo inicial, uma delas proposta por [1] define dois tipos de *operacionalizações*, a *estática* e a *dinâmica*. A *operacionalização estática* expressa a necessidade da utilização de dados

para que um requisito não-funcional seja atendido ou mesmo para atender soluções de projeto. A *operacionalização dinâmica* representa as ações que devem ser implementadas para que os objetivos sejam atingidos. Esta abordagem permite fazer o refinamento do diagrama de classes tornando-o mais completo e com a garantia de que exista a preocupação com o requisito não-funcional no decorrer do projeto.

De acordo com [10] os grafos de requisitos não-funcionais são elaborados de acordo com camadas definidas na fase de análise de projeto. Estas camadas são uma forma de organizar os requisitos não-funcionais de desempenho de forma que cada camada seja uma fonte de tomada de decisões com suas entradas e saídas. O presente trabalho estende esta abordagem, compatibilizando a utilização dos diagramas UML de casos de uso como o método de organização do conhecimento e funcionalidades do sistema e como uma forma de relacionar os requisitos funcionais aos não-funcionais.

Desta forma a utilização dos grafos de requisitos não-funcionais está intimamente relacionada com a definição de casos de uso, sendo que cada um deles terá além de seu contexto funcional o seu contexto não-funcional. A aproximação das duas visões melhora o entendimento dos *objetivos* dos requisitos não-funcionais que devem ser atingidos. A Figura 3 mostra um grafo de requisitos não-funcionais específico para requisitos não-funcionais de desempenho. Com base neste grafo serão estendidos os modelos utilizados no estudo de caso.

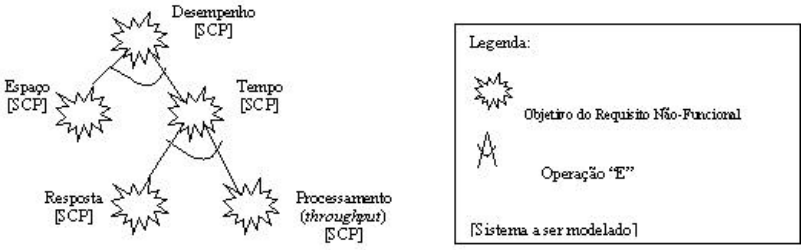

Fig. 3. Grafo de requisitos não-funcionais de desempenho

A partir do grafo da Figura 3, entende-se que o desempenho desejado será obtido através do atendimento dos requisitos de espaço **e** dos requisitos de tempo. O requisito de tempo, por sua vez, só será atendido se os requisitos de tempo de resposta **e** de tempo de processamento (*throughput*) forem satisfatórios.

Para a finalidade do trabalho serão utilizados somente os símbolos[2] apresentados na Figura 3, que serão suficientes para a representação nas fases durante o processo de engenharia de requisitos.

[2] O modelo completo pode ser verificado em [1], [10] e [2].

4 Estudo de Caso – Subsistema de Controle de Painéis (SCP)

O estudo de caso utilizado para a aplicação do processo de engenharia de requisitos para requisitos não-funcionais de desempenho será o de um subsistema de controle de painéis luminosos, que tem a finalidade de exibir mensagens aos usuários para orientação, propaganda e avisos de condições anormais. Este projeto está em desenvolvimento e é parte integrante de um sistema de controle de tráfego.

As principais funcionalidades são a veiculação e armazenamento de mensagens, escalonamento de prioridades, programação de mensagens que são exibidas ciclicamente, controle do diagnóstico de comunicação além do gerenciamento da base de dados de mensagens. Além disso, é possível aplicar efeitos de rolagem e muitos tipos de letras. Maiores detalhes da especificação podem ser vistos em [4].

Em termos de arquitetura de comunicação, os painéis possuem uma interface com um concentrador de painéis que recebe comandos de vários subsistemas que desejem veicular mensagens. Estes painéis são compostos por uma matriz de *leds* luminosos monocromáticos vermelhos e pode ser visualizado na Figura 4, juntamente com um exemplo de mensagem sendo exibida.

Fig. 4. Painel de mensagens

Este subsistema possui requisitos não-funcionais de desempenho, uma vez que as mensagens precisam chegar dentro de um intervalo de tempo aceitável para que o usuário tenha a informação atualizada. A emissão de mensagens para os painéis de aviso é resultado do processamento de vários subsistemas que é o de controle e supervisão, sendo que o gerenciamento de quando e quais mensagens serão exibidas depende do concentrador de painéis que tem a função de organizar a exibição de mensagens no momento e no painel correto. Nesta seção serão descritos os passos do processo utilizado juntamente com a descrição mais detalhada das funcionalidades do subsistema em questão.

4.1 Tratamento dos Requisitos Não-Funcionais de Desempenho

A partir do entendimento do domínio da aplicação e da definição dos requisitos do cliente, os requisitos não-funcionais de desempenho foram identificados, sendo que para isso foi feita uma tabela de requisitos utilizando uma ferramenta de apoio. O conjunto de requisitos não-funcionais de desempenho foi classificado de acordo com a conceituação de [9], recebendo também um código de identificação.

De acordo com os requisitos levantados e com as necessidades do cliente, foram definidos casos de uso para o subsistema de controle de painéis seguindo diretrizes definidas por [8] de acordo com uma seqüência de passos citados na Tabela 2.

A partir destas diretrizes foi identificado o digrama de contexto do subsistema de controle de painéis, definido na Figura 5. Os atores que interagem com o subsistema são os Operadores Local e Central.

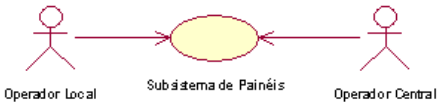

Fig. 5. Diagrama de Contexto do SCP

O diagrama detalhado foi definido depois de identificados os atores, sendo que cada caso de uso descrito na Figura 6 possui uma letra que o identificará nas outras fases do processo do processo de engenharia de requisitos.

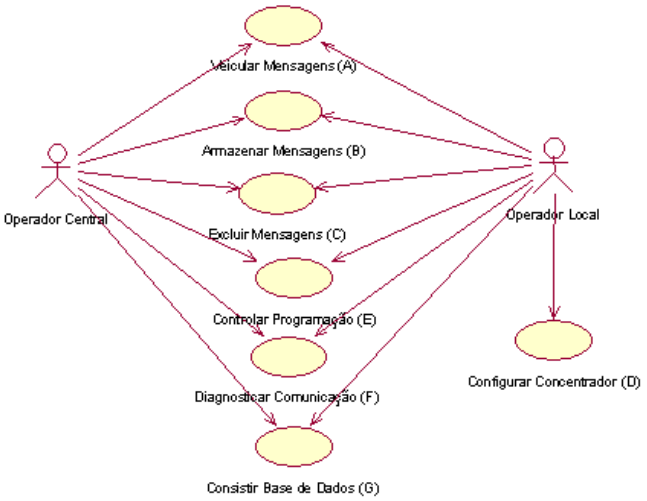

Fig. 6. Diagrama de casos de uso detalhado

As Tabelas 2 e 3 apresentam além dos passos apresentados para definição e descrição de casos de uso, diretrizes sobre como realizar estas atividades.

Table 2. Definição de casos de uso

Os seguintes passos devem ser adotados para se definir um caso de uso: 1. Identificar os grupos de usuários participantes do sistema a ser construído; 2. Definir os papéis dos grupos e, com isto, identificar os atores; 3. Identificar o evento iniciador para cada interação do ator com o sistema; 4. Identificar a ação do sistema para cada evento iniciador; estas ações são os casos de uso; 5. Classificar os Casos de Uso, identificando os relacionados com o negócio e os relacionados com a infraestrutura; 6. Classificar os Casos de Uso, identificando os mais críticos (os mais complexos, os mais desconhecidos, os mais importantes para o negócio, etc.); 7. Descrever sucintamente os casos de uso; 8. Identificar as execuções alternativas para cada caso de uso; 9. Analisar os casos de uso e identificar as partes semelhantes, para separar como casos de uso a serem incluídos.

Depois de definidos os casos de uso e seu diagrama detalhado, foram feitas as suas descrições de acordo com uma diretriz definida por [8] que é apresentada na Tabela 3.

Table 3. Descrição de casos de uso

Os seguintes passos devem ser adotados para se descrever um caso de uso: 1. Identificar os limites do Caso de Uso proposto, baseado na aplicação do sistema; 2. Identificar as entidades que estão fora dos limites e que interagem diretamente com o sistema; 3. Verificar se as entidades identificadas podem ser representadas pelos atores já definidos; se não, incluir os atores faltantes; 4. Determinar as condições iniciais que dão início ao Caso de Uso; 5. Determinar a conclusão lógica de cada uma das transações.; 6. Descrever o objetivo do Caso de Uso; 7. Identificar a seqüência de interações que ocorre numa transação normal para a operação do sistema. Estabelecer as regras de escolha entre as variações que possam ocorrer e as interações; 8. Considerar todas as exceções que possam ocorrer durante uma transação e especificar como estas exceções podem afetar o Caso de Uso.

Um exemplo de descrição de caso de uso, de acordo com a diretriz apresentada, pode ser visualizada na Tabela 4. O mesmo processo de descrição foi utilizado para os demais casos de uso.

Table 4. Descrição do caso de uso veicular mensagens (A)

(A)Veicular Mensagens: ▪ Caso de Uso A: Veicular Mensagens. ▪ Descrição: Este caso de uso descreve o processo de veiculação de mensagens nos painéis. ▪ Evento Iniciador: Solicitação de exibição de mensagem no painel. ▪ Atores: Operador Local e/ou Operador Central. ▪ Pré-Condição: Mensagem armazenada previamente na base de dados. ▪ Sequência de Eventos: i. Operador abre menu e seleciona opção para veicular mensagem; ii. Sistema apresenta opções de mensagens a serem veiculadas; iii. Operador escolhe mensagem a ser exibida; iv. Sistema solicita inserção da prioridade da mensagem; v. Sistema apresenta opções de destino da mensagem; vi. Operador escolhe o destino da mensagem; vii. Operador emite o comando para veiculação da mensagem; viii. Sistema confirma a operação; ▪ Pós-Condição: Mensagem exibida no painel. ▪ RNFD: DES-004, DES-005, DES-0017, DES-0016.

O próximo passo foi a associação de cada requisito não-funcional de desempenho aos casos de uso definidos, como mostra a Tabela 5. Com esta associação se percebeu que cada caso de uso possui uma característica de desempenho diferente e sendo assim, deve ser gerenciado de uma forma distinta, a Tabela 6 ilustra a afirmação.

Table 5. Requisitos não-funcionais de desempenho do subsistema de paíneis associados aos casos de uso

Requisito	Tipo	Caso de Uso
DES-004	Tempo de Resposta	A
DES-005	Tempo de Resposta	A
DES-007.1	Espaço	B
DES-007.2	Tempo de Resposta	C
DES-0011	Processamento	C
DES-0014	Espaço	B
DES-0015	Tempo de Resposta	A
DES-0016	Processamento	C
DES-0017	Tempo de Resposta	A
DES-0018	Tempo de Resposta	C
DES-0022	Espaço	E
DES-0026	Processamento	A
DES-0027	Espaço	B
DES-0028	Espaço	B
DES-0029	Tempo de Resposta	F
DES-0030	Tempo de Resposta	G
DES-0039.1	Tempo de Resposta	E
DES-0039.1	Espaço	E

A identificação dos requisitos não-funcionais de desempenho e a elaboração dos casos de uso do subsistema permitem fazer a ligação entre os requisitos funcionais e não-funcionais, analisar os impactos que os requisitos não-funcionais representam no projeto, identificar o tipo do caso de uso de acordo com sua característica de desempenho, facilitar a atribuição de responsabilidades para profissionais especialistas de acordo com a característica do caso de uso permitindo a utilização da técnica de SPE adequada na implementação do software.

Table 6. Característica de desempenho de cada caso de uso

Caso de Uso	Tipo
A	Tempo de Resposta, Processamento
B	Espaço
C	Tempo de Resposta, Processamento
D	NA
E	Tempo de Resposta, Espaço
F	Tempo de Resposta
G	Tempo de Resposta

4.2 Utilização da PLanguage e Grafos de RNF

A PLanguage foi aplicada para todo o conjunto dos requisitos não-funcionais de desempenho e permitiu, além do detalhamento das características dos requisitos, o registro do domínio do problema através do campo DEFINIÇÃO. Um exemplo pode ser visualizado na Tabela 7.

Além das características da linguagem de especificação foi possível identificar claramente o tipo do requisito, melhorar o entendimento das necessidades do cliente, definir parâmetros de teste, definir métricas, definir responsabilidades e um dos pontos mais importantes foi a possibilidade de aplicar padrões de medição para os requisitos não-funcionais de desempenho de tempo e espaço.

Table 7. Descrição de requisitos em Planguage

Código: DES-004/A	**Requisito:** As IHMs do Controle Central poderão solicitar a veiculação mensagens.
PLanguage	**Descrição**
TIPO	Tempo (tempo de resposta).
DESCRIÇÃO	Comando deve ser aplicado dentro de um intervalo de tempo pré-determinado.
STAKEHOLDER	Cliente, Responsável Técnico, Coordenador de Software, Desenvolvedor, Garantia da Qualidade.
ESCALA	Segundos.
MÉTRICA	Medidas de tempo (em segundos) de envio de comandos.
MÉTODO	Inclusão de marcadores de tempo no código.
FREQUÊNCIA	Para qualquer comando de solicitação de veiculação de mensagens.
RESPONSÁVEL	Desenvolvedor de Software.
REGISTRO	Banco de Dados com arquivo de *log* com o seguinte padrão de nome: BD_001_004A (BD_código da métrica_código do requisito).
NÍVEL MÍNIMO	Cada comando não pode demorar mais que 1 segundo (dada uma rede ethernet com tráfego de 10 Mbps).
PLANO	Monitoração através do software de controle, sendo que a qualquer índice de tempo maior que o esperado deve ser emitido um alarme e o arquivo de *log* deve ser analisado.
NÍVEL DE SUCESSO	Processo constante de monitoramento.
NÍVEL DESEJADO	Melhorar a capacidade da rede para que os comandos possam ser aplicados em menos de 1 segundo.
HISTÓRICO	Medidas anteriores chegaram a 2/3 segundos para a aplicação de um comando.
TENDÊNCIA	Melhoria de desempenho de acordo com as ações planejadas.
HISTÓRICO DE SUCESSO	Comandos aplicados em milisegundos.
DEFINIÇÃO	Controle Central é o conjunto de equipamentos localizados no mesmo espaço físico que possuem gerenciamento do sistema como um todo.
AUTORIDADE	Responsável Técnico e Garantia da Qualidade.

Com a definição detalhada dos requisitos utilizando a PLanguage foi feita a representação através da utilização dos grafos de requisitos não-funcionais. Os grafos construídos foram separados logicamente de acordo com cada caso de uso, desta forma, cada caso de uso possui o seu comportamento não-funcional de desempenho representado de forma gráfica. Um exemplo de grafo de requisitos não-funcionais de desempenho é apresentado na Figura 7, que retrata o comportamento do caso de uso A.

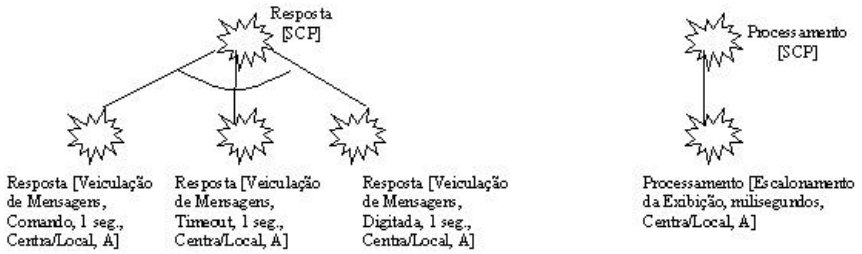

Fig. 7. Grafo de requisitos não-funcionais do caso de uso A

Cada nó do grafo representa uma funcionalidade que deve ser tratada levando-se em consideração o tipo do requisito não-funcional de desempenho que deve ser atendido. No exemplo da Figura 7 para se atender ao tempo de resposta, é necessário que se tenha a funcionalidade de veiculação de mensagens no painel do tipo comando, executada em até 1 segundo tanto para o modo Central como para o modo Local de operação. Da mesma forma devem ser atendidos os requisitos para as funcionalidades de veiculação de mensagens por *timeout* e digitada.

Os grafos de requisitos não-funcionais fazem a representação da PLanguage de forma clara e objetiva, sendo muito eficaz. Além disso, as funcionalidades descritas nos nós dos grafos ajudam na definição dos diagramas de classes nas fases seguintes de projeto, facilitando a obtenção de métodos que atendam aos requisitos. Outro ponto detectado foi a visualização clara das metas de desempenho que cada funcionalidade deve cumprir.

Após a execução das atividades propostas, resta estruturar um documento com todas as informações obtidas. O presente trabalho não tem como objetivo propor uma estrutura de documento, mas sim sugerir a inclusão dos diagramas, especificações em PLanguage e grafos aqui gerados. A itemização do documento deve ficar a cargo de cada organização de acordo com sua padronização interna.

A fase de validação que completa o processo de engenharia de requisitos deve ser feita através da aceitação do documento de especificação pelos clientes internos e externos, somente com o envolvimento de todos os *stakeholders* a especificação aprovada poderá servir como base para as fases seguintes de projeto.

5 Conclusão

O trabalho realizado incluindo a definição de um processo de engenharia de requisitos aplicada a requisitos não-funcionais de desempenho pode ser utilizado em qualquer contexto de projeto e organização.

A integração das visões funcional e não-funcional é importante para que se obtenha uma solução completa para projetos de software, que é representada pela utilização dos diagramas de casos de uso juntamente com suas descrições. A definição dos tipos de casos de uso de acordo com suas características de desempenho é relevante para os aspectos de gerenciamento e implementação da solução. Além disso, a aplicação de

diagramas UML no contexto dos requisitos não-funcionais é um passo fundamental para que se tenha a padronização do conhecimento entre clientes internos e externos, uma vez que UML tem se tornado uma linguagem comum para representar modelos de projetos de software.

Desta forma a utilização de uma linguagem de especificação para melhor definição de requisitos torna-se fundamental para integrar o modelo UML aos requisitos de desempenho. A definição dos requisitos não-funcionais fica clara e é fundamental para que o produto final a ser desenvolvido tenha qualidade. A utilização dos grafos de requisitos não-funcionais de desempenho expressa claramente as definições estabelecidas pela linguagem de especificação *PLanguage* e define os objetivos a serem atingidos pelos requisitos na medida em que estabelece seus objetivos.

Outro ponto que merece destaque é a aplicabilidade do processo de Engenharia de Requisitos especificamente para requisitos não-funcionais de desempenho que configura todos os passos a serem seguidos para que se tenha um resultado final satisfatório incluindo regras e boas práticas.

Referências

1. Chung, L.; Nixon B.A.; Yu, E.(1995)"Using Non-Functional Requirements to Systematically Select Among Alternatives in Architectural Design", In: Proceedings of ICSE17 Workshop on Architectures for Software Systems, pp.31-43.
2. Cysneiros, L.M. e Leite, J.C.S.P. (2001) "Using UML to Reflect Non-Functional Requirements", In: Proceedings of the CASCON 2001.
3. Douglass, B. P., Real-Time UML – Developing Efficient Objects for Embedded Systems. Addison Wesley, 1998.
4. Gastaldo, D.L. Especificação Técnica de Engenharia – Subsistema de Controle de Painéis, 2003.
5. Hammond, J.; R., R. e Hall, A. (2001) "Will it work?", In: Proceedings Fifth IEEE International Symposium on Requirements Engineering, pp. 102-109.
6. Hofmann, H.F. e Lehner, F. (2001) "Requirements engineering as a success factor in software projects", IEEE Software, pp. 58-66 vol. 18, n. 4.
7. Jalote, P. An Integrated Approach to Software Engineering, Springer, 1997.
8. Melnikoff, S.; Filgueiras, L. Definição de Casos de Uso. Relatório Técnico, Escola Politécnica da Universidade de São Paulo, 2002.
9. Mendes, A., Arquitetura de Software, Campus, 2002.
10. Nixon, B.A. (2000) "Management of Performance Requirements for Information Systems", In: Proceedings of the IEEE Symposium on Software Engineering, pp. 1122-1145.
11. Royce, W., Software Project Management, Addison Wesley, 2001.
12. Simmons, E. (2001) "Quantifying Quality Requirements Using *PLanguage*". In: 14th International Software/Internet Quality Week. San Francisco, CA.
13. Smith, C.U., Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software, Addison Wesley, 2001.
14. Sommerville, I. e Kotonya, G., Requirements Engineering, Wiley, 1998.