SISTEMA DE APOIO À ESPECIFICAÇÃO DE REQUISITOS NÃO FUNCIONAIS

Trabalho de Conclusão de Curso

Engenharia da Computação

Diogo Leal Pinto Marvão
Orientadora: Profa. Maria Lencastre Pinheiro de Menezes e Cruz

Universidade de Pernambuco
Escola Politécnica de Pernambuco
Graduação em Engenharia de Computação

DIOGO LEAL PINTO MARVÃO

SISTEMA DE APOIO À ESPECIFICAÇÃO DE REQUISITOS NÃO FUNCIONAIS

Monografia apresentada como requisito parcial para obtenção do diploma de Bacharel em Engenharia de Computação pela Escola Politécnica de Pernambuco – Universidade de Pernambuco.

Recife, junho de 2015.

De acordo

Recife

____/_____/_____

_____
Orientador da Monografia

“O degrau de uma escada não serve simplesmente para que alguém permaneça em cima dele, destina-se a sustentar o pé de um homem pelo tempo suficiente para que ele coloque o outro um pouco mais alto.”

Thomas Huxley

Agradecimentos

Agradeço, em primeiro lugar, a meus pais e minha família que sempre investiram e acreditaram em mim.

Aos parceiros e companheiros de faculdade, cujos quais facilitaram imensamente minha caminhada, dedico meu agradecimento especial na finalização desta etapa.

Resumo

O constante avanço tecnológico tem impulsionado empresas a desenvolver software de forma mais eficiente e com qualidade para se diferenciar dos competidores no mercado. Requisitos não funcionais são atributos de qualidade que um sistema deve possuir ou determinam quão bem ele deve realizar uma determinada função. Esses requisitos são elicitados em paralelo aos funcionais e possuem uma influência direta na qualidade dos sistemas. São esses atributos associados às funcionalidades que tornarão o sistema atrativo ao usuário, seguro, com um bom desempenho, de fácil uso e de fácil aprendizado. Normalmente, requisitos não-funcionais são interdependentes, onde a satisfação de um requisito impacta na satisfação de outro, podendo originar conflitos que precisam ser compreendidos e gerenciados. Modelar requisitos não-funcionais é um desafio diante das características próprias a esses requisitos e da pequena importância a eles dada desde a fase de levantamento de requisitos até o desenvolvimento do sistema. A contribuição desta monografia é propor um *template* que permita uma especificação de requisitos com maior nível de detalhe. Para isso a proposta tomou como base duas técnicas de especificação de requisitos não funcionais: o *Quality atribute Scenarios Essentials* (QaSE) e *The Landing Zone* (LZ). A fim de dar suporte à especificação e uso do template proposto, um sistema chamado SAE-RNF (Sistema de Apoio à Especificação de Requisitos não Funcionais) foi projetado e implementado permitindo colocar em prática os conceitos explorados.

Palavras-chaves: Engenharia de software; Engenharia de requisitos; Requisitos não funcionais; Especificação de requisitos não funcionais.

Abstract

The constant technological advances have driven companies to develop software more efficiently and with quality to differentiate themselves from competitors in the market. Non-functional requirements are quality attributes that a system must possess or determine how well it must perform a certain function. These requirements are elicited in parallel with the functional and have a direct influence on the quality of the systems. These are attributes associated with the features that make the system attractive to users, secure, with a good performance, easy to use and easy to learn. Typically, non-functional requirements are interdependent, where the satisfaction of a requirement impacts on the satisfaction of another, which can cause conflicts that need to be understood and managed. Modelng non-functional requirements is a challenge on the characteristics these requirements have and little importance given to them from elicitation requirements phase to system development. The contribution of this paper is to propose a template that allows a requirements specification with higher level of detail. For this the proposal was based on two non-functional requirements specification techniques: Quality atribute Scenarios Essentials (QaSE) and The Landing Zone (LZ). In order to support the specification and use of the proposed template, a system called SAE-RNF (Sistema de Apoio à Especificação de Requisitos não Funconais) was designed and implemented allowing put into practice the explored concepts.

Key Words: Software engineering; Requirements engineering; Non-functional requirements; Non-functional requirements specification.

Sumário

Índice de Figuras

Índice de Tabelas

Tabela de Símbolos e Siglas

EssUP - Do inglês, Essential Unified Process

IEC – Do inglês, International Electrotechnical Commission

IEEE – Do inglês, Institute of Electrical and Electronics Engineers

LZ – Do Inglês, The Landing Zone

NRF – Do ingês, Non-Functional Requirements

PDF – Do inglês, Portable Document Format

QaSE - Do inglês, Quality atribute Scenarios Essentials

RNF – Requisito não funcional

SAE-RNF – Sistema de Apoio à Especificação de Requisitos não Funcionais

SIG - Do inglês, Softgoal Interdependency Graph

Capítulo 1
Introdução

Neste capítulo estão descritos a motivação para a pesquisa, os objetivos deste trabalho, assim como a descrição da estrutura do documento.

1.1 Motivação e caracterização do problema

A Engenharia de Requisitos é a fase da Engenharia de Software na qual os requisitos são descobertos, analisados e especificados a fim de fornecer uma visão clara e objetiva das necessidades reais do cliente [8]. Esta fase tem sido amplamente reconhecida como um fator crítico de sucesso de projetos de Software, sendo uma das etapas mais críticas, pois se os requisitos não forem devidamente elicitados, os requisitos podem levar o projeto ao fracasso [8]. No entanto, a experiência prática demonstra que os problemas podem facilmente ser identificados nesta fase; por exemplo, eles geralmente estão relacionados à falta de compreensão do negócio pelo analista de sistemas, ao mau entendimento da finalidade do sistema, e a falhas de comunicação entre analistas de negócio e analistas de sistemas [15].

Uma classificação que tem sido muito aceita na comunidade acadêmica divide os requisitos em três tipos: funcionais, não funcionais e organizacionais. Os requisitos funcionais constituem os serviços que o sistema deverá prover e a forma pela qual o mesmo responderá a cada entrada de dados. Os requisitos não funcionais relatam aspectos e restrições relacionados à qualidade (como é o caso de: confiabilidade, desempenho, portabilidade, segurança e usabilidade). Já os organizacionais são os requisitos relacionados às metas da empresa, políticas estratégicas adotadas [22].

De uma forma mais detalhada, pode se descrever os requisitos não-funcionais (RNFs) como atributos de qualidade que o sistema deve possuir, que são analisados de acordo com a necessidade do cliente levando em consideração os

aspectos técnicos e o ambiente no qual o sistema irá operar. Esses requisitos podem estar relacionados ao sistema como um todo, apenas a um requisito funcional ou a um grupo de requisitos funcionais. Os RNFs são cruciais durante o desenvolvimento do sistema, servindo como critério de decisões e aceitação; eles podem influenciar e até mesmo determinar o sucesso ou não sucesso de um projeto, porém tem sido dada pouca atenção a eles no desenvolvimento de software [17].

A questão de pesquisa tratada neste trabalho está relacionada ao fato de que os RNFs têm sido definidos com pouco entendimento e pequena riqueza de detalhes, e tratados como fatores menos críticos. É notório, em *templates* de documento de requisitos, que a seção dos requisitos não-funcionais é mais informal, funcionando como uma espécie de citações de atributos do sistema levantados durante a análise dos requisitos junto aos *stakeholders*. Frequentemente, eles são relatados de forma contraditória, ou seja, a sua elicitação é feita de forma inadequada e até mesmo incorreta, dificultando o entendimento e seu cumprimento durante o desenvolvimento do software, bem como a validação junto ao usuário e/ou cliente [8].

Existem várias propostas voltadas para a especificação e tratamento de RNF, como por exemplo The Landing Zone, Quality atribute Scenarios Essentials, i*, etc. Porém elas têm sido descritas de uma forma independente; apesar de terem aspectos em comum, observa-se que cada uma delas foca em um aspecto especifico.

Considerando-se o problema que vem se encontrando (da falta de detalhes na especifcicação de requisitos, levando-os a serem requisitos pouco claros, difíceis de serem verificados e conflituosos), e dada a sua importância no contexto do desenvolvimento de um sistema, esta monografia propões uma forma integrada de especificação de RNFs, tomando como base duas abordagens de especificação de RNFs existentes.

1.2 Objetivos

Este trabalho tem como objetivo o desenvolvimento de um *template* de especificação de requisitos não funcionais baseado em técnicas existentes. Outro objetivo é a criação de um sistema que utiliza este *template* para a modelagem de requisitos não-funcionais, a fim de proporcionar um melhor suporte à especificação e compreensão desses requisitos (explicitando suas interdependências, conflitos e relatando possíveis impactos que podem ocorrer no sistema com o não atendimento). A meta é introduzir no mercado/academia um *template*, apoiado por um sistema, que suporte uma especificação de requisitos não-funcionais mais completa e intuitiva.

1.3 Metodologia e estratégias de ação

Neste trabalho inicialmente foi feita uma revisão bibliográfica dos principais trabalhos sobre requisitos, classificação de requisitos, e técnicas de especificação e modelagem de RNFs. Em seguida foi proposto um *template*, e por fim projetado e implementado o sistema de apoio. Para ilustrar o sistema foram usados dois exemplos: Segurança em um Sistema de Gestão de Documentos e Disponibilidade no Sistema de Gestão de Pacientes Conveniados [21].

1.3.1 Revisão bibliográfica

Esta fase consistiu no estudo de conceitos sobre RNFs além de metodologias de especificação de requisitos, mais especificamente RNFs.

1.3.2 Proposta do Template para RNFs

Proposta integrada de template para especificação de RNFs. As técnicas que serviram como base para este projeto foram: *The Landing Zone* (por propor uma especificação mais precisa dos requisitos), e pela QaSE (por de uma forma complementar e leve permitr a especificação de RNFs). Essas técnicas foram escolhidas por: serem de fácil aprendizagem; usarem linguagem natural, o que é um diferencial se as compararmos com outras técnicas utilizadas no mercado/academia

como, por exemplo, a i*; e por se mostraram complementares, o que foi muito útil na elaboração do *template* proposto nesta monografia. Ambas são descritas brevemente a seguir.

- *The Landing Zone*: inclui a definição de uma tabela, onde é definida a região de sucesso de um produto ou projeto; as linhas da tabela contêm o subconjunto de RNFs que definem diretamente o sucesso ou a falha; Já as colunas da tabela contêm uma gama de níveis de desempenho. Geralmente uma *Landing Zone* abrange a faixa entre atingir o sucesso e evitar o fracasso. [12].

- QaSE - *Quality atribute Scenarios Essencial*: aborda o uso de cenários para especificação de RNFs, considerando sua elaboração, recomendações, interdependências, impactos e tratamento de conflitos. Detalha informações a respeito de atributos de qualidade que auxiliarão na implementação dessa prática em qualquer tipo de projeto ágil [21].

1.3.3 Projeto e Implementação do sistema

A construção do sistema para suporte à especificação de RNFs incluiu as seguintes etapas:

- Elicitação de requisitos
- Especificação de requisitos
- Análise e projeto de software
- Modelos conceitual, lógico e físico do banco de dados
- Implementação ou codificação
- Testes
- Validação do sistema com o usuário

1.4 Estrutura do documento

Este documento está dividido em 4 capítulos, cujos respectivos resumos estão apresentados a seguir:

Capítulo 1: Introdução: inclui os textos introdutórios deste trabalho: a motivação e a caracterização do problema, os objetivos e a descrição da organização do documento.

Capítulo 2: Background: fornece embasamento teórico do trabalho. São descritos os conceitos e classificação de Requisitos não Funcionais, também são descritas algumas técnicas de especificação de Requisitos não Funcionais: *Quality atribute Scenarios Essencial* (QaSE), *The Landing Zone* (LZ), entre outros.

Capítulo 3: Suporte à especificação de RNFs: apresenta as principais contribuições deste trabalho: a) a proposta de um *template* de especificação de requisitos não funcionais, ilustrado através de dois exemplos; e b) o Sistema de Apoio à Especificação de Requisitos não Funcionais chamado SAE-RNF.

Capítulo 4: Conclusão e Trabalhos Futuros: Neste capítulo são apresentadas as conclusões do trabalho, suas contribuições e possíveis trabalhos futuros.

Capítulo 2
Background

Este capítulo descreve os principais conceitos relacionados à Engenharia de Requisitos, tendo como foco os RNFs e estratégias para a sua definição.

2.1 Conceito de Requisitos

Requisitos podem ser descrições sobre o comportamento do sistema de software, informações sobre o domínio da aplicação, restrições de operação, ou especificação de suas propriedades ou atributos. Os requisitos podem também representar restrições no processo de desenvolvimento: por exemplo, no caso de sistemas críticos, as atividades de verificação e validação são mais rigorosas. Os requisitos devem ser definidos nas fases iniciais do desenvolvimento como uma especificação do que deve ser implementado [15].

Outra definição é dada por [17], em que requisitos de sistemas de software expressam as necessidades do usuário a fim de que seja possível resolver um determinado problema. Além disso, um requisito de software representa a capacidade que precisa ser obtida ou possuída pelo sistema de software, ou seu componente, para satisfazer contrato, padrão, especificação, ou outra documentação formalmente imposta.

Existem dois tipos de requisitos de software, os funcionais e os não funcionais. De acordo com [9], os requisitos funcionais são definições que expressam funções ou serviços que um sistema de software deve ou pode ser capaz de executar ou fornecer; funções ou serviços são entendidos como processos que utilizam entradas para produzir saídas. Requisitos não funcionais, por outro lado, são requisitos que declaram restrições. De acordo com [23] requisitos não funcionais se relacionam aos padrões de qualidade e definem se o sistema de software será eficiente e adequado para a tarefa que se propõe a fazer. Os requisitos não-funcionais, ao contrário dos requisitos funcionais, não determinam as funções a

serem realizadas pelos sistemas de software, mas os comportamentos e restrições que os sistemas de software devem satisfazer [8].

A complexidade de um sistema de software é determinada tanto por seus requisitos funcionais quanto pelos não funcionais (desempenho, confiabilidade, manutenibilidade, portabilidade, custos operacionais, entre outros). Os requisitos não funcionais, mais especificamente, desempenham um papel crítico durante o desenvolvimento de sistemas de software pois, se definidos de maneira incorreta, são de difícil correção e podem gerar grandes impactos em custos [4].

2.2 Requisitos Não Funcionais

Os requisitos não-funcionais de software em geral se relacionam com padrões de qualidade, tais como: confiabilidade, desempenho e robustez. Estes requisitos são importantes, pois definem se o software será eficiente para a tarefa que se propõe a fazer ou não. Segundo [14], os requisitos não-funcionais representam requisitos adicionais que definem as qualidades globais ou atributos a serem exibidos pelo sistema resultante. Já segundo [8], os requisitos não-funcionais, ao contrário dos funcionais, não expressam nenhuma função a serem realizados pelo software, e sim comportamentos e restrições que este software deve satisfazer.

Existem várias denominações para os requisitos não funcioanais, entre eles: atributos de qualidade, restrições, objetivos. Como o termo requisito não funcional tem se consolidado no meio acadêmico ele foi adotado neste trabalho.

Não há como desconsiderar a importância dos requisitos para o processo de desenvolvimento de software. Além disso, Engenheiros de Requisitos e a maioria das linguagens existentes incidem sobre os requisitos funcionais, negligenciado ou esquecendo os requisitos não-funcionais durante a concepção do software. Os atributos de qualidade de software (requisitos não-funcionais), por exemplo, são vistos como conseqüência destas decisões e não como algo que foi pensado. Para mudar este quadro e tratá-los coerentemente, é necessário não apenas métodos e representações, mas também como promover a sua reutilização.

Os requisitos não funcionais desempenham um papel crítico durante o desenvolvimento de sistemas. Erros devido à falta de elicitação ou a elicitação incorreta destes requisitos estão entre os mais caros e difíceis de corrigir, uma vez que um sistema tenha sido implementado [1]. De acordo com [8], para satisfazer um requisito não-funcional é possível que sejam criados conflitos, tanto com outros requisitos não-funcionais, como com requisitos funcionais. Conseqüentemente, o não tratamento dos requisitos não-funcionais durante o desenvolvimento do software pode ocasionar que esses conflitos só apareçam na implementação do software.

Os autores de [4] destacam que os requisitos não funcionais, durante o desenvolvimento de sistemas de software, servem como base para critérios de seleção entre as alternativas de projeto e implementação. Além disso, descrevem os requisitos não funcionais como: a) subjetivos, pois podem ser interpretados e avaliados de maneira diversa; b) relativos, pois sua interpretação e importância podem variar dependendo do sistema de software considerado; e c) interativos, pois o atendimento de um RNF pode ferir ou beneficiar o atendimento de outros RNF. Portanto, soluções especificas, que não consideram o sistema de software como um todo, podem não ser as mais adequadas. Desta forma, o contexto de aplicação de um RNF precisa ser definido para que um RNF possa ser considerado atendido ou não-atendido. Exemplos de contextos aplicáveis para RNFs são: “o sistema de software deve ter acurácia especialmente quando existirem tentativas de fraudes” e “o sistema de software deve ser rápido especialmente em períodos de pico”.

Modelar requisitos não-funcionais é um desafio diante de restrições de relacionamento entre requisitos, como interdependências e conflitos, e da forma parcial que eles são abordados desde a fase de levantamento de requisitos até o desenvolvimento do sistema. A um RNF podem estar associadas funções e/ou restrições que se não forem atendidas poderão causar um impacto global no sistema. O impacto causado por um requisito não atendido poderá afetar globalmente um sistema, sendo difícil descobrir a origem, como e quando o erro ocorreu.

2.2.1 Classificação para RNFs

Acessibilidade, segurança, confidencialidade, desempenho, portabilidade, consistência, manutenabilidade, eficiência, robustez, são alguns exemplos de requisitos não funcionais. Em [4] os autores também consideram a usabilidade como um requisito não-funcional, apesar de não ser catalogada no seu *framework*.

Segundo [4], não existe uma definição formal ou lista completa de requisitos não-funcionais. Também não existe algum esquema de classificação único e universal que acomode todas as necessidades de aplicações de diferentes domínios. Por conta disso para cada aplicação e domínio diferentes classificações são feitas.

Existem diversas classificações para os tipos de RNFs na literatura. Para ilustrar, foram consideradas as classificações da norma IEEE *Standard* 830 (IEEE, 1998), do padrão internacional ISO/IEC 9126 (ISO/IEC, 2001), de [4] e [15].

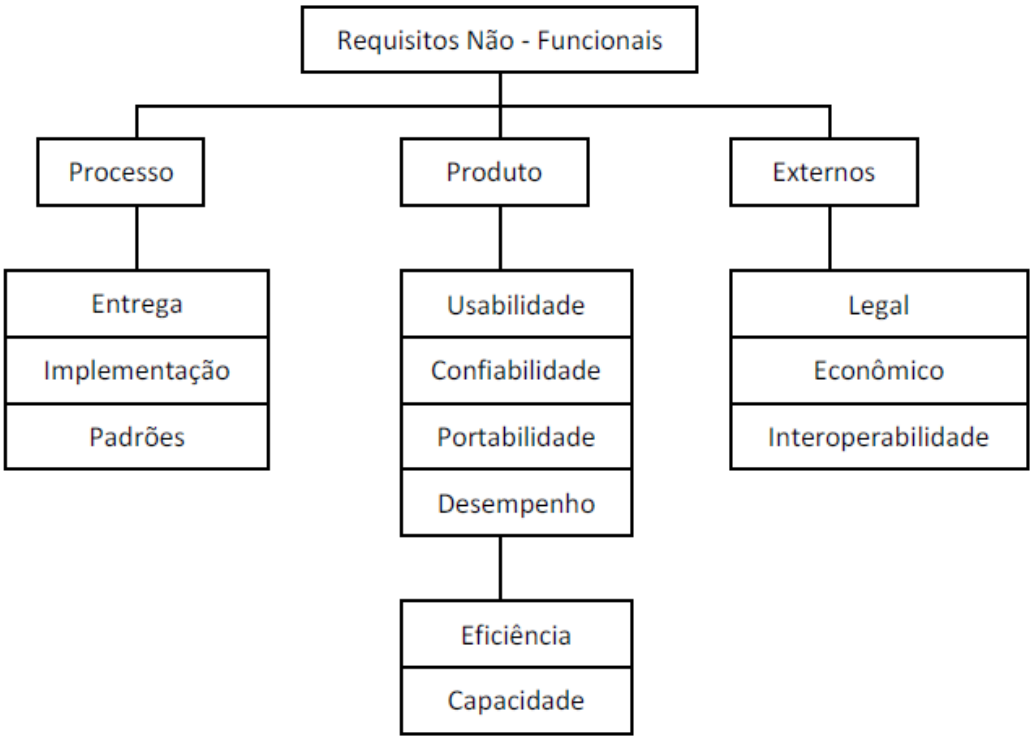
A norma IEEE *Standard* 830 (IEEE, 1998) apresenta uma lista de exemplos de tipos de RNFs: desempenho, interface, recursos, verificação, aceitação, documentação, *security*, *safety*5, portabilidade, qualidade, confiabilidade e manutenibilidade.

O padrão internacional ISO/IEC 9126 (ISO/IEC, 2001) apresenta seis características que definem a qualidade de software: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Estas características de qualidade são subdivididas em sub-características para melhor definir os RNFs dos sistemas de software.

Em [4] uma lista extensa, com mais de uma centena de tipos de RNFs, é apresentada. Dentre eles, os autores detalham o tratamento relativo aos requisitos de acurácia, *security* e desempenho. Já [15] apresenta outra forma de classificação de RNFs, agrupando-os em aqueles relacionados com produto, processo e ambiente externo. Os requisitos de produto são aqueles que especificam as características desejadas que um sistema de software ou subsistema devem possuir e geralmente são os mais usados. Os requisitos de processo são restrições ao processo de desenvolvimento do sistema de software e correspondem à definição de padrões,

uso de ferramentas para gerenciamento, equipe treinada segundo um modelo de qualidade. Os requisitos de ambiente externo influem nos requisitos de produto e de processo e podem ser classificados em: legal, econômico e de interoperabilidade. A Figura 1 apresenta esta classificação.

Figura 1. Classificação dos RNFs adaptada de Sommerville e Kotonya (1998) [4].



2.2.2 Dependencias entre Requisitos

Compreender o relacionamento entre requisitos é importante para o entendimento correto do objetivo e funcionamento do sistema. De uma forma geral, os requisitos de um sistema não podem ser tratados isoladamente, como requisitos independentes, pois eles se inter-relacionam. Focando nos requisitos não-funcionais, identificar essas interdependências no momento de sua modelagem poderá prever falhas durante a implantação do sistema, pois se não forem devidamente tratadas, podem acarretar em retrabalho e insucesso do sistema. Controlar essas dependências pode ser crucial para o sucesso do projeto, mas identificá-las não é simples.

As dependências entre os requisitos poderão favorecer a atuação de um determinado requisito ou poderão afetar o atendimento a outro. Por exemplo, o requisito não-funcional de segurança pode ter interdependências com os requisitos de desempenho, confiabilidade e usabilidade. A interdependência entre segurança e desempenho, segurança e usabilidade são conflitantes, uma vez que aumentar a segurança no acesso ao sistema poderá influenciar negativamente o seu desempenho, tornando a execução de serviços mais lenta. O mesmo acontece com a usabilidade, quanto maior a segurança em um sistema, menor a usabilidade fornecida ao usuário, podendo tornar o uso e aprendizado difíceis. Com relação à segurança e confiabilidade é uma dependência que favorece aos dois RNFs. Quanto mais o acesso aos dados e ao próprio sistema é controlado, tornando-o seguro, maior a confiabilidade e exatidão nos resultados obtidos.

2.3 Especificação de RNFs

Nesta seção são apresentadas algumas técnicas de especificação de requisitos não funcionais que serviram como base teórica para a proposta deste trabalho. São elas QaSE (*Quality attributes Scenarios Essentials*), The Landing Zone (LZ), KAOS e NRF *Framework*.

2.3.1 Especificação através de QaSE

Práticas de melhorias e de modelagem de requisitos foram propostas por [13] no EssUP (*Essential Unified Process)*. Essas práticas foram inseridas em um processo que utiliza um método tradicional, tornando-o ágil; a finalidade foi aumentar a produtividade da equipe e fornecer serviços e produtos de qualidade aos *stakeholder*s. Baseada no EssUP, foi então elaborada uma prática chamada *Quality attributes Scenarios Essentials* (QaSE) que propõe a adoção de *quality scenarios*, para a modelagem de requisitos não-funcionais, em projetos ágeis dentro do fluxo *Scrum.* Um *Scenario* (cenário) é uma técnica de modelagem muito usada na descrição de situações onde o sistema pode estar inserido, permitindo analisar quais os fatores o estimulam e de que forma ele irá responder [2]; além disso também é usado para descrever fluxos de negócio [20].

A proposta de modelagem de requisitos não-funcionais utilizando *Quality Scenarios* aplicada a projetos ágeis tem o objetivo de possibilitar uma melhor compreensão desses requisitos. *Quality Scenarios* modelam os requisitos não-funcionais do sistema, contendo recomendações de uso, descrição do cenário abordado, interdependências, explicação de conflitos quando existirem, impactos causados no sistema quando um NFR não for devidamente atendido. A proposta fornece um tratamento de requisitos não-funcionais de forma leve, sem a grande quantidade de artefatos produzidos, com ênfase em atender as necessidades do *stakeholders,* sendo facilmente aplicável a projetos ágeis [21].

A literatura disponível sobre métodos ágeis não foca a modelagem de RNF de forma explícita. Os RNFs não são abordados com a mesma prioridade que os funcionais (durante a fase de elicitação) e a preocupação de atendê-los, em sua maioria, apenas surge ao final da implementação do sistema. Porém, tais requisitos têm uma grande importância e influência direta no atendimento às necessidades solicitadas e na satisfação do cliente. Assim, a fim de evitar que os requisitos não-funcionais sejam mal interpretados no momento de sua elicitação ou que seja dedicada a eles pouca atenção em seu detalhamento, acarretando em insatisfação dos *stakeholders* [21], foi proposto o QaSE.

2.3.1.1 Quality attributes Scenarios Essentials

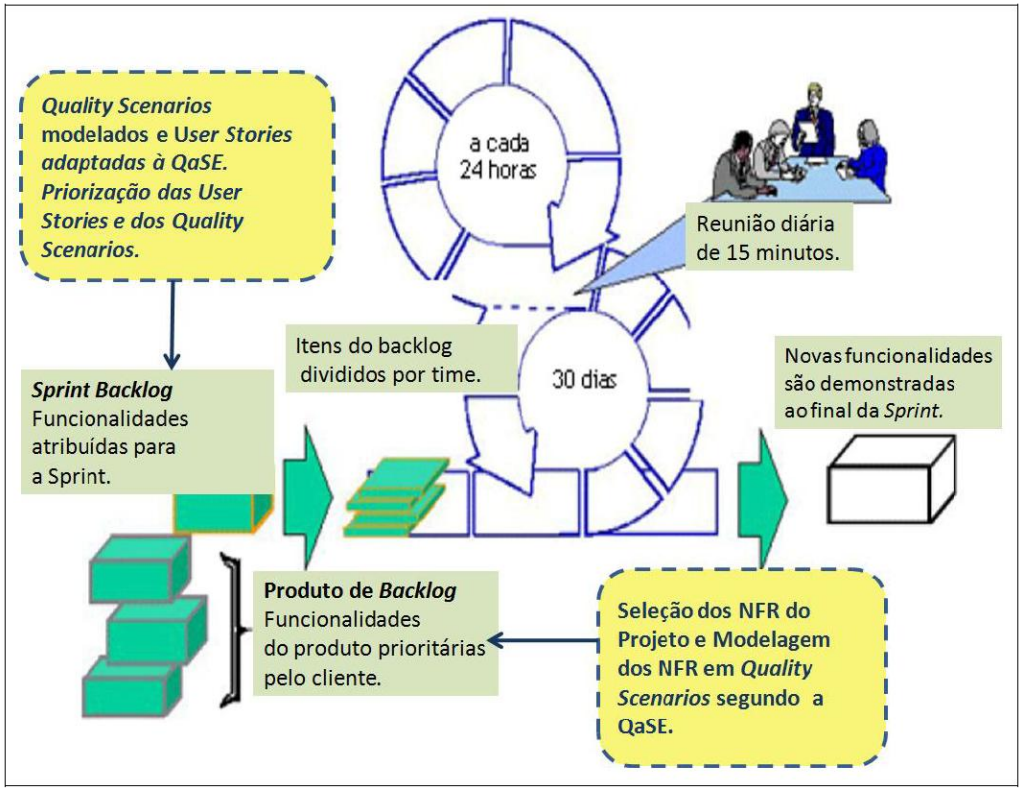
A QaSE descreve a prática de *quality scenarios*, apresentando uma série de informações a respeito de atributos de qualidade que auxiliarão na implantação dessa prática em qualquer tipo de projeto ágil [21].

A *Quality attribute Scenarios Essentials* pode ser integrada a projetos de software que utilizam o método *Scrum,* uma vez que a sua execução acontece dentro do próprio fluxo do processo ágil, durante as fases de definição do *Product Backlog* e do *Sprint Backlog*. Os requisitos não-funcionais que farão parte do escopo do projeto são modelados em *quality scenarios.* Essa prática é adaptável à realidade do projeto a fim de atender às necessidades mais específicas e particulares ao projeto.

No *Sprint Backlog,* os requisitos funcionais terão suas respectivas *user stories,* modeladas na fase de levantamento junto aos stakeholders, adaptadas ao modelo da QaSE. De acordo com a importância para o negócio e para os *stakeholders,* as *user stories* serão priorizadas*.* Após a priorização, os requisitos serão decompostos em tarefas que deverão ser realizadas ao longo da *Sprint* [21].

Enquanto os requisitos funcionais são decompostos em tarefas no *Sprint Backlog*, os não-funcionais são modelados em *quality scenarios* no *Product Backlog.* No *Sprint Backlog,* os requisitos não-funcionais, uma vez modelados, serão priorizados e validados com o time antes de iniciar a implementação do sistema. Na Figura 2 pode-se entender melhor o momento em que são realizadas as atividades relatadas no fluxo do *Scrum* com a prática QaSE representada pelo retângulo amarelo:

Figura 2. Prática QaSE no fluxo do Scrum [21].



No *Product Backlog* serão definidos os atributos de qualidade que estão associados às funções e serviços do sistema. Durante a fase de elicitação de requisitos, os requisitos não-funcionais foram levantados em paralelo aos funcionais.

Esses requisitos devem ser bem definidos, compreendidos e detalhados no começo do projeto, uma vez que representam o funcionamento do sistema e a forma como ele deve reagir a estímulos no contexto onde será incluído.

Os requisitos funcionais e não-funcionais levantados são apresentados ao time para que seja realizada a priorização das funcionalidades e dos atributos de qualidade de acordo com a importância para os *stakeholders* e para o negócio, bem como devem ser avaliadas as interdependências entre os requisitos e os possíveis conflitos que possam vir a ocorrer. As interdependências e os conflitos serão relatados no *quality scenario*, fornecendo informações que possibilitarão a tomada de decisão do time com relação à melhor solução aplicada para que o RNF seja atendido [21].

2.3.1.2 Modelagem de Requisitos Funcionais *Essentials*

Antes da definição da listagem de funcionalidades que irão compor o *Product Backlog,* durante os eventuais encontros com os *stakeholders,* as *user stories* vão sendo elaboradas. Os requisitos funcionais são modelados durante a fase de elicitação de requisitos, onde são descritos de forma direta e objetiva, por meio de uma linguagem natural.

As *user stories* devem abordar a visão dos *stakeholders* em relação ao sistema e as necessidades que deverão ser atendidas, sendo utilizadas como guias de funcionamento do sistema. Deverão ser narrativas mais simples e objetivas se comparadas com os casos de uso do RUP, não se limitando a descrever o processo de interação do usuário com o sistema. Expressam as necessidades dos *stakeholders,* relatando cada item de *backlog* desejável, as regras de negócio que devem ser persistidas em cada funcionalidade, campos existentes, tipo de dados de entrada e de saída [21].

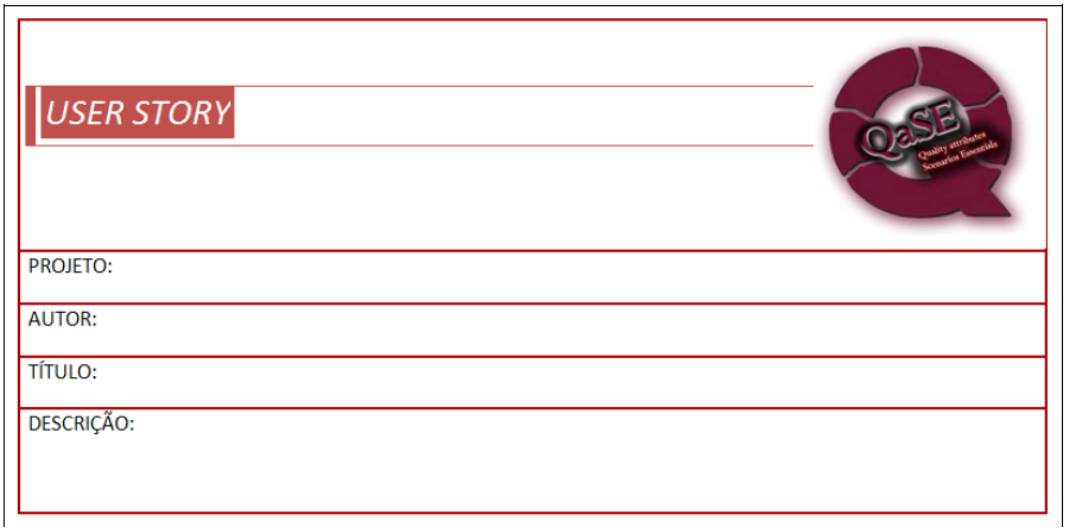
Uma vez definidas e modeladas, as *user stories* serão validadas pelo time, o que inclui a participação do analista de requisitos. A partir dessa primeira análise, as *user stories* serão priorizadas, no momento de definição do *Product Backlog,* respeitando a priorização dada aos requisitos funcionais, estabelecida pelo time.

Ao descrever as informações, preocupando-se com o correto entendimento de cada requisito não-funcional, de suas interdependências e conflitos poderá tornar os requisitos funcionais mais legíveis, revisáveis e compreensíveis.

Essa técnica não segue nenhum padrão formal para a descrição das *user stories. As* informações são expostas em forma de texto pelos *stakeholders, transmitindo* para o time a visão sobre o sistema, o que deseja que o sistema realize e de que forma o sistema poderá interagir no ambiente em que será inserido [21].

As *user stories* nessa proposta modelarão os requisitos funcionais, utilizando o *template,* exibido na Figura 3:

Figura 3. *Template* de *user story* [21].



Utilizando o *template* da *user story*, a visão dos *stakeholders* vai ser passada para a equipe do projeto por meio do analista de requisitos ao informar:

- **Projeto:** Nesse campo os analistas informarão o projeto ao qual a *user story* se refere. Esse campo ajudará a organização e seleção das *user story* no projeto em desenvolvimento.

- **Autor:** Refere-se à pessoa responsável por descrever a *user story*. Com essa identificação, o time saberá a quem recorrer no momento em que ocorrer algum problema ou surgir alguma dúvida durante a implementação do sistema.

- **Título**: Nesse campo, os analistas informarão como a *user story* será chamada e identificada. Esse título deve ser sugestivo referindo-se ao conteúdo da *user story*.

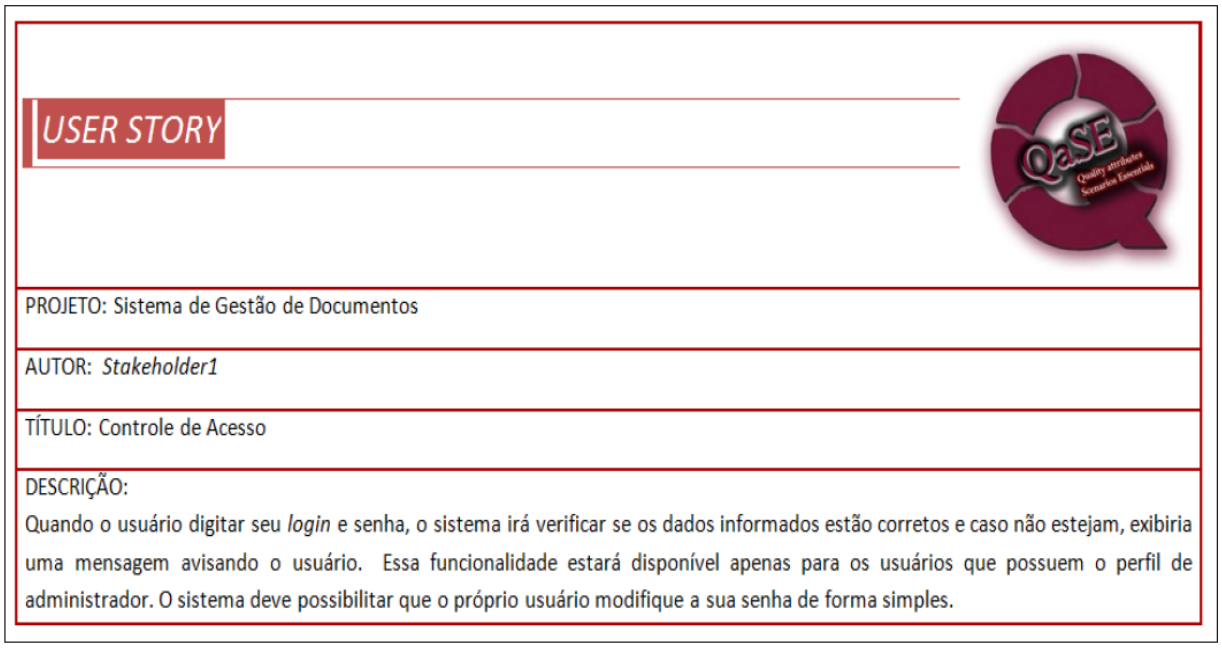
- **Descrição:** Nesse local, todas as informações sobre o sistema deverão ser descritas: o entendimento dos *stakeholders* sobre o sistema, as regras de negócio, validação de campos, o que o sistema deverá realizar, o que será permitido ou não pelo sistema, o que o usuário poderá realizar em seu acesso. Quanto maior o número de informações aqui relatadas mais rica será a *user story*, maior o entendimento do time.

Os engenheiros de software têm se deparado com um problema: *user story* com poucas informações relevantes. Elas têm sido escritas com uma pequena quantidade de informações sobre o sistema. Informações como regras de negócio, regras de validação de campos e de acesso, têm sido omitidas. Por se tratar de uma redação informal e objetiva, as *user stories*, em sua maioria, têm sido escritas de forma rudimentar, sem a preocupação de repassar a correta visão do sistema e todas as ações que ele deverá realizar.

Diante desse contexto, esse *template* foi sugerido para auxiliar na organização das idéias, identificação das *user stories* e para a formalização dessa técnica em um documento que poderá ser arquivado junto a outros documentos do projeto. Esse *template* não vem para impor a formalização do uso de artefatos com padrões e regras, mas sim como uma forma de dar uma nova roupagem a essa técnica tão utilizada em projetos ágeis, trazendo melhorias para a sua elaboração e seu entendimento [21].

A Figuras 4 apresenta exemplo de *user story* com o *template* do QaSE.

Figura 4. Exemplo de user story: Controle de Acesso [21].



A modelagem dos requisitos funcionais seguindo o formato do *template* propõe uma melhor compreensão do funcionamento do sistema sob a ótica dos *stakeholders* e a interação com o time e com o usuário para aprovação final.

2.3.1.3 Modelagem de NFR Essentials

Essa prática refere-se à utilização de modelos para facilitar a compreensão de requisitos não-funcionais, produzindo uma documentação útil para o desenvolvimento de software, conduzindo as atividades de desenvolvimento para:

- Selecionar e priorizar os requisitos não-funcionais;
- Informar os requisitos e o comportamento do sistema ao realizar ações e serviços;
- Enxergar as interdependências e conflitos entre os requisitos e entender como eles estão relacionados um ao outro;
- Empregar o modelo correto para a modelagem de NFR, a fim de atender às necessidades;

- Ser ágil em sua abordagem para modelagem e documentação;
- Focar nos NFR *Essentials* evitando a interrupção da modelagem em *Quality Scenarios*, a produção de documentos desnecessários e que não sejam ágeis.

Após priorizar as *user stories* de acordo com os requisitos funcionais definidos no *Product Backlog*, o próximo passo é modelar os requisitos não-funcionais, aplicando a prática de *quality scenarios*. O primeiro passo é identificar os requisitos não-funcionais que farão parte do *Product Backlog,* priorizando-os de acordo com a importância para o negócio. Diante de todos os requisitos levantados, os requisitos serão selecionados e priorizados. Uma vez definidos, os requisitos não-funcionais são modelados em *quality scenarios* no momento de definição do *Product Backlog.* Os *quality scenarios* deverão ser modelados de forma clara e objetiva por um analista de requisitos que acompanhará de forma ativa a definição do escopo, para a compreensão do time durante o desenvolvimento do sistema [21].

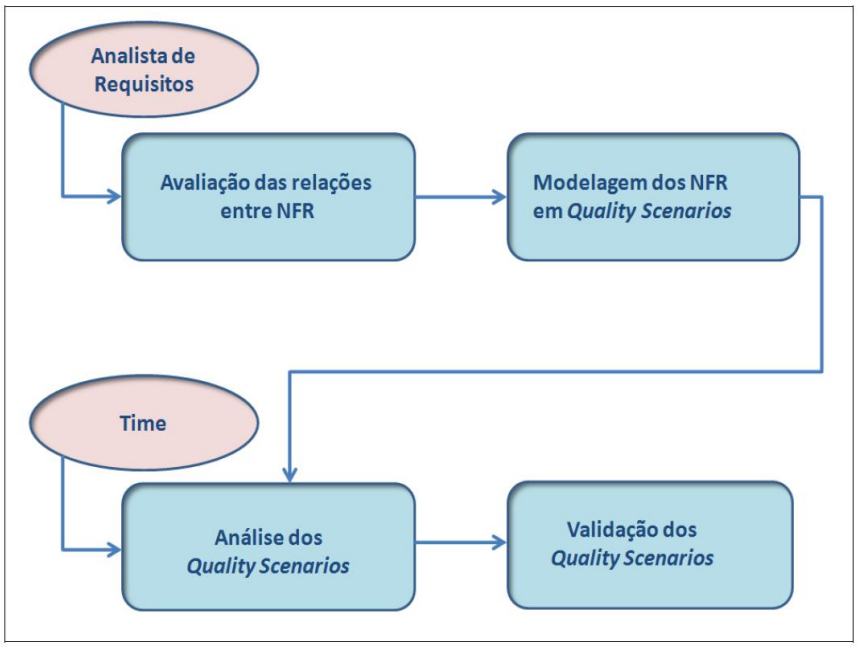
Quality Scenarios descrevem de forma narrativa as necessidades solicitadas referentes ao requisito não-funcional: as situações em que os requisitos serão abordados no sistema, os aspectos comportamentais do requisito, as ações do sistema para atender ao requisito. Essa descrição segue o formato narrativo das *user stories*, relatando as interdependências entre os RNF, os conflitos que podem ocorrer com o atendimento de determinados RNF e os impactos que podem afetar o sistema caso as interdependências e conflitos não sejam tratados de forma correta. Em suma, os *quality scenarios* são um conjunto de informações a respeito dos requisitos não-funcionais que fornecem uma compreensão dos atributos a serem atendidos [21].

Ao elicitar os requisitos não-funcionais de um sistema, deve ser dedicada uma atenção às dependências existentes entre eles. Na descrição dos *quality scenarios*, essas interdependências são avaliadas de acordo com os requisitos que fazem parte do escopo do *Product Backlog.* Serão descritas em uma linguagem natural, sem preocupação de detalhes técnicos específicos, deixando-os sob responsabilidade do time durante a fase de validação que ocorrerá no *Sprint Backlog.* Os analistas de requisitos avaliarão a relação entre os requisitos não-funcionais, compreendendo a

interdependências entre eles bem como os conflitos, e descreverão nos *quality scenarios,* informando ao time. O time verificará se existem mais dependências, analisarão como poderão ser tratadas de modo que não haja conflitos entre eles e não causem impacto no sistema, e por fim, o *quality scenario* será validado [21].

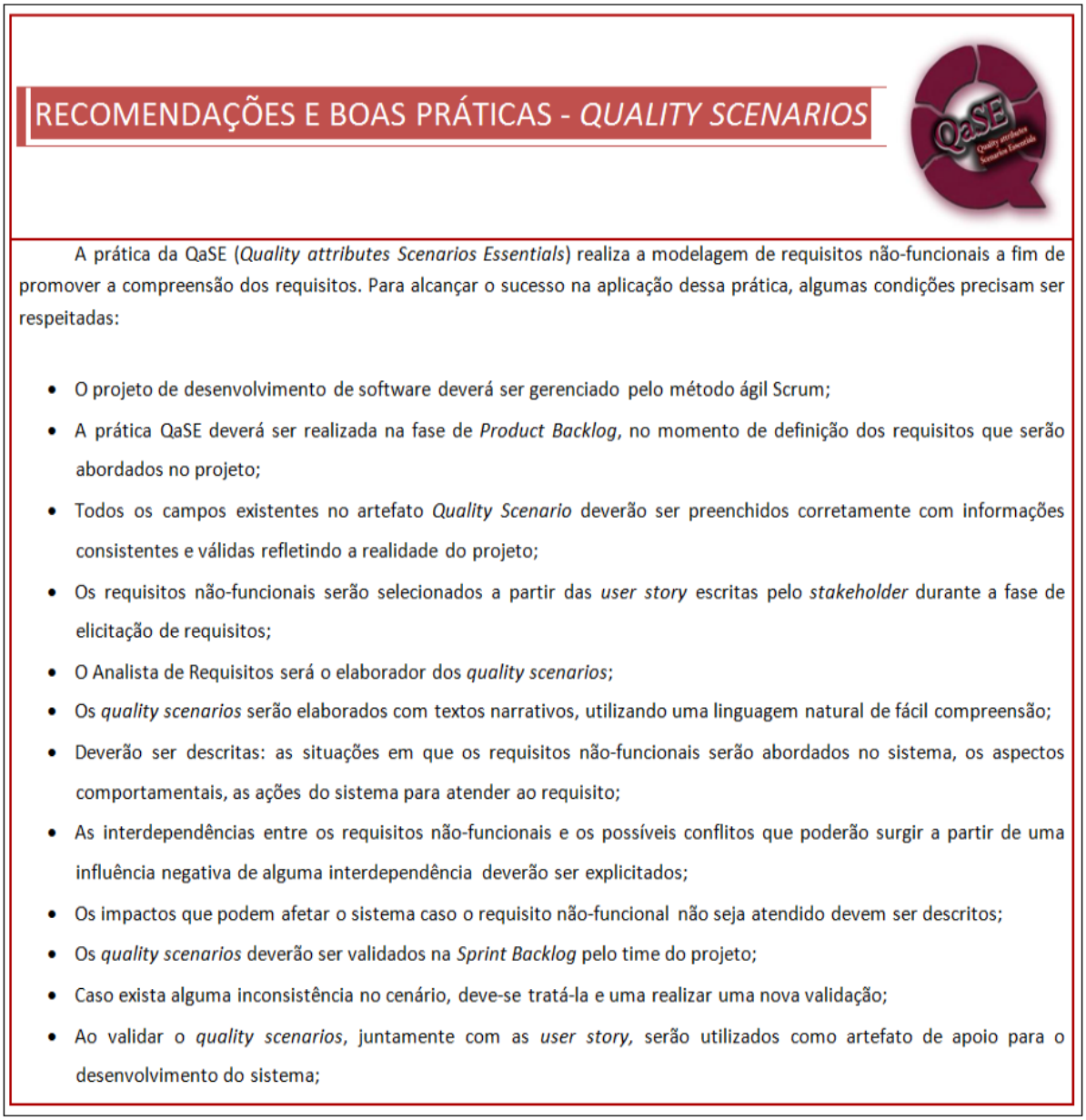
Essa seqüência de passos poderá ser melhor compreendida na Figura 5.

Figura 5. Sequência de atividades da descrição dos quality scenarios [21].



Na Figura 6 estão descritas as recomendações para a aplicação da técnica, tudo o que é necessário para atender à modelagem de RNF.

Figura 6. Recomendações de Uso de Cenários de Qualidade [21].



RECOMENDAÇÕES E BOAS PRÁTICAS - *QUALITY SCENARIOS*

A prática da QaSE (*Quality attributes Scenarios Essentials*) realiza a modelagem de requisitos não-funcionais a fim de promover a compreensão dos requisitos. Para alcançar o sucesso na aplicação dessa prática, algumas condições precisam ser respeitadas:

- O projeto de desenvolvimento de software deverá ser gerenciado pelo método ágil Scrum;
- A prática QaSE deverá ser realizada na fase de *Product Backlog*, no momento de definição dos requisitos que serão abordados no projeto;
- Todos os campos existentes no artefato *Quality Scenario* deverão ser preenchidos corretamente com informações consistentes e válidas refletindo a realidade do projeto;
- Os requisitos não-funcionais serão selecionados a partir das *user story* escritas pelo *stakeholder* durante a fase de elicitação de requisitos;
- O Analista de Requisitos será o elaborador dos *quality scenarios*;
- Os *quality scenarios* serão elaborados com textos narrativos, utilizando uma linguagem natural de fácil compreensão;
- Deverão ser descritas: as situações em que os requisitos não-funcionais serão abordados no sistema, os aspectos comportamentais, as ações do sistema para atender ao requisito;
- As interdependências entre os requisitos não-funcionais e os possíveis conflitos que poderão surgir a partir de uma influência negativa de alguma interdependência deverão ser explicitados;
- Os impactos que podem afetar o sistema caso o requisito não-funcional não seja atendido devem ser descritos;
- Os *quality scenarios* deverão ser validados na *Sprint Backlog* pelo time do projeto;
- Caso exista alguma inconsistência no cenário, deve-se tratá-la e uma realizar uma nova validação;
- Ao validar o *quality scenarios*, juntamente com as *user story*, serão utilizados como artefato de apoio para o desenvolvimento do sistema;

Para a realização da modelagem de NFR utilizando os *Quality Scenarios,* um artefato leve foi proposto. Nesse artefato serão informados o projeto, autor, título, descrição do *scenario*, interdependências entre NFR, conflitos existentes e os possíveis impactos que poderão afetar o sistema caso o NFR não seja atendido. A Figura 7 exibe o *template* desse artefato:

Figura 7. *Template* de *Quality Scenarios* [21].

No *template* do *quality scenario*, o analista de requisitos informará:

- **Projeto:** nesse campo o analista informará qual o projeto ao qual o *scenario* pertence. Esse campo ajudará a organização e seleção das *user stories* no projeto em desenvolvimento.

- **Autor:** refere-se ao analista de requisitos responsável pela modelagem do *quality scenario.* Com essa identificação, o time saberá a quem recorrer no momento em que ocorrer algum problema ou surgir alguma dúvida durante a implementação do sistema. O analista atuará no suporte ao time e o acompanhará durante todo o desenvolvimento.

- **Título**: nesse campo será informado o nome do *quality scenario*. O título deve ser sugestivo referindo-se ao requisito não-funcional que será abordado no *scenario*.

- **Descrição:** nesse local, todas as informações relacionadas às necessidades solicitadas referentes ao requisito não-funcional: as situações nas quais os

NFR serão abordados no sistema, os aspectos comportamentais, as ações do sistema para atender ao requisito, as condições de uso, deverão ser descritas. Nesse campo, uma narrativa explicando o negócio ao usuário e ao time deve ser escrita, utilizando uma linguagem natural, simples e objetiva. Quanto maior a quantidade e qualidade das informações, mais compreensível se tornará o requisito não-funcional.

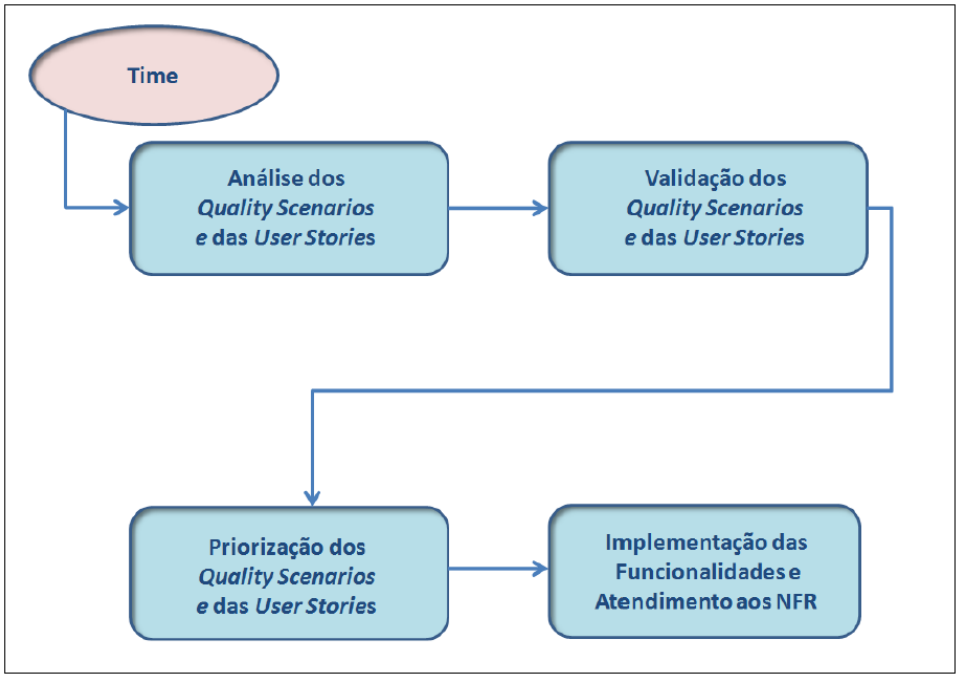
- **Interdependências entre NFR***:* o analista informará quais os NFR que possuem alguma dependência em relação ao requisito modelado no *scenario* e com quais requisitos o mesmo possui dependência. Descreverá como são as dependências entre os requisitos não-funcionais.

- **Conflitos entre NFR**: quando há dependências entre requisitos que de alguma forma interferem negativamente no requisito não-funcional modelado, gera algum tipo de conflito no sistema que precisa ser descrito. Os conflitos são identificados e analisados, para que seja definido pelo time como serão tratados.

- **Impactos do não atendimento ao NFR:** o requisito não-funcional modelado no documento de *quality scenarios* precisa ser atendido, caso não seja, alguns impactos poderão afetar o sistema. Esses impactos precisam ser informados ao time por meio desse campo, a fim de confirmar a importância daquele requisito não-funcional para o sistema.

Após a descrição dos *quality scenarios*, o analista de requisitos os repassará juntamente com as *user stories* descritas pelos *stakeholders* aos engenheiros que validarão tudo o que foi modelado. A validação ocorrerá no *Sprint Backlog,* pois é nesse momento em que o time terá contato com as *user stories* e com os *quality scenarios,* a fim de juntamente com o *Product Owner* priorizarem cada um de acordo com o escopo definido para a *Sprint.* Durante a priorização, melhorias podem ser sugeridas para um melhor entendimento dos NFR, assim como pode ser sinalizada a possibilidade de desenvolver ou não as funcionalidades e atender determinados atributos na *Sprint.* O time avaliará o melhor tratamento para as interdependências e para minimizar o impacto dos conflitos entre os NFR. Essa seqüência de passos poderá ser melhor compreendida na Figura 8:

Figura 8. Atividades da análise/validação das User Stories e Quality Scenarios [21].



A validação dos *quality scenarios* é extremamente necessária para que não haja uma interpretação errada dos requisitos não-funcionais. Todo o time precisa ter o mesmo entendimento dos requisitos para que o sistema seja desenvolvido de modo que o RNF seja atendido. Com o escopo da *Sprint* definido e validados os *quality scenarios,* o time terá subsídios para iniciar o desenvolvimento das funcionalidades e implementar o sistema de modo que atenda ao atributo modelado [21].

Um *template* foi sugerido para auxiliar no detalhamento dos requisitos não-funcionais, organização das idéias e compreensão. Vejamos um exemplo de *quality scenarios* utilizando o *template* proposto pelo QaSE nas Figuras 9:

Ao final da *Sprint,* acontecerá a *Retrospective Sprint,* onde o time se reunirá e repassará todas as dificuldades encontradas durante o desenvolvimento das funcionalidades e o atendimento aos requisitos não-funcionais. Esse momento é crucial para o fechamento da *Sprint* e continuação do projeto, pois é o momento onde serão levantadas as dificuldades, lições aprendidas, melhorias que poderão ser realizadas para favorecer o sucesso do projeto e a conclusão da *Sprint.*

Figura 9. Exemplo de *Quality Scenarios*: Usabilidade do Sistema de Contratos [21].

QUALITY SCENARIO

PROJETO: Sistema de Gestão de Contratos

AUTOR: Analista de Requisitos 1

TÍTULO: Usabilidade do Sistema de Contratos

DESCRIÇÃO:

Quando o usuário fizer o seu *login* pela primeira vez no sistema, ele deverá achar o sistema fácil de utilizar e aprender rapidamente. A interface do sistema deverá ser amigável, sendo de fácil uso e de fácil aprendizado. As funcionalidades do sistema devem ser categorizadas por *menu* e devem estar classificadas de acordo com o fluxo de negócio. De acordo com o perfil, o sistema exibirá as opções de *menu* permitidas para o usuário. Para cadastrar um contrato, o usuário deverá selecionar no *menu* essa opção e o sistema exibirá a tela de cadastro. Todos os itens do cadastro deverão estar em uma única tela. O sistema deve ser intuitivo, livre de erros operacionais, com pequena quantidade de comandos e deve realizar as ações em um pequeno intervalo de tempo, de forma eficiente.

INTERDEPENDÊNCIAS ENTRE NFR *Essentials*:

1. **Usabilidade x Manutenabilidade:** O sistema tem capacidade de ser modificado. Correções, melhorias ou adaptações do software a mudanças no ambiente, nos requisitos e nas especificações funcionais. Essas possíveis modificações e adaptações poderão ser realizadas para tornar o sistema mais usável, de fácil entendimento e intuitivo. São diretamente proporcionais.

CONFLITOS ENTRE NFR *Essentials*:

1. **Usabilidade x Desempenho:** Ao garantir a eficiência e eficácia do sistema, tornando usável, poderá comprometer o desempenho do sistema. Pode ser que a realização de alguma rotina ou serviço favoreça a usabilidade faça o desempenho do sistema cair. É preciso analisar a melhor alternativa para não comprometer o desempenho e atender a usabilidade solicitada pelo stakeholder.
   Ao definir a interface com o usuário deve-se ter a preocupação de
2. **Usabilidade x Segurança:** O usuário espera realizar suas tarefas rapidamente, sem dúvidas e sem erros causados pela dificuldade de usar ou de aprender. Ao adotar medidas de segurança para prover autenticação, autorização e confidencialidade, a usabilidade é afetada negativamente, uma vez que mais passos serão necessários para se realizar as mesmas ações. É preciso avaliar medidas de seguranças mais adequadas para o sistema a ser desenvolvido.
   Para evitar a diminuição da usabilidade

IMPACTOS DO NÃO ATENDIMENTO AOS NFR *Essentials*:

1. A navegabilidade do sistema será comprometida, não sendo intuitiva, dificultando o aprendizado.
2. Se a usabilidade não for atendida, acarretará na insatisfação do *stakeholder*, podendo afetar o sucesso do projeto.

As tarefas não concluídas durante a *Sprint* serão reavaliadas e poderão ser incluídas no próximo *Sprint Backlog.* Os atributos de qualidade serão acompanhados

e mantidos ao longo do projeto, podendo ser novamente abordados na próxima *Sprint.*

Serão verificadas as *user stories* a fim de analisar se todas foram concluídas e os *quality scenarios* são analisados de modo que seja verificado se o sistema está sendo desenvolvido para atender ao final do projeto os requisitos definidos. A retrospectiva é de fundamental importância, pois com a visão geral de tudo o que foi realizado na *Sprint,* é possível realizar ações como a re-priorização de *user stories* e de *quality scenarios,* aquisição de mais integrantes para o time, visando atender às solicitações dos *stakeholders,* o escopo e cronograma do projeto*.*

Os requisitos não-funcionais modelados em *quality scenarios* poderão ser abordados repetitivamente a cada *Sprint,* considerando as novas funcionalidades que deverão ser desenvolvidas. Nesse caso, os *quality scenarios* poderão ser novamente avaliados e validados na *Sprint Backlog,* podendo ter sua complexidade aumentada. Se compararmos essa evolução de *quality scenarios* aplicada em projeto ágil com os casos de uso de um projeto tradicional, por exemplo, o RUP, pode-se verificar que ao longo do projeto, casos de uso que inicialmente continham uma especificação simples de uma determinada funcionalidade, poderá ser avaliado e modificado para atender alguma necessidade ou evitar algum problema para o sistema, tornando-o complexo. O mesmo pode ocorrer com um *quality scenario,* onde a sua complexidade poderá ser ampliada a fim de garantir o seu atendimento ao longo do projeto, evitando conflitos e possíveis impactos no sistema.

A *Quality attributes Scenarios Essentials* utiliza a sistemática de [13] que se baseia em uma série de práticas ágeis que podem ser integradas ao processo de desenvolvimento de software de uma organização aumentando a produtividade do time e garantindo o atendimento às solicitações dos *stakeholders,* acarretando na satisfação e sucesso do projeto.

Diante da sistemática proposta no EssUP, a QaSE possui práticas centradas no desenvolvimento de software, preocupando-se em: fornecer informações com qualidade para melhorar a compreensão dos requisitos não-funcionais, propiciar a interação entre os integrantes do time e, entre time e *stakeholders.* Dessa forma, um projeto onde existam informações de qualidade, entendimento de requisitos,

interação entre integrantes, pequena quantidade de documentação, iteratividade ao longo desenvolvimento, aplicação de práticas ágeis, contribui para o alcance da alta satisfação dos *stakeholders* e do sucesso do projeto [21].

2.3.2 Especificação através de *The Landing Zone*

The Landing Zone (LZ) foi idealizada pela *Intel*, *Oregon* no ano de 2002 [10].

2.3.2.1 Conceito de The landing Zone

O conceito de *Landing Zone* pode ser descrito como uma tabela que define uma região de sucesso de um produto ou projeto. As linhas da tabela contém o subconjunto de requisitos que definem diretamente o sucesso ou a falha (nem todos os requisitos). As colunas da tabela contém uma gama de níveis de desempenho. Geralmente uma *Landing Zone* abrange a faixa entre sucesso (*Outstanding*), alvo (*Target*) e evitar o fracasso (*Minimum*) [12].

The Landing Zone pode ser usada no desenvolvimento ágil para ajudar a definir o sucesso de uma iteração ou um Sprint do Scrum. Dá atenção ao que criará sucesso. Temos um exemplo da aplicação de uma *Landing Zone* na Figura 10.

Figura 10. Exemplo de aplicação do *The landing Zone* [12].

Requirement	Outstanding	Target	Minimum
Retail On Shelf	Nov 15th	Nov. 22nd	Dec 1st
Manufacturing Cost	$9.00	$10.00	$11.50
Peak Project Headcount	250	350	400
Markets at Launch	US, APAC, EMEA	US, APAC	US, APAC
Design Wins	40+	30+	20+
Total First Year Volume	125K	110K	95K

2.3.2.2 Uso da *The Landing Zone*

Landing Zones são úteis para várias coisas:

- Definição de sucesso e fracasso ainda na idealização do projeto.
- Quantificar os níveis de realizações como entrada para viabilidade e análise de risco.
- Orientar discussões e tomadas de decisões ao longo do projeto.
- Monitoramento e comunicação sobre o *status* dos atributos do produto durante o desenvolvimento do projeto.

The Landing Zone ajuda a equipe de desenvolvimento a tomar decisões. Decisões que não violem nenhuma linha de uma *Landing Zone* (LZ) são tomadas pela equipe como uma parte normal de seu trabalho. Enquanto a equipe respeitar as linhas da LZ haverá sucesso [12].

Qualquer decisão que desrespeite os limites impostos pela LZ deve ser tomada pelo gerente do projeto. Isso inclui algo abaixo do *minimum* ou acima do *outstanding*.

The landing Zone pode ser criada para plataformas, componentes, ofertas de serviços, experiências de usuários, projetos, etc.

2.3.2.3 Variantes da *Landing Zone*

Uma variante da LZ adiciona uma quarta coluna chamada *Commit* para monitorar o nível em que o requisito foi implementado no projeto. A figura 11 mostra esta variante da LZ:

Figura 11. *Template* da LZ com a coluna *Commit* adicionada [12].

Requirement	Outstanding	Target	Minimum	Commit

Outra variante é aquela que elimina a coluna *outstanding* e a substitui pela coluna *Kill Switch*. Esta coluna apresenta um valor de alerta, ou seja, se esse valor for atingido deve haver uma reunião da equipe para discutir como o problema deve ser solucionado. A figura 12 mostra esta variante da LZ:

Figura 12. Template da LZ com a coluna Kill Switch adicionada [12].

Requirement	Target	Minimum	Kill Switch

Personalizar o formato e conteúdo de uma *Landing Zone* é importante para o sucesso do projeto [12]. LZ ajuda com a priorização, mas há alguns desafios:

- O formato da LZ deixa muito claro que os requisitos contidos em suas linhas estão competindo por recursos.
- Certas combinações de linhas podem ter maior prioridade que outras combinações, e isto não é facilmente identificado.

Algumas células podem ganhar maior prioridade dependendo de outras. Por exemplo, se a linha 1 atinge o *minimum,* então a linha 11 deve atingir o *outstanding*.

2.3.3 Outras abordagens para Especficação de RNFs

2.3.3.1 KAOS

Entre muitas propostas, as abordagens orientadas à objetivo, como em [22] [5], foram os primeiros a tratar NFRs com mais profundidade. Em [22], talvez pioneira na promoção requisitos orientadas para os objetivos de engenharia, pelo menos do ponto de vista objetivo funcional, os objetivos são: "... modelados por características intrínsecas, tais como a sua natureza e atributos, e por suas ligações com outros objetivos e para outros elementos de um modelo de requisitos. "KAOS aborda ambos os objetivos funcionais e objetivos não-funcionais, que são formalizados em termos de operadores, tais como alcançar, manter e evitar, e por atividades baseadas em lógica temporal acrescida de alguns operadores temporais especiais. Por exemplo, KAOS oferece operadores especiais para conceitos, tais como "em algum momento no futuro" e "sempre no futuro a menos".

Graças à sua natureza formal, representações em KAOS são passíveis de verificação automática e raciocínio. Em KAOS, as metas são caracterizadas como um conjunto de restrições de alto nível sobre os estados. Embora linguagem de representação do KAOS não diferencie entre objetivos funcionais e não funcionais, a linguagem gráfica KAOS faz a diferenciação (*AND*/*OR*). Objetivos de que podem ser atribuídos a agentes individuais não necessitam de decomposição adicional e podem ser "operacionalizados", isto é, eles podem ser descritos em termos de restrições.

2.3.3.2 NFR *Framework*

Não diferentemente do KAOS, o NFR (*Non-Functional Requirements*) *Framework* também promove orientação a metas, mas com ênfase nos NFRs. No NFR *Framework*, requisitos não funcionais são tratados como *softgoals*, ou seja, metas que precisam ser endereçadas não absolutamente, mas em um bom sentido. Este é um reconhecimento das dificuldades que estão associados tanto com o problema e a solução correspondente. No que se refere à definição do problema, é muitas vezes extremamente difícil, se não impossível, definir um termo RNF

completamente sem ambiguidade sem usar qualquer outro termo RNF, que por sua vez terá de ser definido.

No que diz respeito à solução, também é muitas vezes extremamente difícil explorar uma lista completa de possíveis soluções e escolher a melhor solução, ou ótima, devido a várias limitações de recursos, tais como o tempo, mão de obra e dinheiro disponível para tal exploração.

O NFR *Framework* oferece vários tipos diferentes de contribuições pelo qual um *softgoal* satisfaz ou não outro *softgoal* - *MAKE, HELP, HURT* e *BREAK* são os mais proeminentes, bem como AND e OR (estes também incorporaram a noção de "bom o suficiente" em vez de "satisfação absoluta"). MAKE e HELP são usados para representar um softgoal satisfazendo positivamente outro, enquanto BREAK e HURT para representar uma softgoal satisfazendo negativamente (ou negar) outro. Enquanto MAKE e BREAK, respectivamente, refletem o nível de confiança em um softgoal satisfazer totalmente ou negar outro, HELP e HURT respectivamente refletem o nível de confiança em um softgoal parcialmente satisfazer ou negar outro.

No NFR Framework, cada softgoal contribui ou está associada com uma etiqueta, indicando o grau ao qual está satisfeito ou negado. Um procedimento de propagação de etiquetas é oferecido pelo NFR Framework, a fim de determinar o efeito de várias decisões de design, independentemente de serem ou não a nível de sistema ou a nível de software. Além de um uma etiqueta, cada softgoal ou contribuição também pode ser associado com um valor de criticidade, tais como extremamente crítica, críticos e não críticos.

Ao usar o NFR *Framework*, NFRs são definidos como softgoals a serem abordadas ou alcançados, e um processo iterativo e intercalação é aplicado, a fim de satisfazê-los, através de decomposições *AND/OR*, operacionalizações e argumentações.

Ao longo do processo, uma representação visual, SIG (Softgoal Interdependency Graph), é criado e mantido, o que mantém o controle de softgoals e suas interdependências, juntamente com o impacto de várias decisões através de

etiquetas. Neste sentido, um SIG mostra como várias decisões (*design*) são racionalizadas.

A fim de aliviar as dificuldades associadas com a busca de conhecimentos para lidar com NFRs, o NFR Framework oferece métodos para capturar o conhecimento de maneiras de satisfazer NFRs e regras de correlação para o conhecimento dos efeitos colaterais que tais métodos induzem.

Tal como acontece com *goals* em KAOS, softgoals no âmbito NFR estão associados a, e finalmente alcançado, pelas ações de agentes - humanos, *hardware* ou *software*. Isto é consistente com o espírito do modelo de referência [11], no qual os requisitos (funcionais) são satisfeitos através da colaboração entre o comportamento do sistema e fenômenos do ambiente de software que são causadas por agentes no ambiente, embora aqui também se esteja preocupado com softgoals que os requisitos (funcionais) destinam-se a ajudar a alcançar em conjunto com fenômenos ambientais. Note que os requisitos são parte da solução para um problema de uma parte da realidade, e a noção de softgoals pode ser utilizada para representar qualquer coisa que não seja funcional, seja sobre o domínio do problema ou a solução.

O NFR Framework tem uma característica intrínseca que o diferencia de outros métodos NFR - a dependência de uma abordagem qualitativa em direção NFRs ou *softgoals*. No coração do NFR *Framework* encontra-se o conceito de que softgoals são idealizações e, como tal, sem todas as suas propriedades necessariamente estabelecidas.

Capítulo 3
Suporte à especificação de Requisitos não Funcionais

Este capítulo apresenta o processo proposto para suporte à especificação de requisitos não funcionais. Este inclui conceitos da metodologia QaSE relacionando-os aos conceitos da metodologia *The Landing Zone*. Também é apresentado o sistema construído para dar suporte à especificação de RNF.

3.1 Proposta de *Template*

O *template* proposto para esta monografia é apresentado na Tabela 1. Se trata de uma junção do QaSE e *The Landing Zone* apresentados no Capítulo 2. A proposta de modelagem de requisitos não funcionais utilizando este *template* aplicado a projetos ágeis tem o objetivo de possibilitar uma melhor compreensão desses requisitos, explicando as suas interdependências e conflitos. Essa proposta irá contemplar os impactos causados pelo não atendimento às necessidades elicitadas dos *stakeholders*, expor conflitos existentes entre os requisitos não-funcionais. Além disso, determina uma prioridade ao RNF, assim como a justificativa para esta prioridade. Também são traçadas metas entre o mínimo que se espera do requisito e o máxmo a ser alcançado, entre outras coisas.

Tabela 1: *Template* proposto

Atributo	*Informação*
Projeto	Nome do projeto
Autor	Integrante da equipe que especificou o RNF
Nome	Nome do RNF
Descrição	Descrição completa do RNF. Deve ser explicado em que situações ou funcionalidades específicas o RNF deve ser aplicado e de que forma isto deve acontecer.
Interdependências entre RNF	Descrição de quais RNFs possuem alguma dependência em relação ao RNF especificado. Descreve como são as dependências entre os RNFs.
Conflitos entre RNF	Quando há dependências entre requisitos que de alguma forma interferem negativamente no requisito não funcional especificado, gera algum tipo de conflito no sistema que precisa ser descrito. Os conflitos são identificados e analisados, para que seja definido pela equipe como serão tratados.
Impactos do não atendimento ao RNF	O RNF especificado precisa ser atendido, caso não seja, alguns impactos poderão afetar o sistema. Esses impactos precisam ser informados à equipe por meio desse campo,a fim de confirmar a importância daquele RNF para o sistema.
Prioridade	Escada de 1 a 10 que define a prioridade do RNF. Quanto maior o número nesse campo, maior será a prioridade do requisito.
Razão da prioridade	Justificativa do campo anterior.
Mínimo	Se não for possível alcançar o objetivo (próximo ítem), que características mínimas devem ser alcançadas?
Objetivo	Meta de como se deseja que o RNF seja atendido.
Sucesso	Se o objetvo (ítem anterior) for alcançado, o que mais pode ser feito?
Escala	A escala de medida usada para quantificar o requisito.
Medidor	O processo ou dispositivo usado para medir a escala.

3.1.1 Aplicação do *template* a exemplos

Nesta seção são ilustrados dois exemplos de aplicação do *template. Um* relacionado ao sistema de Gestão de Documentos e outro ao Sistema de Gestão de Pacientes conveniados.

3.1.1.1 Exemplo 1: Segurança em um Sistema de Gestão de Documentos

Tabela 2: Exemplo 1

Projeto: Sistema de Gestão de Documentos
Autor: Analista de Requisitos 1
Nome: Segurança na tramitação de documentos
Descrição: O usuário deseja gerenciar os documentos que tramitam dentro da organização. Ao realizar o login no sistema, o usuário escolhe o grupo de trabalho ao qual ele pertence. Quando um

novo documento chegar à organização, o sistema deve permitir que o usuário cadastre as partes envolvidas e os dados do documento. Para realizar a tramitação, o sistema deverá validar novamente o usuário e senha. Após a validação, o sistema permite que o processo de tramitação seja continuado. O usuário tramita o documento e o sistema exibe uma mensagem de confirmação. O log do sistema é atualizado com as operações realizadas.
Interdependências entre RNF: *Segurança x Confiabilidade*: O sistema utiliza algumas medidas de segurança para prover a sigilosidade de documento, assegurar a responsabilidade pela tramitação do documento, tornando o sistema seguro, aumentando assim a confiabilidade nos resultados que serão alcançados.
Conflitos entre RNF: *Segurança x Usabilidade:* O usuário espera realizar suas tarefas rapidamente, sem dúvidas e sem erros causados pela dificuldade de usar ou de aprender. Ao adotar medidas de segurança para prover autenticação, autorização e confidencialidade, a usabilidade é afetada negativamente, uma vez q mais passos serão necessários para se realizar as mesmas ações. É preciso avaliar medidas de segurança mais adequadas para o sistema a ser desenvolvido. *Segurança x Desempenho:* Com a garantia da segurança nas realizações das tarefas no sistema, o desempenho é afetado. Quando são adicionadas medidas de segurança, o sistema torna-se lento pela presença de tantas validações que são realizadas.
Impactos do não atendimento ao RNF • A segurança das informações cadastradas e manuseadas no sistema será comprometida. • A confiabilidade nos resultados obtidos com a tramitação não será garantida, tornando a funcionalidade inútil para o sistema. • O sistema não dará segurança e não será confiável para o usuário.
Prioridade: 8
Razão da prioridade: Segurança é um requisito não funcional bastante importante, pois se a integridade dos dados for comprometida muita informação será perdida ou não terá referência.
Mínimo: A segunda autenticação pode ser retirada, porém, outro tipo de autenticação deve ser implementada, como perguntas sobre o cadastro do usuário.
Objetivo: Descrito no campo “Descrição”.
Sucesso: Descrito no campo “Descrição”.
Escala: Média de vezes que novos usuários completaram a tarefa com sucesso e a segurança foi preservada.
Medidor: Devem ser fetos 30 testes com usuários diferentes para este requisito.

3.1.1.2 Exemplo 2: Disponibilidade no Sistema de Gestão de Pacientes Conveniados

Tabela 3: Exemplo 2

Projeto: Sistema de Gestão de Pacientes Conveniados
Autor: Analista de Requisitos 2
Nome: Disponibilidade do Sistema
Descrição: O sistema deve permitir que o controle e acompanhamento dos pacientes conveniados à cooperativa de fisioterapia sejam realizados em qualquer momento e de qualquer lugar. O sistema deve estar disponível na *web* (*online*), via *browser*, para facilitar o seu uso, independente do local de onde o usuário esteja acessando. O sistema deverá ser tolerante às falhas de software, mantendo os serviços disponibilizados durante o maior tempo possível. O sistema precisa garantir a segurança das informações armazenadas e o banco de dados precisa suportar uma grande quantidade de informações. Se ocorrer um problema e o sistema estiver impossibilitado de funcionar via *web*, o mesmo deve ter uma alternativa que possibilite sua manutenção local, de maneira *offline*.
Interdependências entre RNFs: *Disponibilidade x Confiabilidade*: Ao se manter disponível, tolerante a falhas, o sistema torna-se confiável para o usuário, dando a garantia de uso em qualquer lugar diante de qualquer circunstância. *Disponibilidade x Desempenho:* O desempenho do sistema deve ser mantido estável ao tornar o sistema disponível.
Conflitos entre RNFs:*Disponibilidade x Manutenabilidade*: As modificações, correções, e melhorias no sistema são afetadas pela disponibilidade. O sistema desenvolvido para tolerar qualquer tipo de falha, ao precisar ser feito algum ajuste nele, requererá uma análise criteriosa, pois os impactos poderão ser ainda maiores.
Impactos do não atendimento ao RNF • Os usuários precisam utilizar o sistema de qualquer lugar e a qualquer momento independente das circunstâncias: quedas de energia, falhas de software. • O sistema deve ser desenvolvido a fim de atender a essa necessidade, caso contrário, o seu uso não trará benefício algum para o usuário.
Prioridade: 9
Razão da Prioridade: Disponibilidade é um requisito muito importante em sistemas *web.* Este requisito deve ser respeitado para que as necessidades dos usuários sejam atendidas.
Mínimo: O sistema deve estar disponível 85% do tempo.
Objetivo: O sistema deve estar disponível 95% do tempo.
Sucesso: O sistema deve estar disponível 24 horas por dia, ou seja, 100% do tempo.

Escada: Média do tempo em que o sistema fica disponível.
Medidor: O teste deve ser feito durante 1 semana, 24 horas por dia para verificar sua disponibilidade.

3.2 Sistema de Apoio à Especificação de Requisitos Não Funcionais (SAE-RNF)

Esta seção descreve a proposta do Sistema de Apoio à Especificação de Requisitos não Funcionais (SAE-RNF), com o objetivo de contextualizar o *template* proposto nesta monografia. Este ambiente visa servir como estímulo à adoção da prática da especificação de requisitos não funcionais.

3.2.1 Ferramenta SAE-RNF

A motivação para a criação do SAE-RNF se deve à inexistência de ferramentas integradas que:

- Possibilitem o cadastro dos artefatos que compõem a abstração proposta, que visa ser mais rica;
- Dêem apoio à especificação de requisitos não funcionais estimulando o seu reuso;
- Implementem uma especificação técnica do QaSE ou *The Landing Zone.*

O *SAE-RNF* é uma ferramenta *web,* que se encontra disponível no endereço: www.saernf.orgfree.com, cujos usuários são os engenheiros de negócio que desejam especificar de forma mais detalhada os RNFs, compartilhar seu conhecimento/experiência com outras pessoas, visando o reuso, além de todos aqueles usuários que querem consultar informações relacionadas a problemas e as suas formas de solução.

A linguagem de programação utilizada para desenvolver o SAE-RNF foi a *PHP* em sua versão 5.5.12, por se tratar de uma linguagem bastante difundida, de fácil manutenção e indicada para a plataforma *web*. O banco de dados utilizado foi o *MySQL*, por ser um banco livre, de fácil manipulação, possuindo grande aceitação

entre os que trabalham com desenvolvimento de *software*. A continuidade deste capítulo apresenta a ferramenta SAE-RNF, incluindo as funcionalidades, interface e o seu modelo de dados.

3.2.2 Funcionalidades

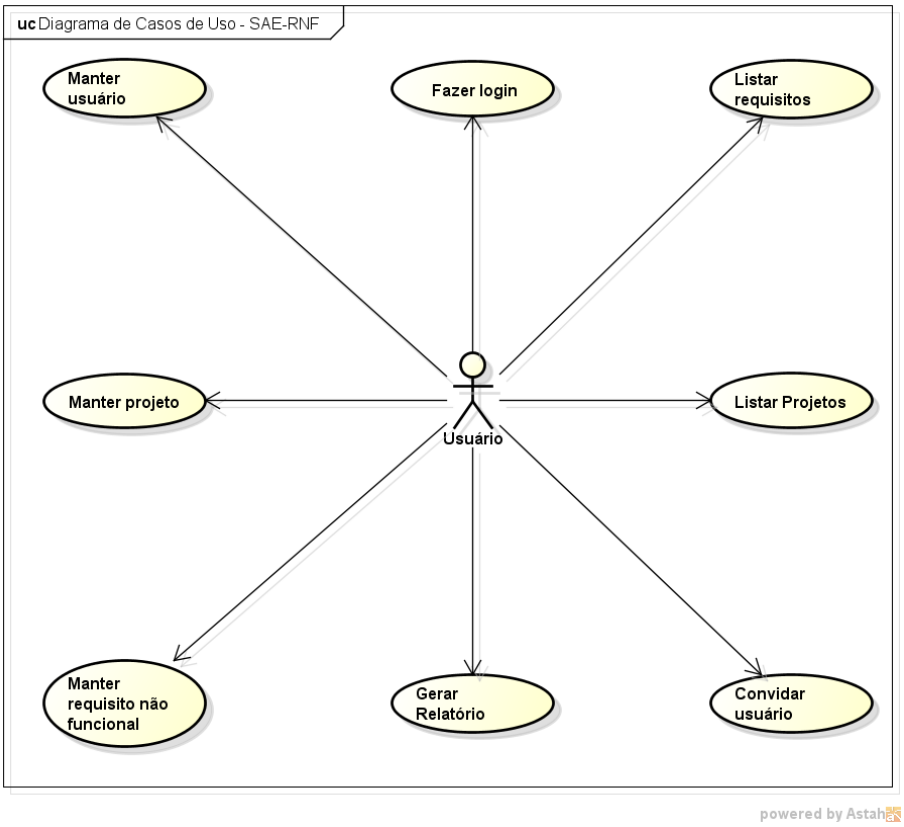
A partir da definição do método de especificação de RNF, foi feito o levantamento dos requisitos para um ambiente que viesse a permitir a especificação desses requisitos.

As principais funcionalidades da ferramenta estão descritas abaixo:

- Cadastro de usuário
- Cadastro de projeto
- Cadastro de requisito não funcional
- Convidar colaboradores
- Impressão do documento ou gerar relatório

A Figura 13 apresenta as principais funcionalidades através do diagrama de caso de uso do *SAE-RNF*.

Figura 13. Diagrama de casos de uso do SAE-RNF.



3.2.3 Interface SAE-RNF

A interface do SAE-RNF é descrita na Figura 14, que representa a tela inicial da ferramenta (Figura 15).

Figura 14. Tela inicial do SAE-RNF.

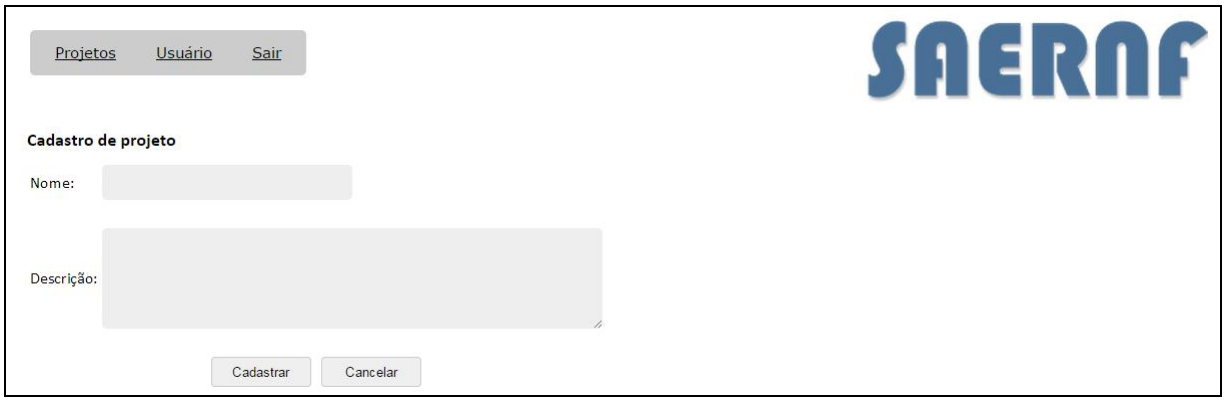
Figura 15. Criar novo usuário.

Após o novo cadastro, o usuário vai ser autenticado no sistema, por meio do acesso à área principal do sistema (Figura 16), que dará ingresso ao menu com as opções de listar projetos, editar cadastro de usuário e sair.

Figura 16. Tela inicial após autenticação do SAE-RNF.

Nesta tela o usuário tem as opções de criar um novo projeto ou visualizar algum projeto existente. A Figura 17 mostra o cadastro de projeto.

Figura 17. Cadastro de projeto.



Quando o usuário visualiza um projeto, o sistema o encaminha para a tela da Figura 18. Nesta tela o usuário tem as seguintes opções:

- **Editar projeto**: O usuário pode mudar o nome e/ou a descrição do projeto.
- **Excluir projeto**: Todos os registros relacionados ao projeto serão apagados do banco de dados.
- **Convidar usuário**: Inclui um usuário no projeto (Figura 19).
- **Gerar relatório**: Gera um arquivo no formato PDF que contém um relatório do projeto, descrevendo todo o seu conteúdo.

- **Desligar participante**: Desfaz o vínculo que existe entre o usuário e o projeto.

- **Criar/visualizar requisito não funcional**: Nesta tela o usuário vai especificar o requisito não funcional (Figura 20).

Figura 18. Visualizar projeto.

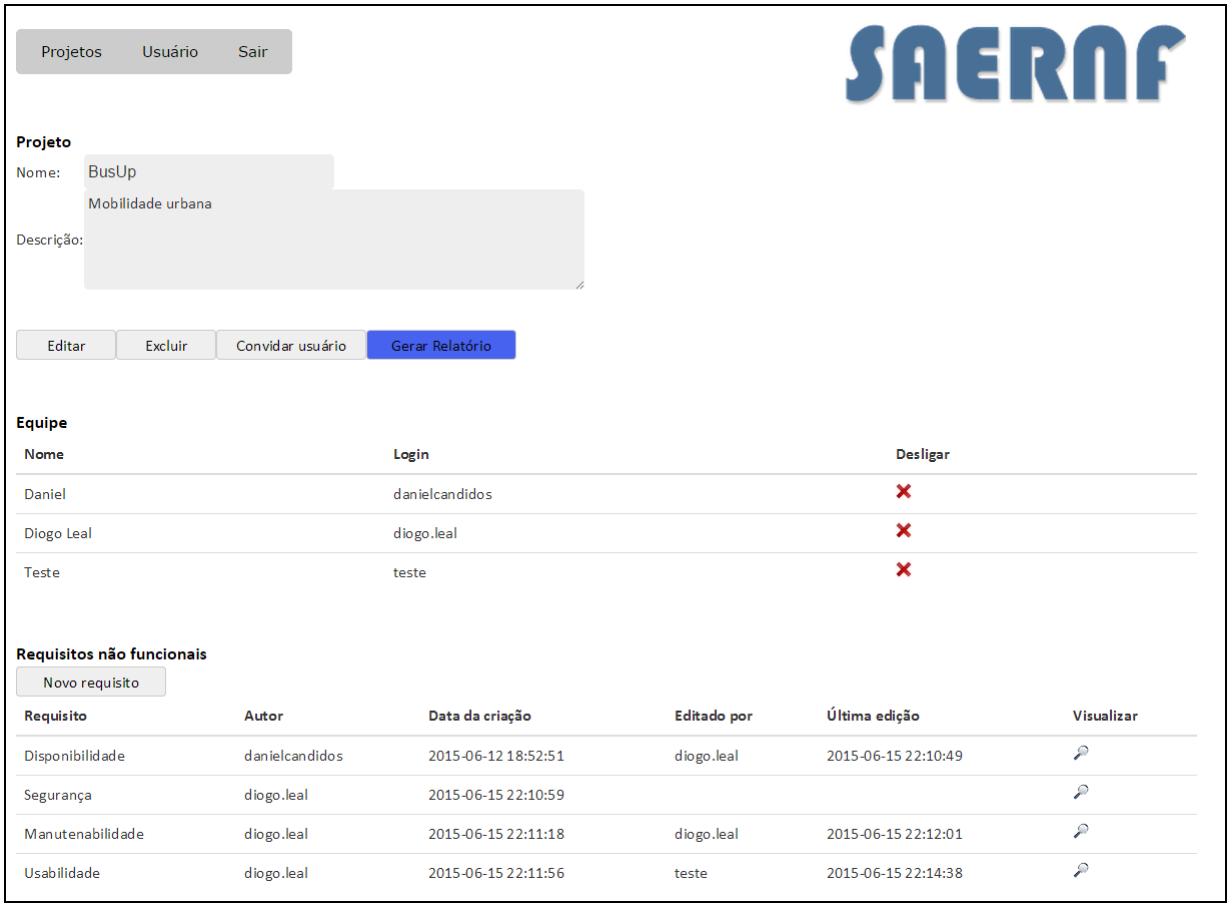


Figura 19. Convidar Usuário.

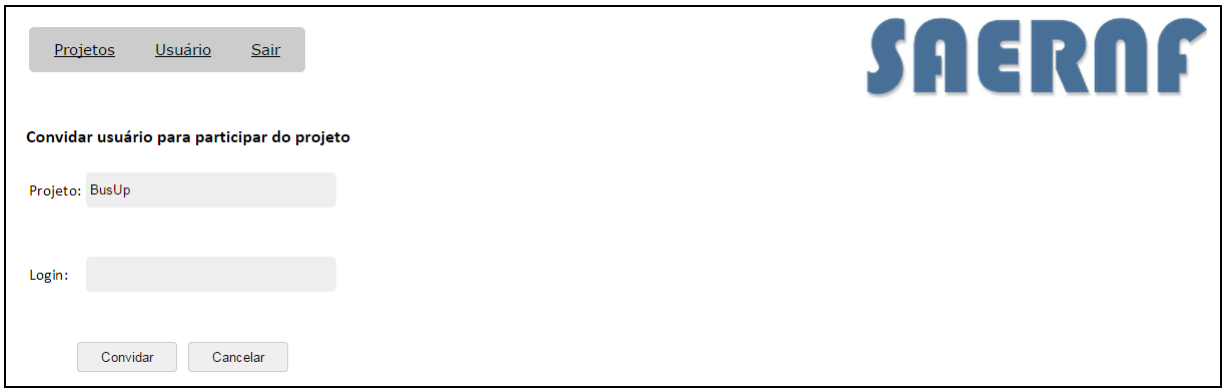
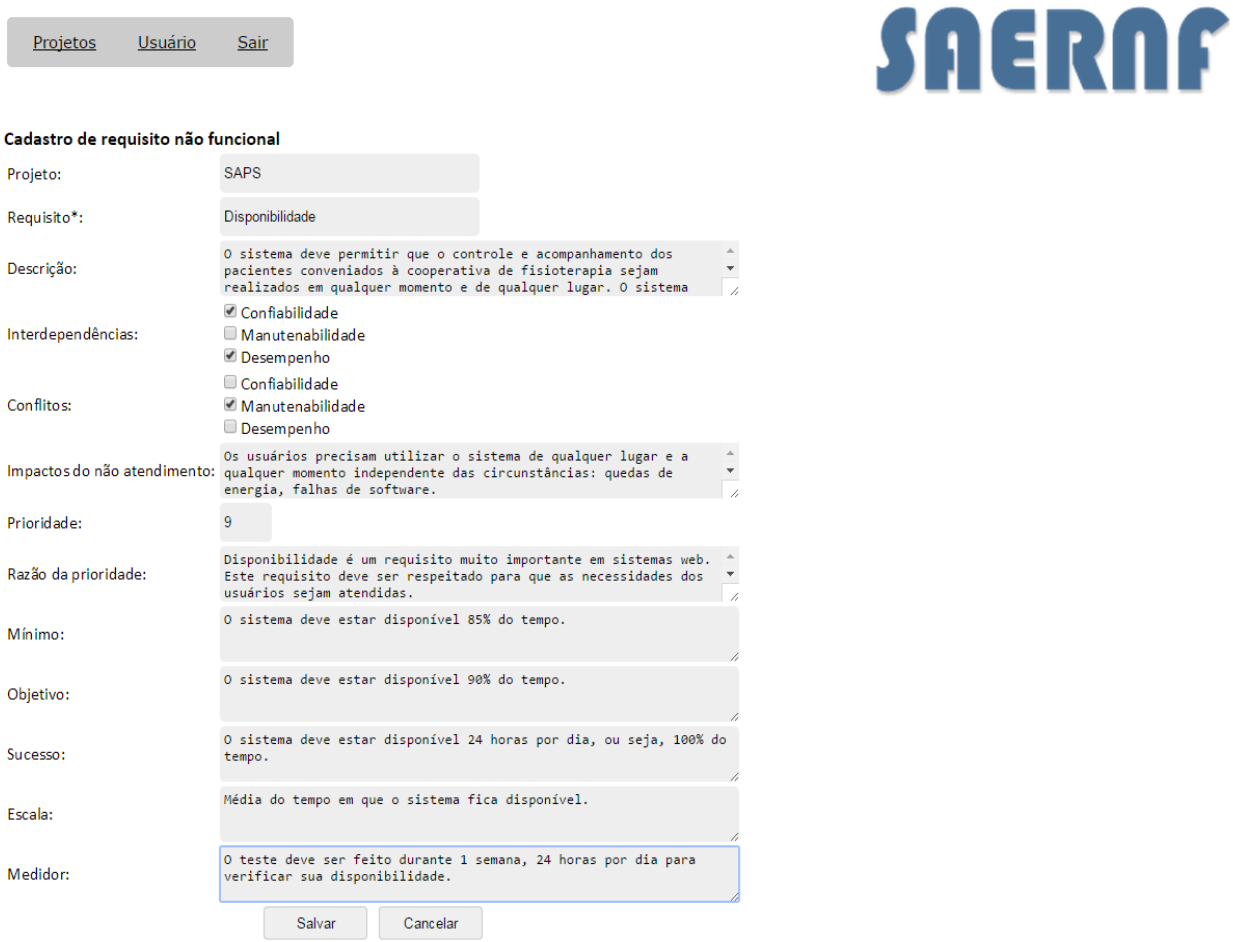


Figura 20. Especficando um RNF no SAE-RNF.



3.2.4 Modelo de Dados

O modelo conceitual de dados Entidade-relacionamento, usado como base da ferramenta, é apresentada na Figura 21. Já o modelo lógico é apresentado na Figura 22.

Figura 21. Modelo conceitual de dados do SAE-RNF.

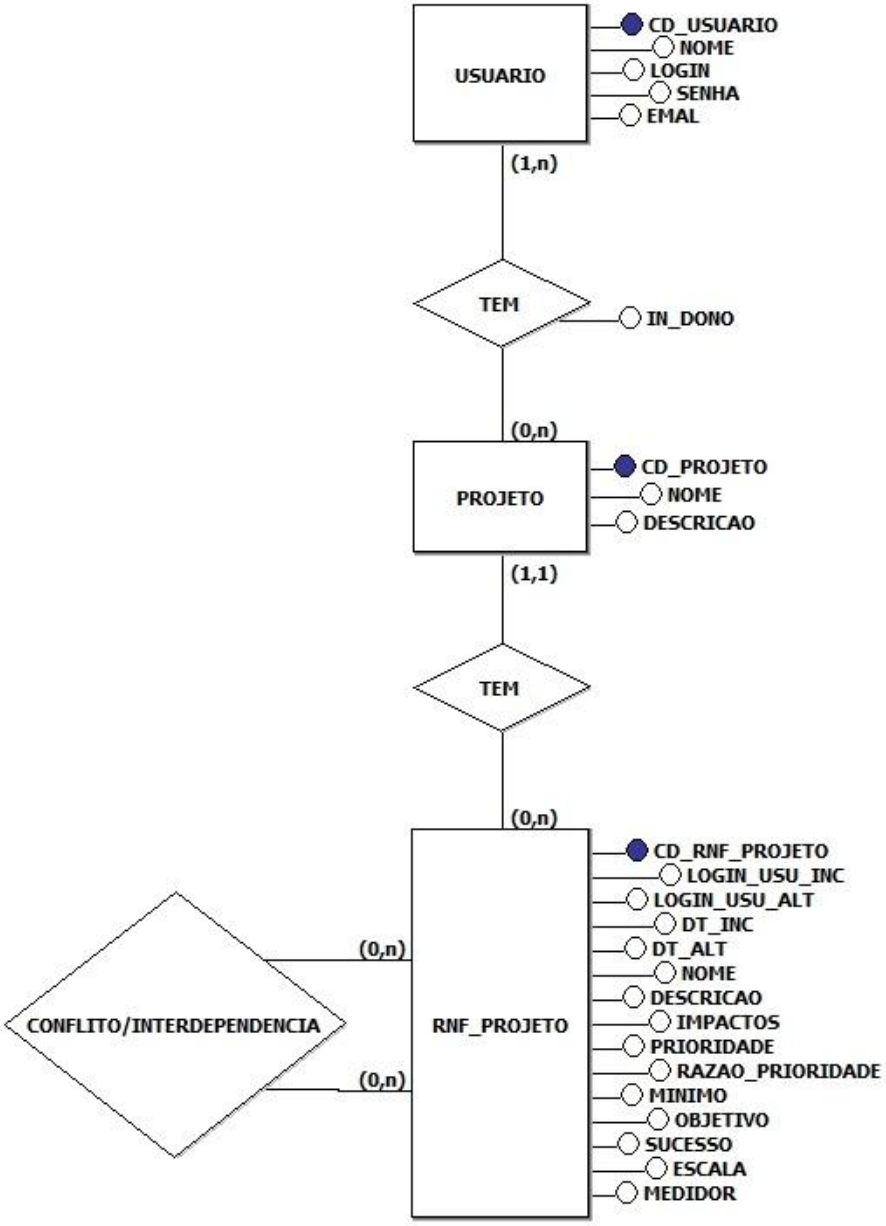
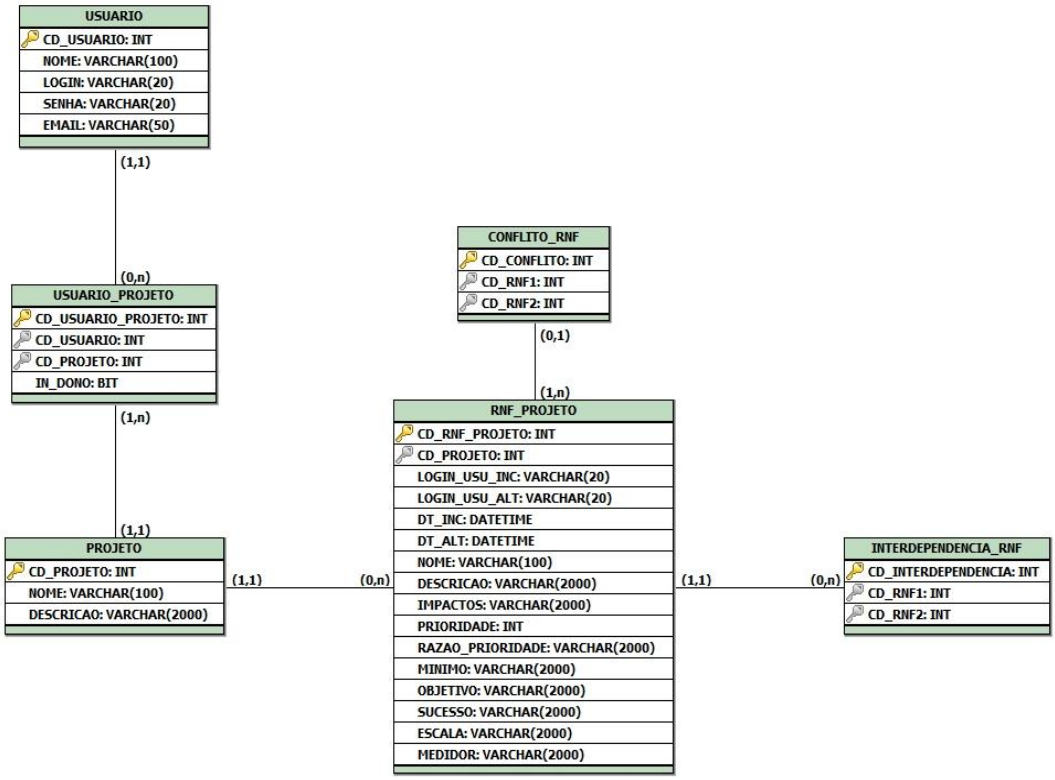


Figura 22. Modelo de dados lógico do SAE-RNF.



Capítulo 4
Conclusão e Trabalhos Futuros

Este capítulo tem como objetivo apresentar algumas considerações finais sobre os principais tópicos abordados nesta monografia e indicações para trabalhos futuros.

4.1 Considerações Finais

A Engenharia de Requisitos é uma fase muito importante na Engenharia de *Software* para o desenvolvimento de sistemas. É a fase em que a equipe de desenvolvimento entende as necessidades dos *stakeholders*. Toda a estrutura e funcionalidades do sistema são determinadas nessa fase do projeto.

Elicitar requisitos é uma tarefa que exige um cuidado especial para que os requisitos sejam compreendidos corretamente e atendam às expectativas neles lançado, uma vez que é o momento no qual os *stakeholders* expõem suas necessidades e a visão do sistema a ser desenvolvido. Por abordar requisitos, necessidades e características do sistema que estão sujeitos a constante mudança, a Engenharia de Requisitos é uma fase do desenvolvimento de *software* onde erros e inadequações a esses requisitos são comuns, podendo não atender às reais necessidades da organização, resultando em retrabalho, custo e principalmente, na insatisfação do cliente.

Durante a elaboração desta monografia, verificou-se que os requisitos não funcionais não recebem a devida importância na fase de elicitação de requisitos. Normalmente, o foco principal da implementação está em atender aos requisitos funcionais, os requisitos não funcionais são verificados apenas na conclusão do desenvolvimento do projeto. Esta prática tem levado muitos projetos ao fracasso. Além disso, a correta compreensão de requisitos não funcionais e de suas relações, como interdependências e conflitos não é alcançada pela escassez de informações a respeito do RNF.

Os requisitos não-funcionais têm uma influência direta no alcance da satisfação do cliente. Eles estão relacionados à eficiente execução das funcionalidades no sistema preocupando-se em proporcionar um fácil uso e aprendizado do sistema, uma interface amigável e compreensível, segurança nas informações inseridas, confiabilidade nos resultados atingidos. Dessa forma, foi vista a necessidade de uma melhoria na abordagem realizada nesses requisitos a fim de proporcionar uma compreensão correta de cada requisito não-funcional, auxiliando na atividade de desenvolvimento do sistema, possibilitando o atendimento às necessidades do cliente ao final do projeto.

4.1.1 Contribuições

Baseado em duas técnicas de especificação de requisitos não funcionais, *The Landing Zone* e QaSE, foi desenvolvido um *template* que auxilia a modelagem dos RNFs, considerando a importância de uma especificação mais efetiva. Foi feita também a aplicação do *template* a exemplos existentes na litetarura, modelando os requisitos não funcionais, fornecendo um entendimento dos requisitos e de sua importância para o sistema. O SAE-RNF (Sistema de Apoio à Especificação de Requisitos não Funcionais), foi desenvolvido visando facilitar a prática do uso do *template* proposto nesta monografia.

4.1.2 Limitações

O *template* proposto se limitou a integração de duas abordagens, não incorporando conceitos referentes a outras propostas que de alguma forma poderiam enriquecer a proposta, mantendo-a ao mesmo tempo leve.

O *template* ainda não foi usado por outras pessoas, nem aplicado em casos reais da literatura.

O sistema desenvolvido foca apenas nos requisitos não funcionais, não estando integrado ao processo de requisitos como um todo.

4.2 Trabalhos Futuros

Como recomendações e sugestões de trabalhos futuros para evoluir a presente pesquisa pode-se destacar:

- Aplicar o *template* proposto a casos reais em empresas e/ou na academia.
- Validar o SAERNF com usuários, avaliar a usabilidade e receber sugestões de melhoria.
- Fazer um estudo comparativo entre o SAE-RNF e outras ferramentas existentes no mercado.
- Rever o processo sistematizado dos RNFs como um todo.
- Estender o projeto com uma ferramenta de visualização, como o NFR de Chung.
- Integrar especificação ao modelo e ferramenta 4REUse [7].
- Migrar o protótipo para nuvem e fazer um *WebService*, o que pode facilitar a integração com outras ferramentas.

Bibliografia

[1] BROOKS, F.: No silver bullet: Essence and accidents of software engineering. IEEE computer, Vol. 20, 10–19, 1987.

[2] BASS, L.; KLEIN M.; BACHMANN, F.: Quality Attribute: Design Primitives. Architecture Tradeoff Analysis Initiative. CMU/SEI- CMU/SEI-2000-TN-017. December, 2000.

[3] CHICHINELLI, M.; CAZARINI, E.: Requisitos Não-funcionais e sua importância no desenvolvimento de sistemas de informação. In: ENEGEP2001 - *National Meeting of Industrial Engineering,* 2001.

[4] CHUNG, L., Nixon, B., Yu, E.; Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Springer, 2000.

[5] CHUNG, L., NIXON, B.A., YU, E., MYLOPOULOS, J.: Non-Functional Requirements in Software Engineering. International Series in Software Engineering, vol. 5, p. 476. Springer, Heidelberg (1999).

[6] CHUNG, L.; MYLOPOULOS, J.; NIXON, B. Representing and Using Non Functional Requirements: A Process-Oriented Approach. In: IEEE Transactions on Software Engineering – Vol.IS.Nº6,1992.

[7] COELHO, M. 4REuse: um Método e uma Ferramenta para o Apoio à Especificação de Requisitos baseado em Reuso. 2012. Dissertação (Mestrado) - Universdade de Pernambuco, Escola Politécnica de Pernambuco. Recfe (2012).

[8] CYSNEIROS, G.; LEITE, J. C.: Utilizando Requisitos Não-funcionais para Análise de Modelos Orientados a Dados. In: WER – *Workshop Requirements Engineering*, 1998.

[9] CYSNEIROS, L.M.: Requisitos Não-Funcionais: Da Elicitação ao Modelo Conceitual. Tese de Doutorado, Departamento de Informática, PUC Rio, Rio de Janeiro, 2001.

[10] GLIB, T. Competitive Engineering: A handbook for systems engineering, requirements engineering, and software engineering using planguage. 2005. Oxford: Elsevier, 2005. 373p.

[11] GUNTER, C., GUNTER, E., JACKSON, M., ZAVE, P.: A Reference Model for Requirements and Specifcations. IEEE Software, 37–43 (2000).

[12] GREGORY, S.: Writing Good Requirements. Intel Corporation. IEEE RE13, 2013.

[13] JACOBSON, I. Essential Practices – The Smart Way. Disponível em: http://www.ivarjacobson.com/resource.aspx?id=408. Último acesso em: 21/10/2010.

[14] KIRNER, T. G. and Davis, A. M.: Nonfunctional requirements of real-time systems. Advances in Computers, Vol. 42, 1–37, 1996.

[15] KOTONYA, G.; SOMMERVILLE, I.: Requirements Engineering – Processes and Techniques. England: John Wiley Professional, 294p, 1998.

[16] LAWSWEERDE, A.; DARIMONT, R. e LETIER, E..: Managing Conflicts in Goal-Driven Requirements Engineering. IEEE Transaction on Software Engineering. Special Issue on Managing Inconsistence in Software Development,1998.

[17] LEFFINGWELL, D.; WIDRIG, J.: Managing Software Requirements – A Unified Approach. United States of América: Addison-Wesley, 491p , 2001.

[18] LUFTMAN, J. N., PAPP, R., and BRIER, T.: Enablers and inhibitors of business-it alignment. Communications of AIS, page p. 1. 1999.

[19] NUSEIBEH, B.; EASTERBROOK, S.*:* Requirements Engineering: A Roadmap. In A. C. W. Finkelstein (ed) "The Future of Software Engineering, 2000.

[20] ROBERTSON, S.; ROBERTSON, J.: Customer Satisfaction/dissatisfaction – Mastering the Requirements Process (Volere). 2ª ed, 2006.

[21] SALES DE BRITO, R.: Uma proposta para Modelagem de Requisitos Não Funcionais em Projetos Ágeis. Dissertação (Mestrado em Ciência da Computação) - Universidade Federal de Pernamuco - UFPE, 2010.

[22] VAN LAMSWEERDE, A.: Goal-Oriented Requirements Engineering: A Guided Tour. In: Proceedings of the 5th IEEE international Symposium on Requirements Engineering, August 27-31, 2001, p. 249. IEEE Computer Society, Washington (2001).

[23] XAVIER, L.: Integração de Requisitos Não-Funcionais a Processos de Negócio: Integrando BPMN e RNF. Dissertação (Mestrado) - Universidade Federal de Pernambuco, Recife, 2009.