UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS I
CENTRO DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO

KARINE XAVIER SILVA

RNFCase: UMA PROPOSTA VOLTADA À DOCUMENTAÇÃO DE REQUISITOS FUNCIONAIS, NÃO FUNCIONAIS E RISCOS

CAMPINA GRANDE
2020

KARINE XAVIER SILVA

RNFCase: UMA PROPOSTA VOLTADA À DOCUMENTAÇÃO DE REQUISITOS FUNCIONAIS, NÃO FUNCIONAIS E RISCOS

Trabalho de Conclusão de Curso de Graduação em Ciência da Computação da Universidade Estadual da Paraíba, como requisito à obtenção do título de Bacharel em Ciência da Computação.

Área de concentração: Engenharia de Software.

Orientador: Prof.ª MSc. Luciana de Queiroz Leal Gomes

CAMPINA GRANDE
2020

S586r Silva, Karine Xavier.
RNFCase [manuscrito] : Uma proposta voltada à documentação de requisitos funcionais, não funcionais e riscos / Karine Xavier Silva. - 2020.
74 p. : il. colorido.

Digitado.

Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências e Tecnologia , 2020.

"Orientação : Profa. Ma. Luciana de Queiroz Leal Gomes , Coordenação do Curso de Computação - CCT."

1. Engenharia de Requisitos. 2. Modelagem de Requisitos. 3. Engenharia de Software. I. Título

21. ed. CDD 005.1

Elaborada por Giulianne M. Pereira - CRB - 15/714 BC/UEPB

KARINE XAVIER SILVA

RNFCase: UMA PROPOSTA VOLTADA À DOCUMENTAÇÃO DE REQUISITOS FUNCIONAIS, NÃO FUNCIONAIS E RISCOS

Trabalho de Conclusão de Curso de Graduação em Ciência da Computação da Universidade Estadual da Paraíba, como requisito à obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 27 de Fevereiro de 2020.

Luciana de Queiroz Bezerra Gomes

Profa. MSc. Luciana Queiroz (DC - UEPB)
Orientador(a)

Prof. Dr. Daniel Scherer (DC - UEPB)
Examinador(a)

Sabrina de F. Souto.

Profa. Dra. Sabrina de Figueirêdo Souto (DC - UEPB)
Examinador(a)

Aos meus pais e irmã, pelo apoio, companheirismo e amor, DEDICO.

AGRADECIMENTOS

Ao meu Deus pelas suas misericórdias e fidelidade para comigo, sem Ele nada poderia ser conquistado, pois é o meu refúgio em todo o tempo.

Aos meus pais e minha irmã Cristiane pelo carinho, apoio, paciência e por toda a ajuda possível. É nos momentos de dificuldade que percebemos a importância dessa base para nos mantermos no caminho certo.

A minha orientadora Luciana de Queiroz pela paciência, palavras de motivação, orientação e por todo o seu tempo dedicado a este trabalho, o meu muito obrigada.

Aos meus colegas de classe, por todos os momentos passados durante o curso, a amizade, o incentivo e a cumplicidade de vocês também foram determinantes para a conclusão desta graduação.

A todo corpo docente do curso de Ciência da Computação, por todos os ensinamentos, os quais contribuíram para minha vida profissional e pessoal.

“Se você não identifica os requisitos certos, não importa quão bem você execute o resto do projeto.”

(Karl Wiegers)

RESUMO

Desenvolver softwares de qualidade, que satisfaçam as necessidades dos usuários, tem sido um desafio para a área de Engenharia de Software, pois entregar ao cliente algo diferente daquilo que ele necessitava e que não consegue trazer retorno ao investimento é extremamente dispendioso. Portanto, é uma tarefa essencial para o desenvolvimento de software, a obtenção, a análise e documentação dos requisitos do sistema junto ao cliente. A engenharia de requisitos propõe um conjunto de técnicas que tem por objetivo o entendimento entre clientes e equipe de desenvolvimento sobre as necessidades dos usuários (os requisitos) e para que isso seja feito da melhor forma possível. Estes comumente são divididos em requisitos funcionais, aqueles que determinam as funções do sistema; e não funcionais, aqueles que determinam condições ou qualidades que o sistema deve atender. Para que o entendimento dos requisitos entre seus interessados seja feito é comum a utilização de modelos, os quais mostram diferentes visões sobre os requisitos do sistema. Em razão da importância dos requisitos e da modelagem e especificação, buscamos explorar a partir de uma pesquisa bibliográfica os modelos existentes a fim de identificar deficiências ou questões não abordadas, com o intuito de desenvolver novas possibilidades para um melhor entendimento dos requisitos de software por parte dos seus interessados. Identificamos que embora os tipos de requisitos se relacionem entre si e tenham um impacto mútuo, essas relações não são mostradas com clareza nos modelos ou se tem uma preferência por um tipo específico de requisito. Esse trabalho, então, se propôs a desenvolver um modelo de software em que fosse possível mostrar com mais clareza as relações existentes entre os tipos de requisitos, resultando no modelo RNFCase que é a principal contribuição deste trabalho.

Palavras-Chave: Engenharia de Requisitos. Modelagem de Requisitos. Engenharia de Software.

ABSTRACT

Developing quality software that meets the needs of users has been a challenge for the Software Engineering area, because delivering to the customer something different from what he needed and that cannot bring return on investment is extremely expensive. Therefore, it is an essential task for software development, obtaining, analyzing and documenting system requirements with the customer. Requirements engineering proposes a set of techniques aimed at understanding between customers and the development team about users' needs and that this is done in the best possible way. Software requirements are commonly divided into functional requirements, those that determine system functions, and non-functional requirements, those that determine conditions or qualities that the system must meet. For the understanding of the requirements among its stakeholders to be made it is common to use models, which show different views on the requirements of the system. Due to the importance of requirements modeling and specification, we seek to explore existing models from a bibliographic search in order to identify deficiencies or issues not addressed, in order to develop new possibilities for a better understanding of software requirements by of your stakeholders. We identified that although the types of requirements are related to each other and have a mutual impact, these relationships are not shown clearly in the models or there is usually a preference for a type of requirement. This work, then, proposed to develop a software model in which it was possible to show more clearly the relationships between the types of requirements, resulting in the RNFCase model which is the main contribution of this study.

Keywords: Requirements Engineering. Requirements Modeling. Software Engineering.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BPMN	*Business Process Modelling Notation*
RF	Requisitos Funcionais
RNF	Requisitos Não Funcionais
NFR	*Non-Functional Requirements*
UML	*Unified Modeling Language*
UML-MC	Extensão da UML que suporta Mapas Conceituais

SUMÁRIO

CAPÍTULO 1 – INTRODUÇÃO

1.1 Motivação e Contexto

Os sistemas computacionais hoje em dia têm sido utilizados em todas as áreas da vida humana a fim de facilitar e melhorar a realização de tarefas. Todavia, apesar do crescente conhecimento em linguagens e técnicas de programação, para desenvolver softwares de qualidade é necessário não apenas o conhecimento computacional, mas também o entendimento das necessidades do cliente. Nesse sentido, Magalhães (2006) afirma que para um software ter qualidade "é necessário que o software esteja em conformidade com os seus requisitos, atenda aos requisitos e expectativas do cliente e seja bem aceito por seus usuários".

Portanto, para desenvolver bons softwares é fundamental adquirir conhecimento sobre o usuário e suas necessidades, a fim de que sejam produzidos sistemas que não apenas ofereçam um conjunto de funções, mas que satisfaçam os seus usuários (CHAVES;AGUIAR, 2016).

Para que esse objetivo seja alcançado, é importante que se tenha uma preocupação em fazer um bom levantamento e análise dos requisitos do sistema utilizando os princípios e práticas da Engenharia de Software a fim de ter como resultado softwares bem planejados e construídos (GASTALDO; MIDORIKAWA, 2003).

Entendem-se por requisitos de um sistema as descrições daquilo que um sistema deve oferecer aos seus usuários e refletem as necessidades dos clientes que serão atendidos por ele. Esse processo que inclui as etapas de descobrir, analisar, documentar e verificar esses serviços e restrições é denominado Engenharia de Requisitos (SOMMERVILLE,2018).

O não entendimento dos requisitos do cliente pela equipe responsável pelo software, pode acarretar em graves prejuízos, uma vez que ao avançar as fases do processo de desenvolvimento torna-se cada vez mais caro consertar um erro no sistema (CHAVES; AGUIAR, 2016).

Um relatório emitido pela The Standish Group Internacional (2015) mostrou que o sucesso de projetos de software tem sido raro, haja vista que apenas 36% destes são lançados com sucesso (no prazo, dentro do orçamento e com escopo completo), com 19% deles cancelados ou nunca usados e 45% mudaram (atrasaram, estouraram o orçamento, e/ ou reduziram o escopo). Por isso é essencial a boa compreensão dos requisitos dos usuários e isso pode ser feito através da modelagem de requisitos.

A engenharia de requisitos possui múltiplas técnicas de modelagem, estas tentam tornar claro tanto para o cliente como para a equipe de desenvolvimento aquilo que o sistema deve

oferecer ao usuário. Cada tipo de modelo de requisitos busca mostrar um tipo diferente de visão sobre o sistema, seja ela de interação, estrutural ou comportamental (GASTALDO; MIDORIKAWA, 2003).

Os requisitos estabelecidos pelo cliente são comumente separados em requisitos funcionais, que são aqueles que irão determinar as funções do sistema, e requisitos não funcionais, que são os que determinarão o comportamento do software ao executar essas funções, muito embora seja difícil fazer essa separação, pois os requisitos geralmente estão intimamente relacionados (SOMMERVILLE,2018).

Todavia, é comum dentro dos documentos de especificação de requisitos a separação dos requisitos funcionais dos não funcionais sem deixar claro a relação que existe entre eles, o que pode dificultar o entendimento das necessidades do cliente (SOMMERVILLE, 2018). Pois, a relação entre eles deixa claro a motivação por trás do estabelecimento de um requisito, influindo diretamente em sua rastreabilidade, como também os requisitos terem a capacidade de impactar entre si de forma negativa ou positiva.

Considerando a dificuldade de encontrar modelos que mostrem uma visão clara da relação existente entre os requisitos funcionais e não funcionais de software, optou-se neste estudo propor um modelo de requisitos que satisfizesse essa necessidade, a fim de fornecer um melhor suporte a equipe de desenvolvimento do software e ao cliente no entendimento do problema.

1.2 Problema de Pesquisa

Este trabalho emerge da necessidade de desenvolver um modelo capaz de obter uma visão mais clara dos relacionamentos entre os tipos de requisitos do sistema. Com isto foi obtida a seguinte pergunta-problema: “É possível desenvolver uma alternativa de modelagem em que se possa identificar de forma clara o relacionamento entre os tipos de requisitos, como também entre os requisitos não funcionais, de forma ajudar no entendimento das necessidades do cliente?”

1.3 Objetivos

1.3.1 Objetivo Geral

Desenvolver um modelo de requisitos em que se possa colocar em evidência a relação entre os requisitos funcionais e não funcionais, considerando que os outros modelos de software não o fazem ou não fazem de forma clara essa relação.

1.3.2 Objetivos Específicos

- Associar requisitos funcionais aos requisitos não funcionais em diagrama para que fique mais clara a visão geral dos requisitos mais importantes e sua relação.
- Incluir elementos ao diagrama que possam evidenciar os riscos envolvidos com os requisitos, como também o impacto mútuo entre os requisitos.
- Estabelecer métodos para o uso do modelo.
- Desenvolver uma alternativa de modelagem para os requisitos não funcionais.
- Fazer uma comparação entre a proposta de modelagem e os demais modelos existentes.

1.4 Metodologia

Nesta seção será apresentada a metodologia a qual foi dirigida este trabalho, a fim de expor o tratamento sistemático ao qual se deu o planejamento e desenvolvimento da pesquisa e criação do modelo que se objetiva esta monografia.

Em relação a classificação da pesquisa, quanto aos objetivos, pode-se defini-la como um estudo exploratório, por inicialmente observar os processos de modelagem e especificação de requisitos já estabelecidos a fim de identificar deficiências ou questões não abordadas, com o intuito de desenvolver novas possibilidades para um melhor entendimento dos requisitos de software por parte dos seus interessados. Quanto aos procedimentos técnicos, foi feita uma pesquisa bibliográfica para suprir as informações necessárias para o desenvolvimento dos objetivos de pesquisa e, posteriormente, foi desenvolvido um modelo e realizado experimentos para a criação e refinamento da proposta (WAZLAWICK, 2010).

A metodologia adotada para esta pesquisa consistiu em seis etapas. Na primeira etapa deste estudo foi realizado um levantamento bibliográfico necessário para o estudo a fim de explorar o conhecimento já estabelecido na academia.

Por conseguinte, na segunda etapa foi realizada uma análise crítica relacionada ao material obtido na primeira etapa, com o objetivo de determinar as intervenções a serem sugeridas concernentes a modelagem e especificação de requisitos.

Na terceira etapa foi desenvolvida uma proposta denominada RNFCase com base nas deficiências identificadas nos modelos encontrados na segunda etapa da pesquisa. Decidiu-se por adaptar o diagrama Casos de Uso, por sua facilidade de entendimento e flexibilidade, para incluir novos elementos considerados como relevantes para facilitar o entendimento dos requisitos do sistema.

Na etapa seguinte, a quarta, o modelo proposto foi aplicado em exemplos de softwares (acadêmico e bancário) com a finalidade de avaliar experimentalmente seu uso.

Após a aplicação, na quinta etapa, o modelo foi refinado observando os problemas encontrados na proposta e objetivando trazer mais clareza ao leitor do modelo. Nesta etapa também, foi estabelecido um método para uso do modelo a fim de facilitar o uso da proposta.

Na sexta etapa, foi feita uma comparação entre o modelo proposto, RNFCase, com outros modelos já criados, verificando as vantagens e desvantagens do mesmo.

Figura 1 – Metodologia utilizada na pesquisa

Fonte: Elaborada pelo autor, 2019.

Por fim, os dados deste estudo após analisados e observados os resultados criticamente foram reunidos a fim de serem apresentados na escrita desta monografia.

1.5 Organização do Trabalho

Esta seção destina-se a apresentação da organização deste trabalho. No primeiro capítulo foi apresentada a introdução com seu contexto e motivação, o problema de pesquisa, os objetivos e a metodologia seguida por este estudo.

No segundo capítulo, é apresentada a fundamentação teórica, na qual conceituamos requisito de software e seus tipos, apresentamos os diversos tipos de modelagem utilizados e os trabalhos relacionados a este.

No terceiro capítulo, apresentamos a modelagem de Casos de Uso com mais detalhes e desenvolvemos a proposta do RNFCase com todos os seus elementos. Como também,

definimos algumas considerações sobre a proposta e fazemos uma comparação com outros tipos de modelagem e especificação de requisitos.

No quarto capítulo, demonstramos a utilização do RNFCase em dois tipos de sistemas, um sistema de controle escolar e um sistema de aplicação bancária.

No quinto capítulo apresentamos a conclusão deste trabalho relatando os resultados alcançados e limitações, como também a apresentação dos trabalhos futuros.

CAPÍTULO 2 - FUNDAMENTAÇÃO TEÓRICA

Nesta seção, abordar-se-á os conceitos sobre requisitos de software, requisitos de domínio, requisitos de negócio, requisitos funcionais, requisitos não-funcionais e requisitos inversos. Também será conceituado Engenharia de Requisitos, documentação de requisitos, técnicas de modelagem e escrita de requisitos, riscos e serão analisados trabalhos anteriores.

2.1 Requisitos de Software

Antes do desenvolvimento de um software se faz necessário decidir o que será construído, esta é uma questão crucial para o projeto, pois entregar ao cliente algo diferente daquilo que ele deseja é um desperdício de tempo e recursos. Fazer modificações depois do software pronto é algo incrivelmente custoso, logo é essencial determinar tempo para entender o que o cliente realmente deseja do software no início do desenvolvimento. Torna-se, portanto, uma atividade essencial, ao se projetar e construir sistemas, o levantamento e análise dos requisitos junto aos usuários.

Os requisitos de um sistema são a entrada para o projeto e desenvolvimento de um software, eles são descrições daquilo que um sistema deve oferecer aos seus usuários. Segundo Sommerville (2011, p. 57) “os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferece e as restrições a seu funcionamento. Esses requisitos refletem as necessidades dos clientes para um sistema que serve a uma finalidade determinada [...]”.

Para Machado (2016, n.p.):

> Os requisitos expressam as características e restrições do produto de software do ponto de vista de satisfação das necessidades do usuário e, em geral, independem da tecnologia empregada na construção da solução, sendo a parte mais crítica e propensa a erros no desenvolvimento de software. (MACHADO, 2006, n.p.)

Os requisitos de software geralmente são divididos em dois tipos: requisitos funcionais e requisitos não funcionais. Os requisitos funcionais, de forma geral, são os responsáveis por determinar “o que” o sistema irá fazer, quais as funções o sistema deve fornecer para o usuário. Por outro lado, os requisitos não funcionais irão responder “como” o sistema irá realizar essas funções. Estes são frequentemente associados a questões de qualidade como segurança, usabilidade, confiabilidade, entre outros. Todavia, essa separação entre requisitos não é tão clara, pois um requisito não funcional como segurança que implique em impedir acesso não autorizado, pode se desdobrar em vários requisitos funcionais como o uso de login e senha para

ter acesso ao sistema. Logo, os requisitos são, no geral, dependentes entre si, podendo ou não gerar ou restringir outros requisitos.

Deve-se considerar também que além destes tipos de requisitos, existem ainda outros que são igualmente relevantes e devem ser considerados ao se realizar o levantamento de requisitos, como os requisitos de domínio, os requisitos de negócio e os requisitos inversos.

2.1.1 Requisitos de Domínio

Os requisitos de domínio são o ponto inicial da engenharia de requisitos, eles surgem do campo de análise denominado domínio do problema, onde se concentrará os esforços da Engenharia de Requisitos para propor uma mudança (VASQUEZ e SIMÕES, 2016).

Eles são os responsáveis por definir o escopo inicial do sistema no que concerne a sua área de atuação, que pode se concentrar: em uma organização ou em áreas funcionais/departamentos desta; nas partes interessadas chave, como os agentes responsáveis por elaborar e perseguir os objetivos da empresa através de seus recursos; e as interações das partes interessadas com os recursos que estão são disponíveis dentro da organização (VASQUEZ e SIMÕES, 2016).

2.1.2 Requisitos de Negócio

São requisitos que indicam o porquê da empresa querer o desenvolvimento do sistema. Segundo Vasquez e Simões (2016,n.p.):

> Requisitos (ou necessidades) de negócio são declarações de mais alto nível de objetivos, metas ou necessidades da organização. Eles descrevem as razões pelas quais um projeto foi iniciado, as metas que o projeto deve atingir e as métricas que serão utilizadas para aferir o seu sucesso. (VASQUEZ e SIMÕES, 2016, n.p)

Entender os requisitos de negócio significa entender quais as reais necessidades do cliente, e isso é importante principalmente para definir prioridades e o que irá pertencer ou não ao escopo do projeto, pois qualquer requisito da solução que não esteja associado a algum requisito de negócio não deve ser considerado como necessário.

2.1.3 Requisitos Funcionais

Quando se pensa em desenvolver um software, os requisitos funcionais são os primeiros que vem à mente, pois são a essência do novo software, sua razão de ser. Segundo o Swebok

(2014) os requisitos funcionais descrevem as funções que o software deve executar, por exemplo, formatando algum texto ou modulando um sinal. Conhecidos também como capacidades ou recursos.

Para Sommerville (2011, p.59) “eles são declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações”.

Machado (2016) por sua vez afirma que “os requisitos funcionais são aqueles que descrevem o comportamento do sistema, suas ações para cada entrada, ou seja, é aquele que descreve as funcionalidades, as quais se espera que o sistema forneça”.

Tratando-se, por exemplo, de um software para uma clínica médica, alguns dos requisitos funcionais poderiam ser: "o usuário médico deve ser capaz de criar e editar informações do prontuário dos pacientes", "O usuário administrador poderá fazer o download dos relatórios referentes às consultas realizadas por dia, mês, semestre e ano", “O sistema deve acessar a webcam integrada ao computador e tirar uma foto do paciente” .

2.1.4 Requisitos Não Funcionais

Os requisitos não funcionais, por outro lado, “são aqueles que agem para restringir a solução. Por vezes, conhecidos como restrições ou requisitos de qualidade” (SWEBOK, 2014, v.3.0, 1-3, tradução nossa), além disso eles têm o papel de apresentar como o sistema se comportará de fato dado um problema específico.

Sommerville (2011, p. 59), por sua vez, afirma que:

> estes são restrições aos serviços ou funções oferecidos pelo sistema. Incluem restrições de timing, restrições no processo de desenvolvimento e restrições impostas pelas normas. Ao contrário das características individuais ou serviços do sistema, os requisitos não funcionais, muitas vezes, aplicam-se ao sistema como um todo. (SOMMERVILLE, 2011, p. 59)

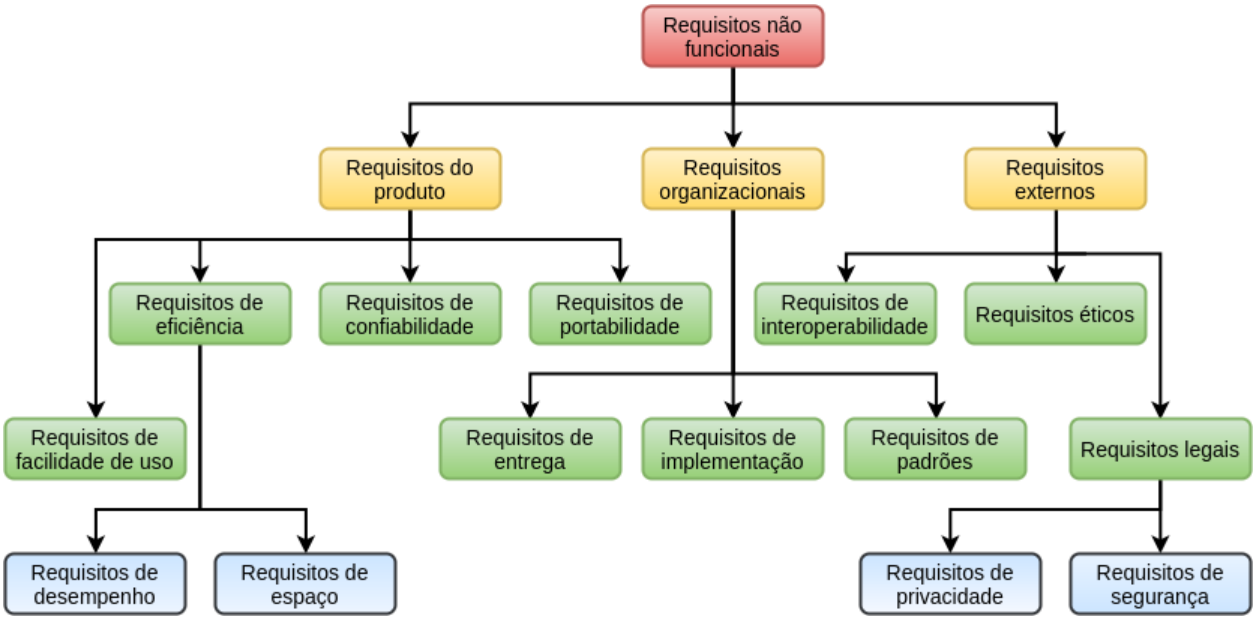
Tais requisitos podem afetar a arquitetura geral de um sistema em vez de apenas componentes individuais. Em razão disso, devem ser considerados de extrema importância na fase de elicitação de requisitos. Sobre isso, Sommerville (2011, p. 60) afirma ainda que,

> Os requisitos não funcionais são frequentemente mais críticos que requisitos funcionais individuais. Os usuários do sistema podem, geralmente, encontrar maneiras de contornar uma função do sistema que realmente não atenda a suas necessidades. No entanto, deixar de atender a um requisito não funcional pode significar a inutilização de todo o sistema. (SOMMERVILLE, 2011, p. 60)

Eles podem ser classificados de acordo com sua categoria: desempenho, manutenibilidade, segurança, confiabilidade, interoperabilidade, entre outros.

Diferente dos requisitos funcionais, quando se desenvolve um novo sistema, podemos ter uma ideia de quais serão os requisitos não funcionais que devemos nos preocupar e consultar o cliente como deseja que seja desenvolvido, pois dentro de áreas de sistemas (saúde, educação, militar, aéreo, etc) os requisitos geralmente são semelhantes. Devido a multiplicidade de requisitos não funcionais, estes são classificados de acordo com Sommerville (2011): requisitos de produto (especificam ou restringem o comportamento do produto), requisitos organizacionais (que traduzem políticas da organização do cliente e do desenvolvedor) e requisitos externos (derivados de fatores externos como leis, regulamentos ou ética), tais como mostrado na figura abaixo, nela os requisitos de produto se desdobram em requisitos de eficiência, facilidade de uso confiabilidade e portabilidade.

Figura 2 – Classificação dos requisitos não funcionais



Fonte:SOMMERVILLE, 2011.

2.1.5 Requisitos Inversos

Embora os esforços da engenharia de requisitos se concentrarem naquilo que o sistema deve fazer, existe uma classe de requisitos cuja preocupação é especificar aquilo que o sistema não deve fazer, os chamados requisitos inversos ou escopo negativo. Eles são importantes, pois declaram junto aos clientes os limites do sistema e evitam com que se gere expectativas irreais sobre ele (VASQUEZ e SIMÕES, 2016).

Esse tipo de requisito geralmente tem a frase “não deve” somado a alguma declaração, exemplos: o sistema não deve realizar pagamento com cartão de débito, o sistema não deve ser acessado pela internet, etc.

O uso desse tipo de requisito na especificação deve ser feito com moderação. Pois, a principal preocupação deve ser entender o escopo do sistema ou o que o sistema deve fazer. Porém há casos em que torna-se necessário ter uma certeza sobre ações que o sistema não deve realizar, geralmente associado a sistemas críticos, nesses casos os requisitos inversos devem estar presentes.

2.2 Engenharia de Software e Documentação de Requisitos

A Engenharia de Software dispõe de uma disciplina chamada Engenharia de Requisitos, a qual propõe através do uso de técnicas, estabelecer, principalmente nas fases iniciais do desenvolvimento, uma comunicação com o usuário para o entendimento daquilo que ele necessita como solução para seu negócio.

Segundo Pressman (2011, p. 126) “Antes de iniciar qualquer trabalho técnico, é uma boa ideia aplicar um conjunto de tarefas de engenharia de requisitos. Estas levam a um entendimento de qual será o impacto do software sobre o negócio, o que o cliente quer e como os usuários finais irão interagir com o software”.

Vasquez e Simões (2016, n.p.), por sua vez, definem a Engenharia de Requisitos como,

> A Engenharia de Requisitos pode ser definida como uma disciplina da Engenharia de Software que consiste no uso sistemático e repetitivo de técnicas para cobrir atividades de obtenção, documentação e manutenção de um conjunto de requisitos para software que atendam aos objetivos de negócio e sejam de qualidade. (VASQUEZ e SIMÕES, 2016, n.p.)

A Engenharia de Requisitos, segundo Pressman (2011), propõe algumas etapas, são elas: Concepção, que define o escopo e o problema; o Levantamento de Requisitos, que busca definir o que é necessário e elaborar e refinar um conjunto de requisitos básicos; a Negociação, para definir prioridades, necessidades e itens opcionais; a Especificação, em que os requisitos são modelados e documentados; a Validação, a qual procura garantir que a especificação esteja correta e sem ambiguidades.

Este trabalho, no entanto, procura explorar a etapa de especificação da Engenharia de Requisitos devido a importância dessa documentação de requisitos que serve de comunicação

entre todos os stakeholders do projeto, esclarecendo as necessidades do cliente e que guiarão os desenvolvedores no projeto e serão a base para a validação do sistema.

Para conseguir o que se espera da documentação de requisitos, esta deve ainda satisfazer um conjunto de características de qualidade. Sobre a qualidade da escrita dos requisitos Filho (2013, p. 168) afirma que

> [..] Um enunciado deve ser: **Correto** - Todo requisito presente realmente é um requisito do produto a ser construído; **Preciso** - Todo requisito presente possui apenas uma única interpretação [...]; **Completo** - Reflete todas as decisões de especificação que foram tomadas; **Consistente** - Não há conflitos entre nenhum dos subconjuntos de requisitos presentes; **Priorizado** - Cada requisito é classificado de acordo com a sua importância, estabilidade e complexidade; **Verificável** - Todos os seus requisitos são verificáveis; **Modificável** - Sua estrutura e estilo permitem a mudança de qualquer requisito, de maneira fácil, completa e consistente; **Rastreável** - Permite a fácil determinação dos antecedentes e das consequências de todos os requisitos.
> (FILHO, 2013, p. 168)

Os requisitos de um projeto podem ser modelados e documentados de diversas maneiras, dependendo da organização, do processo de software e do ciclo de vida do projeto, o que abordaremos mais à frente.

2.3 Técnicas de Modelagem e Escrita de Requisitos

A modelagem de requisitos faz uso de múltiplas técnicas a fim de que a informação contida nos requisitos seja representada de forma visual e descritiva, facilitando a análise e comunicação, para que se possa estabelecer um acordo entre usuários e equipe de desenvolvimento acerca das funcionalidades e características do software.

Pressman (2011, p. 152) afirma que "o modelo de requisitos deve alcançar três objetivos primários: descrever o que o cliente solicita, estabelecer uma base para a criação de um projeto de software e definir um conjunto de requisitos que possa ser validado assim que o software for construído".

De acordo com Sommerville (2011, p. 66) “desde o início da engenharia de software a linguagem natural tem sido usada para escrever os requisitos para o software. É expressiva, intuitiva e universal. Também é potencialmente vaga, ambígua e seu significado depende do conhecimento do leitor”. Todavia em meio a todos esses empecilhos, a linguagem natural é o meio mais utilizado até hoje de especificação de requisitos.

Sommerville(2011) também aponta outras formas de especificar requisitos, tais como a linguagem natural estruturada em que os requisitos são escritos em linguagem natural mas

dentro de formulários padrão ou *templates*, notações gráficas no qual são utilizados modelos gráficos auxiliados por descrições textuais, tais como os diagramas de caso de uso. Existem também as especificações formais ou matemáticas que são baseadas em modelos matemáticos, máquinas de estado ou conjuntos, estas são normalmente utilizadas em sistemas críticos em razão de diminuírem a ambiguidade, porém pode ser de difícil entendimento para os clientes, o que dificulta a validação por parte destes.

A tarefa de modelagem não é algo simples, podendo um conjunto de requisitos serem analisados de diferentes perspectivas e detalhados conforme as necessidades do processo de software e do próprio sistema. Segundo Vasquez e Simões (2016) "requisitos isolados não costumam ser complexos; são o relacionamento e a interdependência entre esses requisitos isolados que levam à complexidade".

2.3.1 Templates

Os *templates* ou formulários padrão são meios muito utilizados para especificação de requisitos. Eles propõem que estes sejam escritos em linguagem natural de forma sistemática, a partir do preenchimento de campos de um formulário cuidadosamente elaborado para que um conjunto de informações importantes seja evidenciado, tornando a comunicação em uma forma padrão, para que tanto a equipe de desenvolvimento como os usuários possam compreender.

No mapeamento sistemático de literatura realizado por Benitti e Silva (2011) foram identificados três trabalhos sobre padrões de escrita de requisitos, os quais fizeram uso de *templates* para a especificação e documentação. Estes trabalhos foram os artigos de Toro *et al*. (1999), Withall (2007) e Franch *et al.* (2010). Neles são propostos *templates* e padrões de linguagem, como também padrões de requisitos, baseados nas experiências em desenvolvimento de sistemas dos autores, os quais identificaram qual a linguagem e geralmente quais as informações estão associadas a determinados requisitos, transformando esse conhecimento em um formulário genérico e pré-preenchido no qual engenheiros de requisitos pudessem utilizar e adaptar ao seus projetos.

O padrão IEEE Std. 830 (1998) recomenda um conjunto de práticas para a escrita de uma boa especificação de software, como também sugere um *template* e um conjunto de sugestões para modelar o item “requisitos específicos”, organizando-os por: modo, classes de usuário, objetos, recursos, estímulos, hierarquia de funções ou em múltiplas organizações.

Na tabela abaixo, vemos um exemplo de formulário padrão para requisitos funcionais de software, em que é possível inserir nome, descrição, pré-condição, pós-condição e sua sequência de passos.

Tabela 1 – Exemplo de *Template*

RF		
Descrição		
...	...	
Pré-condição		
Sequência	Passo	Ação
	1	...
	2	...
Pós-condição		
...	...	

Fonte: Elaborado pelo autor (2020)

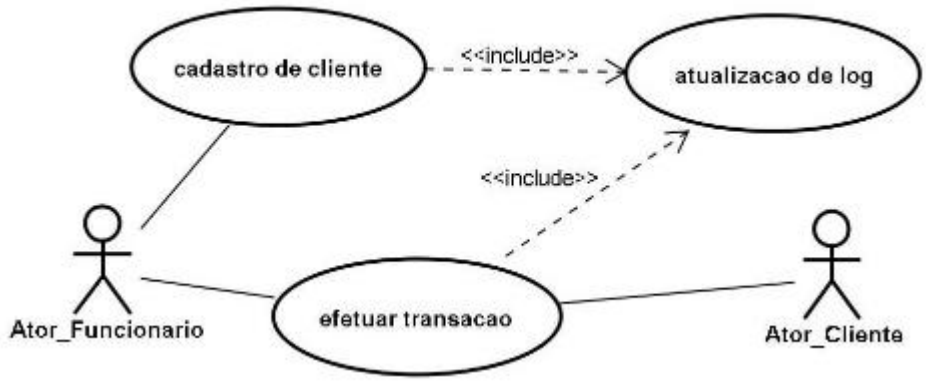
2.3.2 Casos de uso

Originalmente desenvolvido por Jacobson *et al* (1992) e depois incorporado ao primeiro release da UML (*Unified Modeling Language*), um caso de uso de acordo com Vasquez e Simões (2016, n.p.) é "um conjunto de passos que descreve um cenário principal e possíveis cenários alternativos para um ator alcançar um objetivo com o uso do sistema".

Em geral, os casos de uso são descritos em linguagem natural, de forma narrativa e informal ou podem ser reescritos formalmente utilizando um *template* estruturado caso o software a ser desenvolvido tenha muitos detalhes ou seja crítico, nesses casos deve-se considerar definir o objetivo do caso de uso, as precondições, o disparador ou evento que o inicia, a descrição do cenário e as exceções, e outras informações que se julgarem importantes.

Casos de uso podem ser modelados de forma gráfica por diagramas UML, em que as interações do sistema são representados por atores (sejam eles pessoas, outros sistemas ou hardware) na forma de bonecos palito que são ligados através de linhas a elipses, as quais representam as funcionalidades ou interações do sistema. A esse diagrama de alto nível é geralmente adicionada informação adicional através de descrição textual ou outros diagramas tais como o diagrama de sequência da UML.

Figura 3 – Exemplo de diagrama Caso de Uso



Fonte: Robinson, 2014.

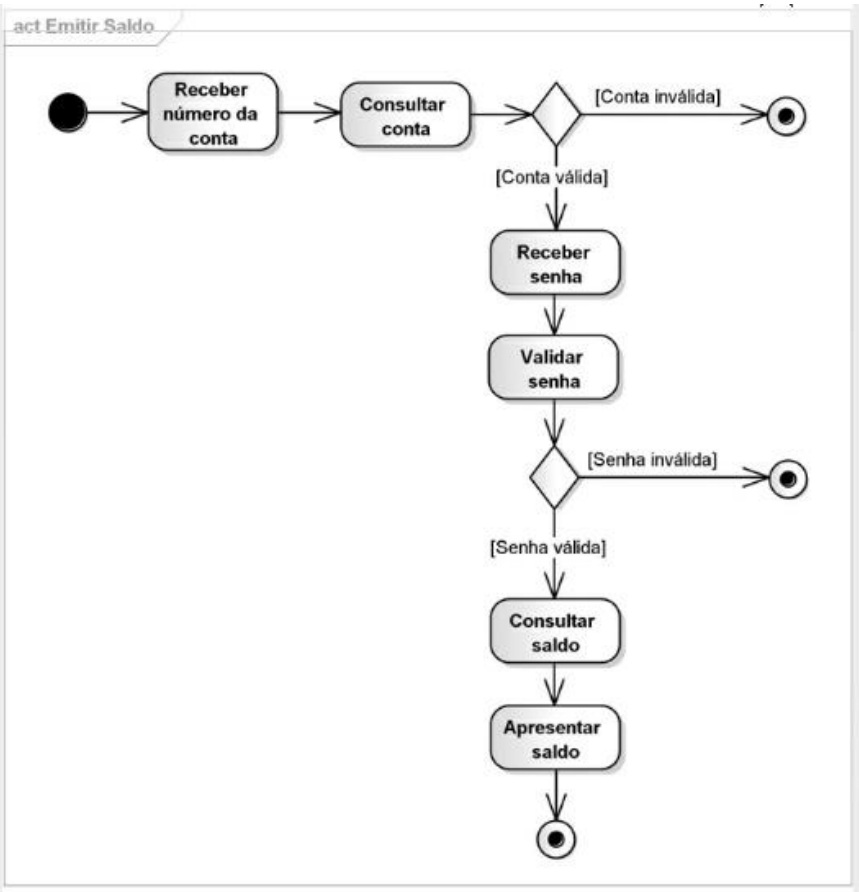
Vasquez e Simões (2016) afirmam que com a informação organizada em um modelo funcional baseado em casos de uso, é possível trabalhar em três áreas muito importantes nos projetos: Definição de requisitos, Comunicação com os clientes e Geração de casos de teste. Além destas indicadas pelos autores, os casos de uso podem ser usados para diversas atividades de planejamento de projeto, como calcular esforço estimado para desenvolvimento do software, para construção de cronograma e por consequência, seu custo estimado, entre outras.

2.3.3 Diagramas de atividade

São representações gráficas que pretendem mostrar as atividades que compõem um processo de sistema, como também o fluxo de controle em um cenário. São semelhantes a fluxogramas e incrementam os casos de uso adicionando detalhes que às vezes ficam implícitos.

O início de um processo é determinado por um círculo preenchido, as atividades por retângulos arredondados, as setas representam o fluxo de uma atividade a outra e o fim do processo é dado por um círculo preenchido dentro de outro círculo.

Figura 4 – Exemplo de diagrama de atividade



Fonte: GUEDES, 2018

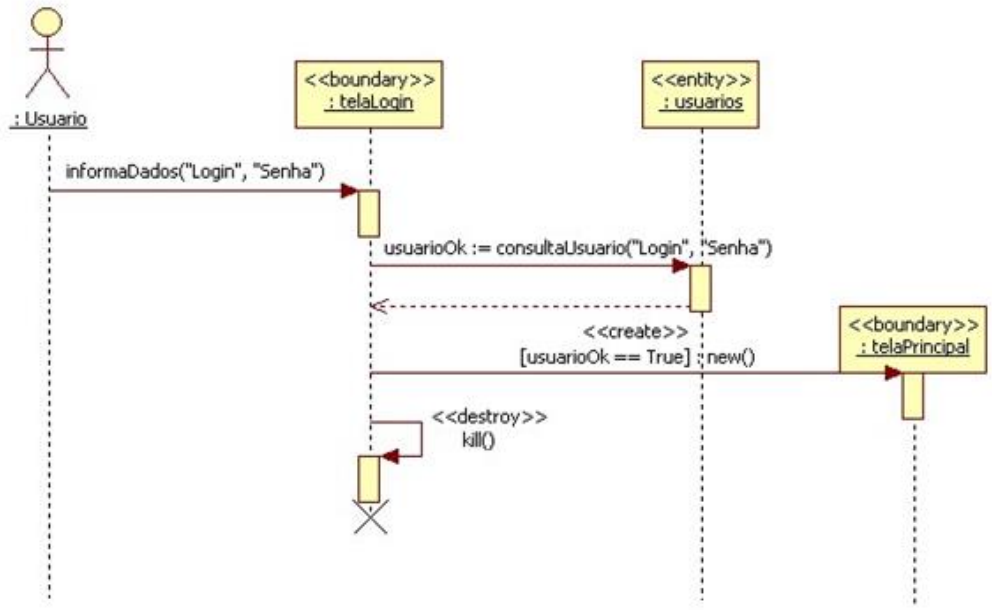
2.3.4 Diagrama de sequência

São diagramas comportamentais usados para demonstrar as interações entre atores e objetos e as interações entre os próprios objetos, preocupando-se com a ordem temporal em que as mensagens são trocadas entre os objetos.

Geralmente eles modelam casos de uso específicos, mostrando os atores na parte superior com uma linha pontilhada verticalmente a qual se coloca um retângulo para indicar a linha de vida do objeto, as interações são indicadas por setas anotadas com informações sobre chamadas dos objetos, seus parâmetros e valores de retorno.

Embora seja uma notação utilizada na fase de projeto, diagramas de sequência são usados para representar cenários de maneira gráfica e não ambígua. Também podem ser úteis para validar casos de uso, numa fase posterior ao processo de desenvolvimento do software.

Figura 5 – Exemplo de diagrama de sequência



Fonte: http://www.theclub.com.br/restrito/revistas/201308/umld1308.aspx

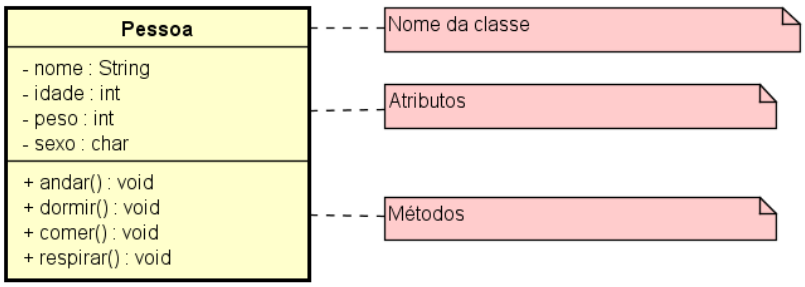
2.3.5 Modelagem baseada em classes

Trata-se de um modelo estrutural para exibir a organização de um sistema através de seus componentes e relacionamentos. Ele representa os objetos, os métodos e serviços aplicados a esses objetos e os relacionamentos e colaborações entre as classes.

Para tanto deve-se analisar os cenários de interação provenientes da elicitação de requisitos junto aos stakeholders e verificar quais elementos do espaço do problema deverão ser considerados como classes, as quais terão um conjunto de atributos, e determinar como se dará o relacionamento entre elas. Para o efeito de análise e especificação de requisitos ainda não se faz necessário determinar as operações de cada classe, pois fazem parte de decisões de projeto.

Na UML uma classe é representada como um retângulo com três divisões, em que a primeira parte se destina ao nome da classe, a segunda para os seus atributos e a terceira para os métodos conforme mostrado no exemplo abaixo. Além disso, os relacionamentos entre as classes podem ter um significado diferente dependendo da linha/seta que os une, mostrando relacionamentos de associação, agregação, composição ou generalização/especialização.

Figura 6 – Exemplo de representação de Classe



Fonte: https://medium.com/gdgcampinas/conceitos-b%C3%A1sicos-de-orienta%C3%A7%C3%A3o-a-objetos-b58809b2d809

2.3.6 Histórias do usuário

Utilizada inicialmente no XP e posteriormente nos demais tipos de desenvolvimento ágil de software. É um tipo de especificação de requisitos simples, livre de detalhes, geralmente produzidas como declarações breves dos usuários sobre suas necessidades em relação ao produto de software. Em razão disso, ambientes burocráticos ou onde níveis de documentação são exigidos pode-se preferir outro método de documentação (VASQUEZ e SIMÕES, 2016).

A redação de uma história de usuário é livre, todavia deve-se considerar colocar quais são as partes interessadas, o valor para o negócio e uma visão de alto nível da funcionalidade requerida.

A partir delas, dentro do desenvolvimento ágil, é criado o *product backlog*, que é o artefato onde estarão todos os itens que a equipe de desenvolvimento deve construir. As histórias do usuário passam a ficar no nível de objetivo do usuário, e podem ser posteriormente ser subdivididas, para servirem como base para decisões de projeto e serem parte dos ciclos de interação.

As histórias do usuário geralmente são observadas como itens de base para requisitos funcionais. Nesse sentido, Vasquez e Simões (2016, n.p.) alertam que “as histórias dos usuários não abordam explicitamente como documentar requisitos não funcionais”.

Alguns critérios de qualidade são definidos pelo acrônimo INVEST (Independente, Negociável, Valioso, Estimável, Pequeno e Testável) e o CCC (Cartão, Confirmação e Conversação).

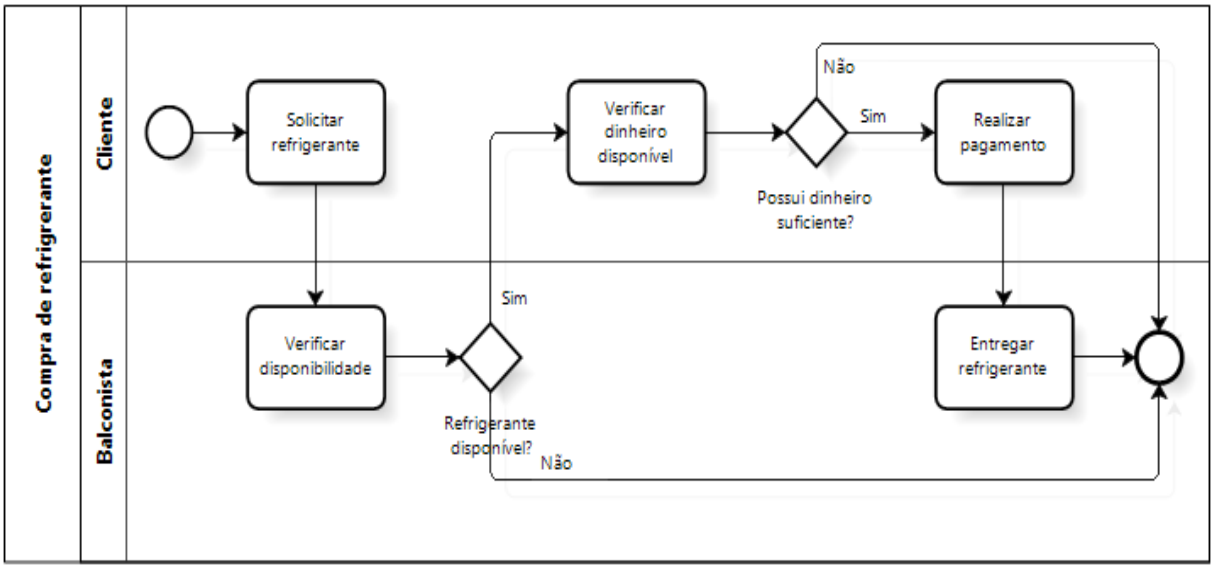
2.3.7 BPMN

BPMN (*Business Process Modelling Notation*) é uma notação muito utilizada para representar a modelagem de processos de negócio. Ela define um modelo gráfico, diagrama

BPD (*Business Process Diagram*) baseado em técnicas de fluxograma de forma que seja entendido por todos os envolvidos no processo de definição da modelagem do negócio e ao mesmo tempo consiga lidar com as complexidades envolvidas no processo.

Faz uso de elementos familiares como retângulos que representam atividades e losangos para representar decisões, como mostrado na figura a seguir.

Figura 7 – Exemplo de modelo BPMN



Fonte: http://blog.iprocess.com.br/2012/11/um-guia-para-iniciar-estudos-em-bpmn-i-atividades-e-sequencia/

2.3.8 Abordagem para WebApps

Pressman (2011) propõe um conjunto de classes para representar as principais características de um WebApp, são elas: modelo de conteúdo, modelo de interações, modelo funcional, modelo de navegação e modelo de configuração.

O modelo de conteúdo representa os requisitos de conteúdo, os quais englobam os objetos de conteúdo (com suas descrições, seus relacionamentos e/ou hierarquia) e as classes de análise, normalmente representados por uma simples lista, mas que pode ser beneficiar de elementos gráficos como uma árvore de dados.

O modelo de interação é composto por um ou mais dos elementos: casos de uso, diagramas de sequência, diagrama de estados e/ou protótipos de interface do usuário. Ele pretende mostrar como se dará o diálogo entre o usuário e as funcionalidades, o conteúdo e o comportamento da aplicação. Na maioria dos projetos um conjunto de casos de uso é suficiente, mas se o projeto for complexo uma representação mais rigorosa deve ser aplicada.

O modelo funcional se preocupa com as funcionalidades observadas pelo usuário no WebApp e as operações contidas nas classes de análise que implementam comportamentos associados à classe. O modelo de requisitos, então, passa a descrever o processamento a ser realizado pelas operações das classes de análise, nesse sentido pode-se utilizar o diagrama de atividades da UML para esta representação.

O modelo de configuração em geral é uma lista de atributos no servidor e no cliente que pode agregar ainda alguns detalhes que poderiam ter impacto no projeto (como distribuição de carga entre vários servidores). Para projetos complexos o diagrama de disponibilização da UML deve ser utilizado.

Por fim, o modelo de navegação se preocupa com os requisitos de navegação, os quais procuram determinar como se dará a navegação de cada categoria de usuário de um elemento a outro dentro do WebApp e responder questões tais como a existência ou não de mapas de navegação, se o projeto deverá ser dirigidos por comportamentos do usuário ou pela importância percebida pelos elementos, entre outros.

2.4. Risco

Algo importante que deve ser considerado na construção de qualquer projeto são os riscos que estão associados a ele. É interessante que desde a análise dos requisitos do sistema eles sejam levantados e ao longo do processo de entendimento do problema serem refinados, contribuindo em sua gestão e contingência.

Para Pressman (2011, p. 648) “O risco é um problema potencial (...). Independentemente do resultado, é aconselhável identificá-lo, avaliar sua probabilidade de ocorrência, seu impacto e estabelecer um plano de contingência caso o problema realmente ocorra”. Duas características essenciais na análise do risco, como frisado nessa definição, são a probabilidade (ou a incerteza) e o impacto (ou perda, consequências indesejadas) que são utilizados como parâmetros para categorizar os riscos.

Sobre os riscos de software, a força aérea americana publicou algumas diretrizes que determinam quatro tipos de componentes de risco de software: de desempenho (adequação aos riscos e adequação ao uso); de custo (manutenção do orçamento); de suporte (relacionado a facilidade de correção, adaptação e melhoria); de cronograma (manutenção do cronograma). E o impacto é categorizado em uma escala de: Catastrófico, Crítico, Marginal e Negligenciável. (PRESSMAN, 2011, apud American Air Force, 1988). Na tabela abaixo fizemos uma adaptação da elaborada por Pressman (2011, p. 652) para atender os objetivos deste estudo.

Tabela 2 – Componentes e Escala de Risco

Categoria		Componentes	
		Desempenho	**Suporte**
Catastrófico	**1**	Falha em satisfazer o requisito resultaria em inutilização do software	
	2	Degradação significativa até não cumprimento do desempenho técnico	Software que não responde com agilidade ou que é difícil de dar suporte
Crítico	**1**	Falha em satisfazer o requisito degradará o desempenho do sistema até um ponto no qual a utilização do software é questionável	
	2	Alguma redução no desempenho técnico	Pequenos atrasos nas modificações de software
Marginal	**1**	Falha em atender o requisito resultaria na degradação de funções secundárias do sistema	
	2	De mínima a pequena redução no desempenho técnico	Suporte responsivo de software
Negligenciável	**1**	Falha em atingir o requisito criaria inconveniência ou impacto não operacional	
	2	Nenhuma redução no desempenho técnico	Software de fácil manutenção

Legenda: [1] Potencial consequência de erros ou falhas de software não detectadas.
[2] Potencial consequência se o resultado esperado não é obtido.
Fonte: Adaptada de Pressman (2011, p. 652)

Um outro método muito utilizado para avaliação de risco são as chamadas matrizes de risco, as quais listam todos os possíveis riscos ao que o software pode estar sujeito e os associam a sua probabilidade e impacto. Leopoldino (2004, p. 45) afirma que a matriz de risco "permite o levantamento das variáveis de risco de forma subjetiva, sua mensuração, priorização e visualização de forma singular".

Garvey e Lansdowne (1998) apresentam uma matriz de risco que tem como foco o par requisito-tecnologia como fundamento para identificar a existência de riscos para um programa, a matriz tem sete colunas com os itens: Requisito; Tecnologia; Risco; Impacto (I) que pode ser categorizado como Crítico (C), Série (S), Moderado (Mo), Menor (Mi) e Mínimo (N); Probabilidade de Ocorrência (P%) que pode ser classificada como Muito Baixa (0-10%), Baixa (11-40%), Moderada (41-60%), Alta (61-90%), Muito Alta (91-100%); Nível de risco (R) que pode apresentar os valores Alto (H), Médio (M) ou Baixo (L); Gerenciamento/Mitigação, como mostrado na tabela abaixo.

Tabela 3 – Matriz de Risco

Requerimento	Tecnologia	Risco	I	P	R	Gerenciamento/mitigação
Comunicação em um raio de 100 milhas	ARC-210	-Performance da antena	S	61-90	Médio	Definir a performance como parâmetro chave do programa de testes.
Compatibilidade com A-10, F-16, JSTARS e ABCCC	Tecnologia não disponível atualmente	-Taxas de suprimento de energia incorretas - Conectores errados	Mi	0-10	Baixo	Vistoria nos aviões durante a reunião em terra
Controle do rádio localizado na cabeça do piloto	-	- Dificuldade em obter consenso entre os pilotos	Mi	91-100	Alto	Apresentações logo no início do projeto
Agenda: entrega em 24 meses	-	- Tempo de entrega do circuito integrado	S	11-40	Médio	Incentivar a entrega em tempo

Fonte:(Leopoldino, 2011, p.46 apud Garvey e Lansdowne, 1998, p.18-22).

A partir de matrizes de risco é possível a elaboração de matrizes de probabilidade e impacto na forma de *cross-tabs* onde são definidas escalas de frequência do fator e de consequências danosas caso ocorram, que resultam em uma melhor análise dos riscos para definir a priorização das ações sobre os eles.

2.5 Trabalhos Correlatos

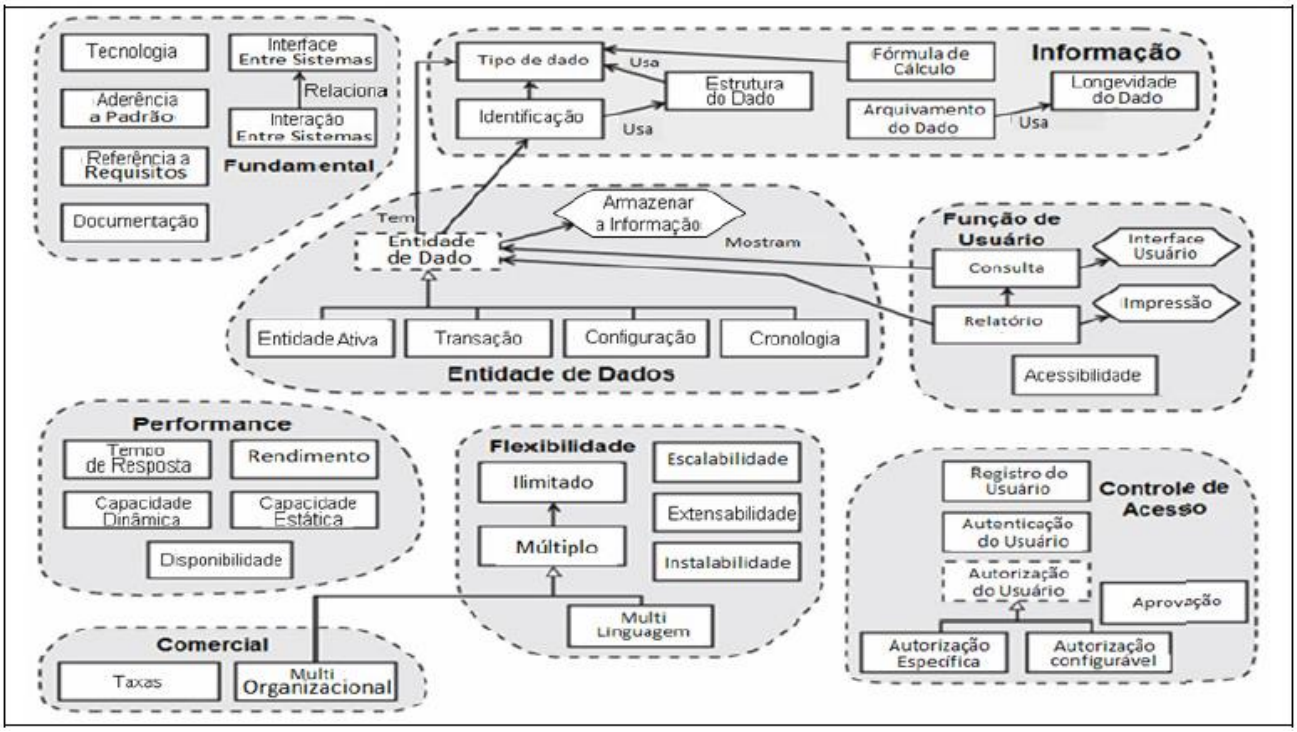
A seguir, mostra-se alguns modelos utilizados por outros pesquisadores na modelagem de requisitos, são eles: a abordagem de Withall, Mapas Conceituais e Árvores Características.

2.5.1 Abordagem de Withall

No trabalho de Withall (2007), a partir de um estudo em projetos reais, ele conseguiu destacar oito tipos de domínios, os quais agregam 37 padrões de requisitos que poderão agregar ou compartilhar informação entre eles. Esses padrões, então, seriam formas de orientar a escrita dos requisitos na sua especificação melhorando sua qualidade e tornando-a mais simples e rápida, ao sugerir informações, sugestões e alertas sobre os requisitos. Segundo o autor, seu catálogo pode ser adaptado a quaisquer sistemas ou servir de ponto de partida de acordo com o tipo de usuário e problema. A utilização do catálogo é descrita no website do autor através de uma ferramenta a qual faz uso dos padrões em um formato de *template*, no qual o usuário recebe

uma sugestão de redação do requisito de acordo com o padrão desejado, . O catálogo de withall (2007) é mostrado a seguir.

Figura 8 – Catálogo de Withall



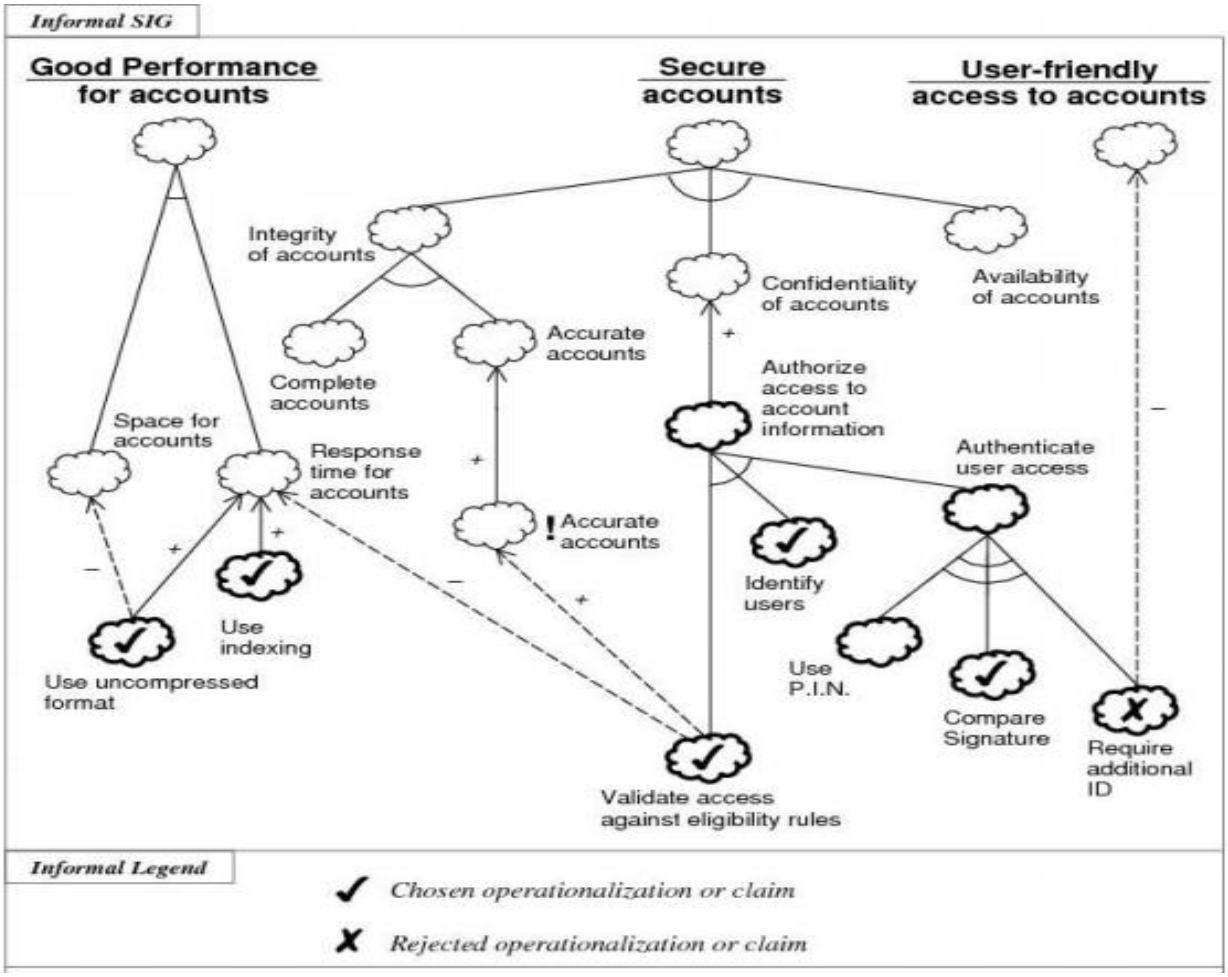
Fonte: Benitti e Silva, 2011.

2.5.2 NFR Framework

É uma abordagem orientada a metas, focada nos requisitos não funcionais. Consiste num processo em que, inicialmente, são definidos os requisitos gerais e de maior interesse para o sistema e depois estes são refinados até satisfazer o requisito geral. Para tanto, oferece uma estrutura em grafos, chamados *Softgoal Interdependency Graph* (SIG), como também um catálogo de conhecimento que inclui técnicas de desenvolvimento.

Há dois elementos bastante importantes no SIG que são os *softgoals* que representam os requisitos não funcionais e são desenhados em um formato de nuvem, e as operacionalizações que são as interações entre os elementos do grafo, estas podem ter um forte impacto no projeto determinando influências positivas e negativas, como também mostrar interdependências implícitas e explícitas entre *softgoals*. Um exemplo de SIG é mostrado na figura abaixo.

Figura 9 – Exemplo de utilização NFR Framework



Fonte: Xavier, 2009.

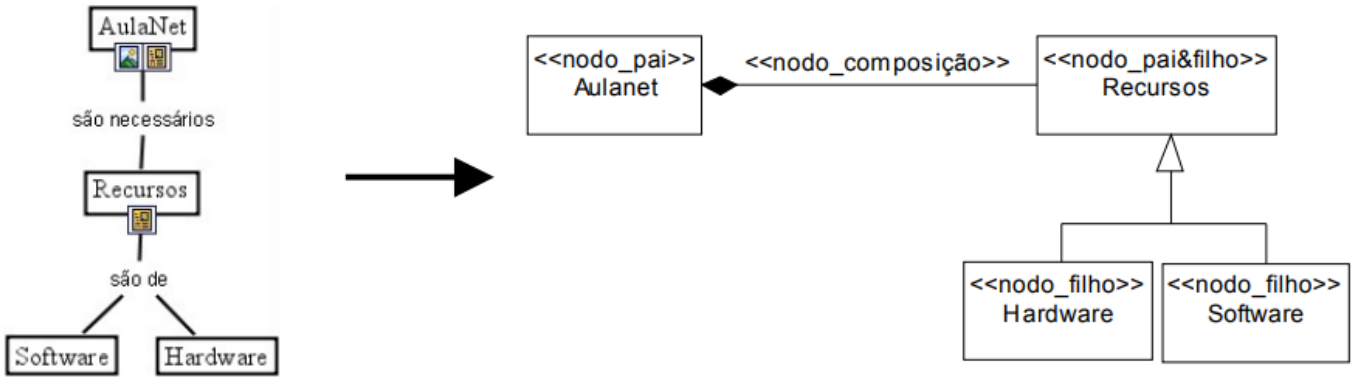
2.5.3 Mapas Conceituais

Os mapas conceituais são uma ferramenta amplamente utilizada para organizar e representar conhecimentos, em razão dessa popularidade e o fácil entendimento, eles foram trazidos para a área de engenharia de requisitos.

Robinson *et al* (2004) propuseram uma abordagem utilizando mapas conceituais através da linguagem UML. Para tanto, foi utilizado mecanismos de extensão da UML (*stereotypes*, *tagged values* e *constraints*) em conjunto com o diagrama de classes, formando a UML-MC, procurando agregar conceitos principais, intermediários ou o mais específicos, como também a hierarquia entre conceitos, que são próprios de um mapa conceitual.

A figura abaixo mostra um exemplo da utilização da UML-MC para uma parte do mapa conceitual AulaNet.

Figura 10 – Exemplo de utilização da UML-MC



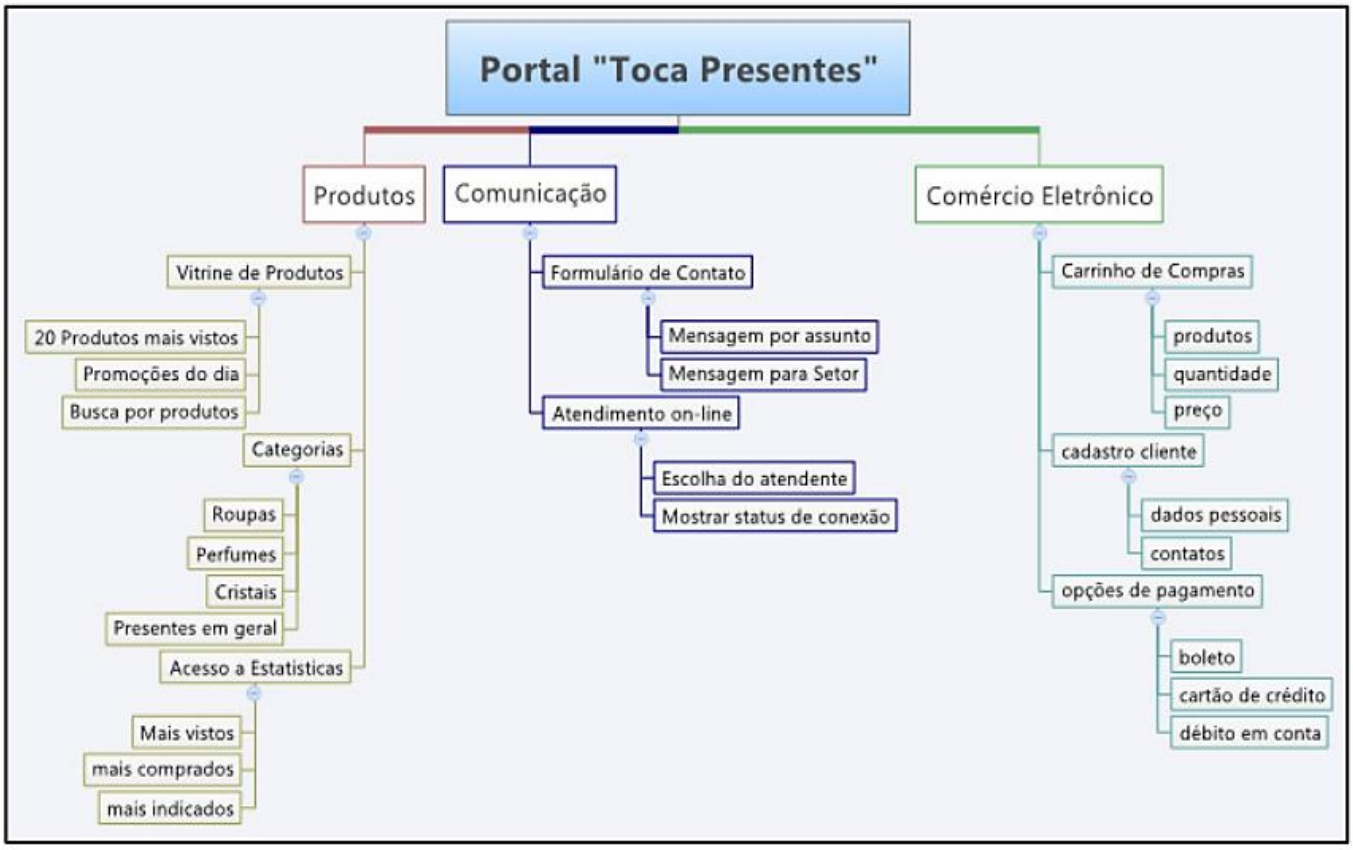
Fonte: Robinson et al., 2004.

2.5.4 Árvore de Características

Trata-se de um modelo gráfico, no qual as características ou requisitos, expressos em linguagem natural, e suas interdependências são dispostos em um formato de árvore. Para sua construção, define-se o objetivo ou sistema e delineia-se as suas principais características, dessas características, então, são criadas outras características importantes para cada item formando folhas na árvore. As ligações entre as características podem ainda ser definidas como obrigatórias, alternativas, opcionais e dependentes de acordo com a linha que se coloca entre elas. O tamanho da árvore, portanto, é dependente do nível de detalhamento em que se desdobram as características.

Por sua facilidade para comunicação com o usuário é uma ferramenta que é bem aceita na atividade de elicitação de requisitos como mostrado por Oliveira e colaboradores (2010).

Figura 11 – Exemplo de Árvore de Características



Fonte: Oliveira et al, 2010.

CAPÍTULO 3 - PROPOSTA

A partir da análise dos métodos de modelagem e especificação de requisitos, constatou-se que os métodos até então vistos separam os requisitos funcionais dos requisitos não funcionais, muitas vezes dedicando-se a apenas um deles, não mostrando a relação entre ambos, o que dificulta o entendimento do problema e das necessidades do cliente pelo engenheiro de software (Sommerville, 2019).

Os requisitos quando vistos em conjunto proporcionam uma visão geral do sistema, revelando as relações entre eles e como estas se somam ou se contrariam. Sem essa relação, gasta-se mais tempo para analisar a documentação, como também dificulta a validação dos requisitos pelo cliente, haja vista não se ter uma visão geral daquilo que foi comunicado e daquilo que foi entendido.

Portanto, neste trabalho propomos um diagrama e sua documentação de software, a partir da extensão do diagrama de Casos de Uso da UML, que foi escolhido por ser flexível e de fácil entendimento, para agregar a partir do levantamento de requisitos o conjunto de requisitos funcionais e não funcionais e a relação entre eles, concentrando atenção aos requisitos de produto conforme exemplificado no item 2.1 deste trabalho.

Além desta importante característica, consideramos igualmente importante adicionarmos ao modelo um elemento que trouxesse o conceito de risco para ajudar a justificar decisões de projeto, como também contribuir para a rastreabilidade dos requisitos.

A partir disso, desenvolvemos uma nova abordagem para que fosse possível ao cliente e a equipe de desenvolvimento identificar o conjunto de funcionalidades desejadas em conjunto com a forma em que ele deseja que elas operem, e nesse momento também verificar os riscos envolvidos com a não implementação dos requisitos propostos ou sua implementação parcial. Como também uma especificação textual associada que revele de forma mais aprofundada as características do diagrama.

Nesse contexto, descrevemos a seguir a proposta do RNFCase, fazendo uma introdução com os elementos nativos do diagrama Casos de Uso e dos elementos que promovem extensibilidade na UML, como também mostramos os elementos que adicionamos ao diagrama Casos de Uso para atingirmos os objetivos deste trabalho e o modelo de documentação que deve estar associada ao diagrama.

3.1 Elementos do diagrama Caso de Uso UML

Retomando o que foi brevemente explicado no item 2.3.2 deste trabalho, apresentaremos os elementos nativos do diagrama de Casos de Uso da UML.

3.1.1 Atores

Um dos itens essenciais do diagrama são os bonecos palito, que representam os atores do cenário que irão interagir com os casos de uso, estes podem ser qualquer elemento externo que interaja com o software, como pessoas, outros sistemas ou até mesmos hardwares. Logo abaixo ao seu desenho deve-se colocar sua descrição que apontará o papel que ele irá assumir no diagrama.

Figura 12 –Exemplo de Ator

3.1.2 Casos de Uso

Um segundo elemento essencial ao diagrama são os casos de uso, eles representam os serviços, as funcionalidades ou as tarefas do sistema as quais os atores irão interagir, mostrando os comportamentos pretendidos de acordo com os requisitos identificados. Seu desenho gráfico é uma elipse com uma breve descrição da funcionalidade em seu interior.

Figura 13 – Exemplo de Caso de Uso

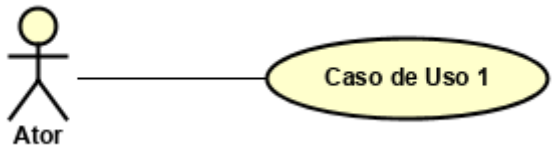
3.1.3 Relações

Dentro da UML também estão definidas algumas relações possíveis entre os elementos, são estes: a associação, a generalização/especialização, a inclusão e a extensão que irão demonstrar os comportamentos do conjunto de funcionalidades do software.

3.1.3.1 Associação

A relação de associação representada por uma linha, representa uma interação entre os elementos do diagrama. Esta pode conter em sua extremidade uma seta que indicará o sentido em que as informações trafegam (se não há setas a informação trafega nos dois sentidos), como também indicar quem inicia a comunicação.

Figura 14 – Exemplo de Associação

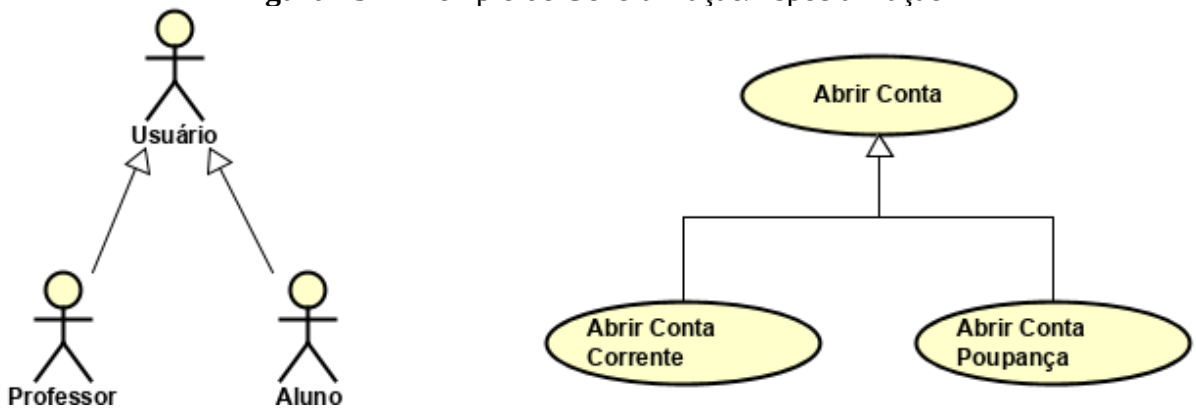


Fonte: Elaborada pelo autor, 2019.

3.1.3.2 Generalização/Especialização

É um tipo de associação em que tem-se elementos que tem muitas características em comum e poucas que os diferenciam. Nesse caso, cria-se um elemento geral que agregue essas características em comum e depois relacioná-lo a esses elementos que irão conter apenas suas características específicas e herdar as características do elemento geral. Esse tipo de associação é realizada entre atores ou entre casos de uso e é representada por uma linha que sai de uma elemento especializado e que atinge o elemento mais geral a partir de uma seta mais grossa, como mostrado no exemplo abaixo.

Figura 15 – Exemplo de Generalização/Especialização

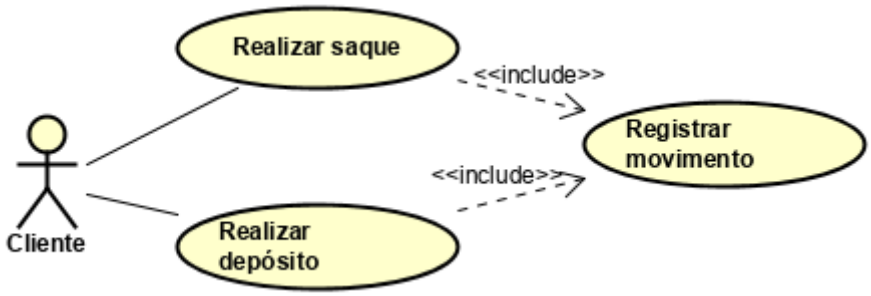


Fonte: Elaborada pelo autor, 2019.

3.1.3.3 Inclusão

A inclusão é um tipo especial de associação em que temos um cenário (semelhante a uma subrotina) que é utilizado obrigatoriamente por um ou mais casos de uso. Esse relacionamento é representado por uma linha tracejada com uma seta que aponta para o caso de uso incluído e com o estereótipo com o texto "*include*" entre os sinais de menor (<) e maior (>).

Figura 16 – Exemplo de Inclusão

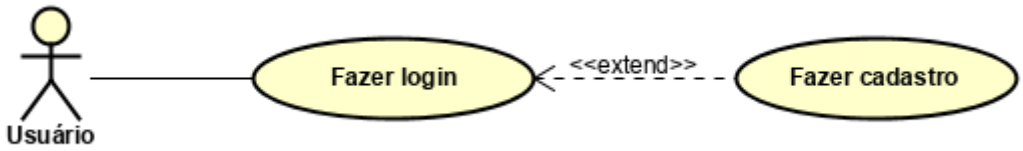


Fonte: Elaborada pelo autor, 2019.

3.1.3.4 Extensão

A extensão é um tipo de associação que diferente da inclusão utilizamos para indicar cenários que são chamados por outro de forma opcional, caso ocorra alguma situação específica. É representada por uma linha tracejada com uma seta que aponta para o caso de uso que utiliza o caso de uso estendido e contém o estereótipo com o texto "*extend*" entre os sinais de menor (<) e maior (>).

Figura 17 – Exemplo de Extensão



Fonte: Elaborada pelo autor, 2019.

3.2 Elementos que promovem Extensibilidade

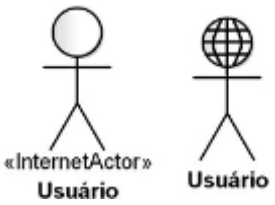
A UML provê alguns mecanismos a fim de adaptar a linguagem para propósitos as quais não projetada ou pensada originalmente. Entre esses mecanismos estão os estereótipos, as *tags* e as restrições.

Os estereótipos são elementos que permitem certo grau de extensibilidade aos demais elementos (componentes e associações) da linguagem UML a fim de lhes conferir novas características (GUEDES, 2018). Eles são amplamente utilizados nos diagramas UML e podem ser categorizados em dois tipos: estereótipos de texto e estereótipos gráficos.

Os estereótipos de texto apresentam apenas um texto entre sinais de maior e menor sobre o componente acima da descrição do seu nome. Por outro lado, estereótipos gráficos modificam o desenho padrão do componente.

A figura 18 mostra um exemplo de estereótipos, nela podemos perceber um elemento Ator chamado de Usuário com o estereótipo de texto "*InternetActor*" para representar um ator que acessa o sistema apenas pela internet, como também um estereótipo gráfico para representar a mesma ideia, mas representado agora como uma figura diferente da representação original, mas associada ao estereótipo "*InternetActor*".

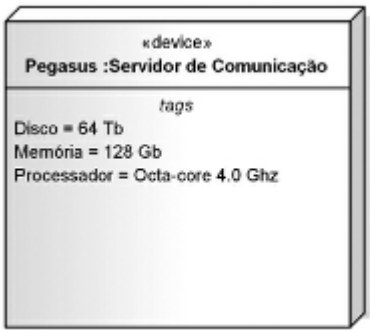
Figura 18 – Exemplo de Estereótipo



Fonte: Guedes, 2018.

As tags ou etiquetas são também uma forma de estabelecer propriedades extras para elementos da UML. Nos diagramas de implantação, por exemplo, podemos ter elementos como nós representando itens de hardware, aos quais desejamos adicionar informações de configuração, então podemos fazer isso por meio de tags, as quais definimos um atributo e informação referente a ele. No exemplo a seguir temos um nó representando um elemento de hardware (um servidor de comunicação chamado Pegasus) com configuração definida pelas tags disco, memória e processador.

Figura 19 – Exemplo de Tag/Etiqueta



Fonte: Guedes, 2018.

As restrições são informações como condições, validações entre outras, que devem ser aplicadas a um determinado componente ou situação na forma de um texto entre chaves (GUEDES, 2018). As restrições geralmente são apresentadas dentro de notas explicativas, no formato de um retângulo com a ponta superior direita dobrada, ligada ao componente através de uma seta tracejada chamada âncora, mas não é algo obrigatório, porém é utilizado por fins de organização. No exemplo abaixo vemos uma restrição sobre uma associação de extensão que informa e limita a associação ao outro caso de uso.

Figura 20 – Exemplo de Restrição

Fonte: Guedes,2018.

3.3 Elementos adicionados ao diagrama UML

Para adicionarmos elementos ao diagrama, utilizamos do recurso de extensibilidade de estereótipo da UML, haja vista que com eles podemos atribuir outras características aos elementos de acordo com a intenção do nosso modelo.

3.3.1 Elemento RNF

Nesse sentido, baseado na necessidade do diagrama agregar informações não apenas do requisitos funcionais, mas também dos requisitos não funcionais adicionamos um elemento em formato de pacote preenchido na cor verde para facilitar a identificação com o estereótipo RNF (abreviação de requisitos não funcionais) entre os sinais duplos de menor (<) e maior (>).

Figura 21 – Exemplo de elemento RNF

Esse elemento tem o papel de descrever em seu interior o requisito não funcional associado ao caso de uso, no formato "(RNF01) Descrição do Requisito", o qual deve conter um número de identificação e um texto curto que destaque as principais informações referentes ao requisito.

3.3.2 Elemento Risco

Para incorporarmos um elemento em que pudéssemos incluir o risco ao diagrama, acrescentamos um elemento em formato de pacote preenchido na cor vermelha, o qual deverá conter uma descrição curta do risco envolvido em um caso de uso ou em um elemento não funcional que esteja no diagrama.

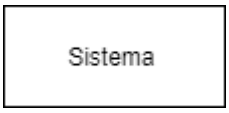
Para a descrição do risco, adaptamos o formato CTC (Condição-Transição-Consequência) tal como apresentado por Pressman (2011) cujo texto é do tipo: *"Considerando (condição) há uma preocupação que (consequência)",* para o nosso propósito a fim de manter o texto curto e a clareza tanto do diagrama como do texto. Logo, na descrição do risco utilizaremos o formato "(R01) nome do risco: explicação breve", no qual a letra "R" identifica que se trata de um risco e o número sua identificação.

Figura 22 – Exemplo de elemento Risco

3.3.3 Sistemas

No RNFCase, também utilizamos um elemento no formato de objeto para representar os sistemas. Ele consiste em um retângulo na cor branca com a denominação do sistema em seu interior, como mostrado na figura abaixo, e é utilizado para demonstrar a relação entre o sistema e os seus requisitos não funcionais.

Figura 23 – Exemplo de representação de sistemas no RNFCase



3.3.4 Associações entre elementos

Para visibilidade dos relacionamentos entre os elementos do diagrama, principalmente no que concerne aos novos elementos, definimos o formato das associações entre eles, as quais seguem a seguir.

3.3.4.1 Associação simples

Como na UML, a associação é dada por uma linha entre os casos de uso, utilizando-se uma seta sempre que se desejar esclarecer o fluxo da comunicação entre os elementos. Todavia, determinamos para fins de visualização que na associação entre os casos de uso com os elementos RNF e Risco ou entre estes, deve-se sempre utilizar uma seta apontando para direção do elemento que lhe acrescenta informação.

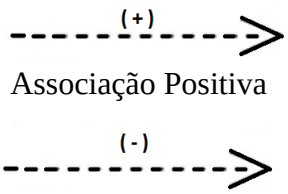
3.3.4.2 Associações positivas e negativas

Entre os requisitos não funcionais em um projeto é comum que exista algum tipo de interligação, que pode ser somativa, ou seja, os requisitos contribuem mutuamente para o mesmo fim o que chamamos de associação positiva ou quando essa relação entre eles é conflitante, ou seja, quando os requisitos podem de alguma forma impactar um ao outro gerando conflito, o que chamamos de associação negativa.

Por exemplo, quando temos um requisitos de disponibilidade e de segurança no mesmo sistema podemos ter uma ocasião de conflito no sistema ou associação negativa, haja vista que uma das estratégias muito utilizadas para a implementação da disponibilidade em sistemas é a replicação de informação o que torna o sistema mais vulnerável a ataques. Por outro lado, quando temos, por exemplo, requisitos de confiabilidade e segurança no mesmo sistema podemos identificar uma associação positiva, uma vez que o requisito de confiabilidade promove estratégias que ajudam a estabelecer o requisito de segurança.

Para representar esses dois tipos distintos de relacionamento no diagrama entre os elementos RNF, utilizamos o formato de seta de dependência em conjunto com sinal de “mais” para associação positiva e o um sinal de “menos” para indicar uma associação negativa, ambas entre parênteses, como mostrado nas figuras abaixo.

Figura 24 – Exemplo de Associação positiva e negativa



Associação Positiva

Associação Negativa

Esses dois tipos de associações foram acrescentados ao diagrama para tornar claro ao leitor do documento os impactos mútuos entre os requisitos para consequentemente respaldar junto ao cliente possíveis decisões de projeto em momentos posteriores.

3.4 Documentação Associada ao Diagrama

Além do diagrama, propomos um *template* associado, o qual detalharia de forma textual os requisitos encontrados. Incluímos nele elementos para a descrição do caso de uso, dos requisitos não funcionais associados e dos riscos.

3.4.1 Especificação dos Casos de Uso

A Tabela 4 apresenta o *template*, em que determinamos que os casos de uso, os requisitos não funcionais e os riscos devem ter um número de identificação associado, para facilitar a rastreabilidade dos requisitos. Considerando que a documentação deve ser direta, não ambígua e de fácil leitura.

Também utilizamos muitos itens já incorporados à documentação textual dos casos de uso, como os itens: Ator principal, Atores secundários, Pré-condições, Pós-condições, Fluxo Principal e Fluxo Alternativo, conforme mostrado em Guedes (2018). E acrescentamos os itens Prioridade (para identificar os casos de uso mais relevantes para o sistema) e Origem/Autor do caso de uso para determinar de onde originou-se esse requisito, como por exemplo de um questionário aplicado, em uma entrevista com o usuário X, etc.

Embora, o conceito de risco esteja associado à probabilidade e ao impacto, nos concentraremos principalmente no tipo do impacto que o requisito oferece ao sistema. Para tanto, utilizaremos a classificação de risco de acordo com a tabela 2 da seção 2.4 deste trabalho, a qual determina quatro categorias de risco: Catastrófico, Crítico, Marginal e Negligenciável.

Tabela 4 – Modelo de Especificação de Requisitos do RNFCase

Nº do Caso de Uso	
Nome do Caso de Uso	
Origem (Autor) do Caso de Uso	
Prioridade	
Objetivo	
Ator Principal	

Atores Secundários	
Pré-condições	
Pós-condições	
Fluxo Principal	
Ações do Usuário	**Ações do Sistema**
Fluxo Alternativo	
Ações do Usuário	**Ações do Sistema**
Fluxo de Exceção	
Ações do Usuário	**Ações do Sistema**
Requisitos Não Funcionais Associados	
Riscos Associados	
Discriminação	**Categoria**

3.4.2 Especificação dos Requisitos Não Funcionais

Consideramos também necessário criarmos um *template,* apresentado na Tabela 5, para a especificação dos Requisitos Não Funcionais, devido a sua importância não apenas para as funções em particular, mas especialmente para o sistema como um todo. Nesse sentido, a especificação textual dos RNF deve constar seu nome e uma descrição do requisito que deve informar como o RNF se relaciona com os Casos de Usos.

Como citado na especificação dos Casos de Uso, o texto a ser inserido na especificação dos RNF também deve ser escrito procurando evitar ambiguidades, de forma clara, concisa e coerente.

Tabela 5 – Modelo de Especificação de Requisitos Não Funcionais

Nome do RNF:	
Descrição	
RF(s) relacionado(s) ao RNF:	

3.5 Considerações sobre a Proposta

Os elementos adicionados ao diagrama Casos de Uso devem ser utilizados com cautela, pois se colocados indiscriminadamente podem comprometer a clareza e tornar o gráfico sobrecarregado de informações.

Para evitar esse tipo de problema pode-se acrescentar os elementos de forma a que apenas os requisitos mais relevantes para o problema sejam abordados. Ou ainda, se utilizar de uma representação por ator ou por iteração.

Em processos de desenvolvimento dirigido a planos, em que é necessário fazer toda a especificação do software antes de começar as demais atividades, é preferível utilizar o diagrama particionado por ator para que se possa relacionar todos os requisitos não funcionais, como também todos os riscos, a fim de ter uma documentação mais completa.

Por outro lado, em processos ágeis, em que a especificação, o projeto e a implementação são feitos de forma intercalada, pode-se utilizar a modelagem por iteração em que se utilizaria o diagrama com os casos de uso do novo incremento juntamente com os requisitos não funcionais e os riscos relacionados. Todavia, para esse tipo de abordagem é necessário um levantamento inicial dos requisitos não funcionais antes do início do desenvolvimento, haja vista eles serem necessários para a construção da arquitetura inicial do sistema, o que recomendamos ser em forma de grafo para que se possa demonstrar os relacionamentos entre eles, como mostraremos nos exemplos posteriores.

3.6 - Comparação com outros Modelos

Propõe-se neste modelo, uma nova visão sobre os requisitos levantados junto ao cliente, levando em consideração não apenas os requisitos funcionais, como também os requisitos não funcionais e os riscos em relação a eles. Para fins de comparação entre este modelo e os demais modelos relatados neste trabalho, definimos alguns critérios, os quais são:

a) **Critério 01 - Representação gráfica dos Requisitos Funcionais e Requisitos Não funcionais**

Este critério define se o modelo possui uma representação gráfica para requisitos e se são abrangidos os funcionais e não funcionais.

b) **Critério 02 - Associação dos Requisitos Funcionais e Requisitos Não funcionais**

Este critério define se existe na representação do modelo algum tipo de associação entre os requisitos funcionais e não funcionais.

c) **Critério 03 - Rastreabilidade dos requisitos a partir da solução**

A rastreabilidade pode ser definida como “o processo de identificar e documentar os elos (ou vínculos) que envolvem um determinado requisito, para que seja possível rastrear sua origem, os artefatos derivados e os demais requisitos relacionados” (VASQUEZ e SIMÕES, 2016). Este critério, portanto, define se é possível identificar mecanismos de rastreabilidade dos requisitos a partir do modelo.

d) **Critério 04 - Associação de Riscos aos requisitos**

O risco pode ser considerado um evento ou condição incerta que, caso aconteça, pode ter um efeito negativo ou positivo em um projeto. Portanto, esse critério define se existe alguma associação dos requisitos a riscos envolvidos a eles.

Após estabelecidos esses critérios, compara-se as abordagens a fim de identificar a presença ou a falta destas características nos modelos.

Tabela 6 – Comparação entre os Modelos

	Critérios			
Modelo	**01**	**02**	**03**	**04**
RNFCase	É uma das propostas principais do modelo	Procura estabelecer uma relação direta entre requisitos funcionais e não funcionais a partir de associações entre eles.	O modelo busca prover rastreabilidade dos requisitos a partir da associação entre eles. Podemos inferir, por exemplo, a causa de criação de mecanismos de segurança (funções, protocolos, etc) a partir do estabelecimento de requisitos não funcionais sobre casos de uso.	Adiciona um elemento para indicar riscos relacionados aos requisitos do sistema.
Árvore de Características	Nesse sentido, constitui-se em uma representação com foco em requisitos funcionais, muito embora possa conter requisitos não funcionais, mas na representação gráfica não há distinção entre eles.	O modelo tem foco na obtenção de requisitos funcionais do sistema, podendo conter requisitos não funcionais mas não é o intuito da abordagem, logo não há uma distinção entre eles, sendo as associações de acordo com a estrutura da árvore.	O modelo consegue trazer elementos de rastreabilidade a medida que relaciona os requisitos a partir das associações feitas da raiz até as folhas.	Não oferece.
Mapas Conceituais	Os requisitos são representados por elementos de conceitos e relações, todavia não há uma preocupação em distingui-los ou defini-los em RF ou RNF.	O modelo não faz distinção entre os requisitos do sistema. Os conceitos podem representar ambos requisitos funcionais e não funcionais, que se associam de acordo com o raciocínio do(s) elaboradores.	Os mapas conceituais podem trazer elementos de rastreabilidade a medida que segue um raciocínio que estabelece conceitos importantes sobre o domínio do problema e os relaciona.	Não oferece.

NFR Framework	Trata apenas de requisitos não funcionais, a fim de refinar requisitos que são difíceis de ser estimados.	Esse tipo de associação não é possível, pois o modelo trata apenas de requisitos não funcionais.	É possível encontrar rastreabilidade dos requisitos à medida que o modelo faz as operacionalizações a partir de requisitos mais abrangentes até o seu refinamento em soluções possíveis, apoiando as decisões de projeto.	Não oferece.
Padrões de requisitos de software (Withall)	A abordagem propõe um grafo para relacionar os requisitos, mas não fica muito claro como fazê-lo.	A abordagem propõe um grafo para relacionar os requisitos, mas não fica muito claro como fazê-lo	A abordagem a partir da sua especificação textual e do grafo proposto tenta relacionar os dois tipos de requisitos, mas não deixa claro como fazê-lo no diagrama.	Não oferece.

Fonte: Elaborada pelo autor, 2020.

A partir desta comparação podemos perceber que a nossa abordagem favorece o entendimento da relação entre requisitos funcionais e requisitos não funcionais propondo uma forma gráfica e que também associa os riscos a estes requisitos, promovendo uma atenção aos riscos inerentes ao projeto já na fase de levantamento e análise dos requisitos, diferente das demais abordagens.

Podemos considerar como ponto negativo da nossa abordagem o fato de que ao acrescentar mais elementos ao diagrama podemos deixá-lo sobrecarregado de informações, para contornar o problema podemos utilizar estratégias como fazer a modelagem por ator ou iteração, ou acrescentar apenas os requisitos mais relevantes para o sistema e utilizar o formato do grafo proposto para detalhar com mais clareza e completude os requisitos.

CAPÍTULO 4 - EXEMPLOS DE UTILIZAÇÃO DO RNFCASE

Para demonstrar a utilização do modelo RNFCase, o aplicamos em dois exemplos de software que serão mostrados abaixo. Para tanto, escolhemos utilizar o diagrama particionado por ator, para melhor visualização dos componentes e interações da proposta, e adicionamos o grafo para cada um deles a fim de demostrar o conjunto dos requisitos não funcionais envolvidos no exemplo.

4.1 Sistema de Controle Escolar

Inicialmente, aplicamos o RNFCase a um software não crítico de Controle Escolar para Instituições de Ensino Fundamental e Médio. O sistema tem por objetivo gerenciar as informações dos alunos no que concerne as suas informações pessoais como Nome, Data de Nascimento, Registro Civil, RG, CPF, Deficiências, etc, além de informações escolares como notas, médias, frequência e histórico. Como também deve guardar informações pessoais e profissionais dos funcionários da escola. Todas essas informações devem ser manter em sigilo, podendo ser acessadas apenas por pessoas autorizadas de acordo com suas funções.

O sistema deve ser acessado por pessoas da área administrativa como secretários, além de professores e alunos. Os usuários do tipo alunos têm acesso ao sistema apenas para acessar os seus próprios dados escolares, os professores, por outro lado, tem acesso aos seus dados pessoais e profissionais, como também são responsáveis por inserir no sistema informações sobre as aulas ministradas nas séries as quais ele foi alocado, como frequências dos alunos, notas e registros de aulas.

O usuário do tipo secretário é o que acumula mais funções no sistema, ele é o responsável por gerenciar as informações privadas dos demais stakeholders como profissionais de educação e alunos, além de funções diretamente relacionadas aos objetivos do negócio como matrículas, manutenção de turmas, criação de históricos escolares e certificados.

Analisando as tarefas e o contexto do sistema, percebemos a necessidade dos requisitos não funcionais de segurança da informação e de integridade dos dados, considerados como os mais relevantes para o sistema e os associamos aos casos de uso que consideramos que são diretamente afetados por eles.

Também adicionamos uma regra de negócio associada aos casos de criação de históricos e certificados dos alunos, na qual os alunos ao finalizarem as séries 9º ano do Ensino Fundamental ou a 3ª série do Ensino Médio, com situação aprovada, devem ter seus históricos

e certificados criados automaticamente pelo sistema, de acordo com as informações inseridas pelo secretário e professores da escola.

Baseado nos requisitos identificados, pudemos associar riscos envolvidos a eles como o uso dos recursos e informações confidenciais por pessoas não autorizadas e dados incorretos serem salvos no servidor sem que o usuário perceba.

Com o RNFCase ainda é possível estabelecer uma associação entre os requisitos não funcionais e determiná-la como positiva ou negativa. No exemplo, achamos necessário ainda esclarecer que existe esta relação entre os requisitos, a qual foi considerada positiva uma vez que as medidas necessárias para estabelecer o requisito de segurança da informação devem contribuir para manter a integridade dos dados no servidor.

Figura 25 – RNFCase aplicado a um Sistema de Controle Escolar

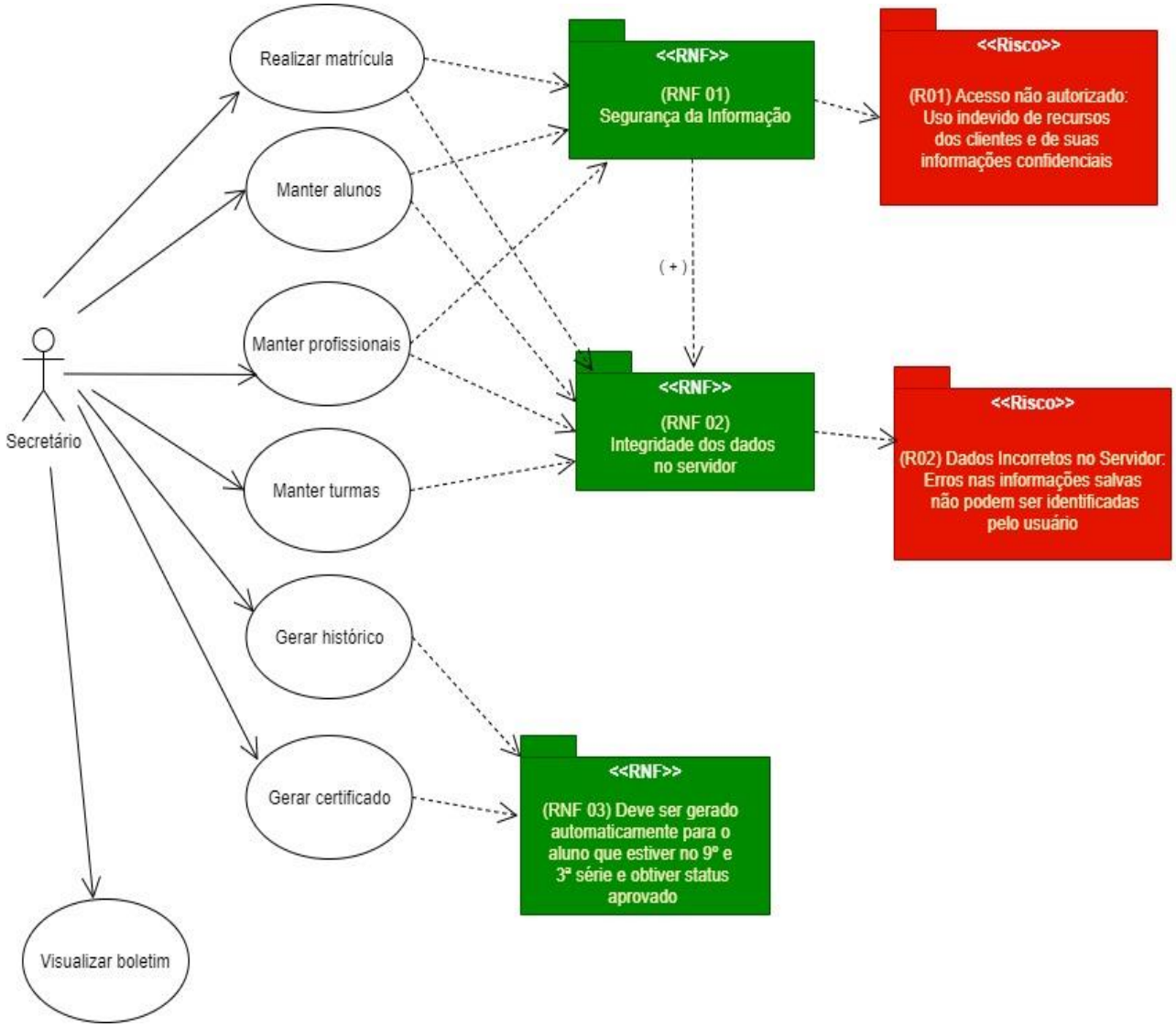


Figura 26 – Grafo do RNFCase aplicado ao sistema de Gestão Escolar

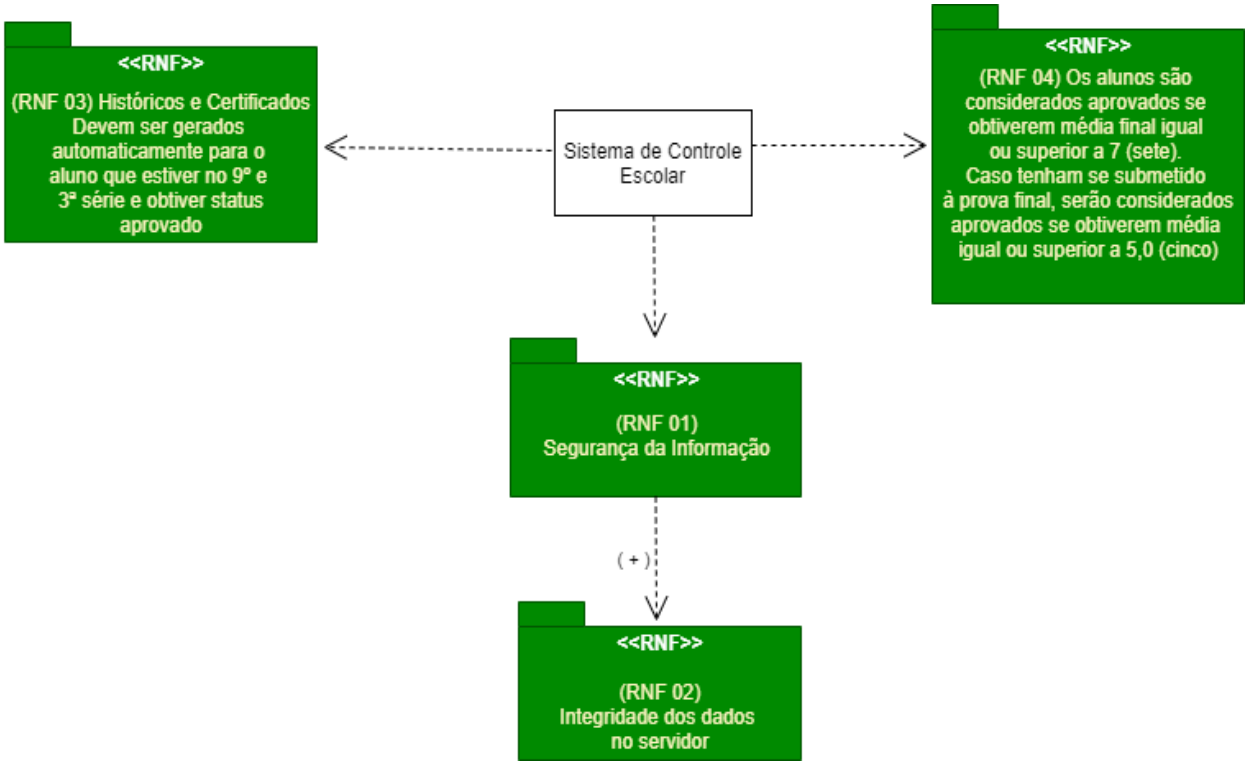


Tabela 7 - Especificação do Caso de Uso Realizar Matrícula

Nº do Caso de Uso	01
Nome do Caso de Uso	Realizar Matrícula
Origem (Autor) do Caso de Uso	Entrevista com diretor escolar
Prioridade	Alta
Objetivo	Fazer a matrícula de um aluno em uma turma de uma determinada série da escola
Ator Principal	Secretário
Atores Secundários	
Pré-condições	- A turma na qual o aluno será matriculado deve existir no sistema (cadastrada

	previamente). - O aluno que será matriculado deve ter um cadastro no sistema com seus dados pessoais.
Pós-condições	
Fluxo Principal	
Ações do Usuário	**Ações do Sistema**
1. O usuário deve escolher a série que o aluno irá se matricular.	
	2. O sistema mostra as opções de turmas para a série escolhida
3. O usuário escolhe a turma que o aluno irá se matricular.	
	4. O sistema retorna as informações da turma escolhida e solicita nome do aluno para pesquisa
5. Usuário digita nome do aluno e solicita para pesquisar no sistema	
	6. Sistema fornece lista de alunos cadastrados no sistema com o nome fornecido em conjunto com data de nascimento e nome da mãe.
7. Usuário escolhe um aluno da lista e confirma os dados do responsável pelo aluno. Em seguida solicita salvar as informações.	
	8. O sistema salva as informações no servidor.

Fluxo Alternativo	
Ações do Usuário	**Ações do Sistema**
1. Usuário digita nome do aluno e solicita pesquisa no sistema	
	2. O sistema fornece lista de alunos cadastrados no sistema com o nome fornecido em conjunto com data de nascimento e nome da mãe.
3. Usuário escolhe um aluno da lista	
	4. O sistema retorna as informações pessoais do aluno
5. Usuário solicita fazer matrícula	
	6. Sistema solicita série que o aluno vai se matricular
7. O usuário deve escolher a série que o aluno irá se matricular.	
	8. O sistema mostra as opções de turmas para a série escolhida
9. O usuário escolhe a turma que o aluno irá se matricular.	
	10. Sistema pede confirmação de matrícula
11. Usuário fornece confirmação de matrícula	
	12. Sistema salva as informações no servidor.

Fluxo de Exceção	
Ações do Usuário	**Ações do Sistema**
1. Usuário digita nome do aluno e solicita pesquisa no sistema	
	2. Busca no servidor verifica que aluno não tem cadastro no sistema. Emitir alerta ao usuário e fornecer opção de cadastramento.
3. Usuário seleciona para iniciar procedimento de cadastro	
Requisitos Não Funcionais Associados	RNF01 – Segurança da Informação
	RNF02 – Integridade dos Dados
Riscos Associados	
Discriminação	**Categoria**
R01 – Acesso não autorizado	Crítico
R02 – Dados incorretos no servidor	Crítico

Tabela 8 - Especificação dos Requisitos Não Funcionais do Sistema de Controle Escolar

Nome do RNF:	RNF01 – Segurança da Informação
Descrição	
Este requisito se relaciona aos casos de uso de funções que agregam informações pessoais de clientes e profissionais do estabelecimento. Logo, o sistema deve fornecer mecanismos de segurança para manter em sigilo os dados dos usuários, como também dos demais recursos da escola, para que apenas pessoas autorizadas tenham acesso.	
RF(s) relacionado(s) ao RNF: Realizar Matrícula, Manter Alunos e Manter Profissionais	

Nome do RNF:	RNF02 – Integridade dos Dados
Descrição	
Este requisito se relaciona aos casos de uso de funções importantes para os objetivos do negócio e de entrada de dados para o sistema, logo deve-se garantir que os dados sejam salvos e mantidos sem erros no servidor.	
RF(s) relacionado(s) ao RNF: Realizar Matrícula, Manter Alunos, Manter Profissionais e Manter Turmas	

4.2 Sistema de Aplicação Bancária

Consideramos que o RNFCase pode ser igualmente aplicado a sistemas críticos em seu levantamento de requisitos, por isso aplicamos a proposta também a um sistema online de aplicação bancária.

Trata-se de um sistema no qual um usuário cliente do banco consegue efetuar operações financeiras sobre sua conta. A aplicação online deve operar em navegadores web e oferecer segurança aos clientes e ao banco quanto às transações financeiras. O cliente deve ser capaz de visualizar a quantia que está em sua conta e todas as transações realizadas nela. Além disso, o sistema deve fornecer ao cliente as funções de transferência entre contas do próprio banco e para outras instituições financeiras, como também realizar pagamentos de boletos através de

código de barras. Adicionalmente, a aplicação deve sugerir e fornecer opções de empréstimos bancários e de investimentos de acordo com o perfil do cliente;

Ao levantar e analisar os requisitos do software para o ator Cliente, identificamos os casos de uso: visualizar saldo, visualizar extrato, fazer transferência, fazer pagamento, realizar empréstimo e realizar investimento. E como requisitos não funcionais: integridade, não repudio, confiabilidade, disponibilidade, segurança da informação e dependabilidade. Todavia, para melhor visualização do diagrama acrescentamos apenas o requisito de dependabilidade que consegue incluir e expressar os demais requisitos, representando apenas no grafo a totalidade dos requisitos do software. Em razão da dependabilidade ser um requisito essencial para o sistema, acrescentamos três riscos (Integridade dos dados, Indisponibilidade e Acesso não autorizado) que podem incorrer se houver falta ou o atendimento parcial do requisito.

Figura 27 – RNFCase aplicado a um Sistema de Aplicação Bancária

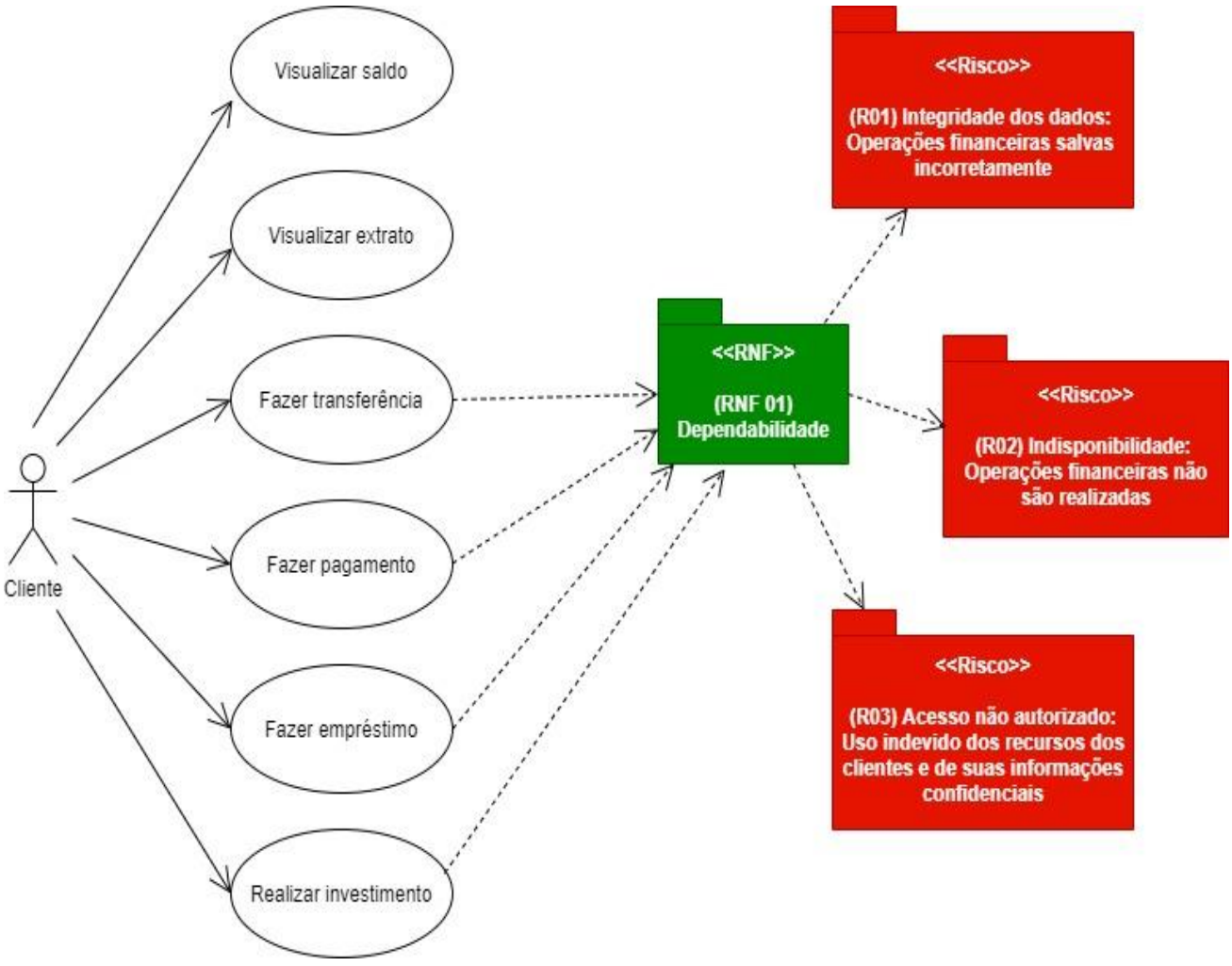
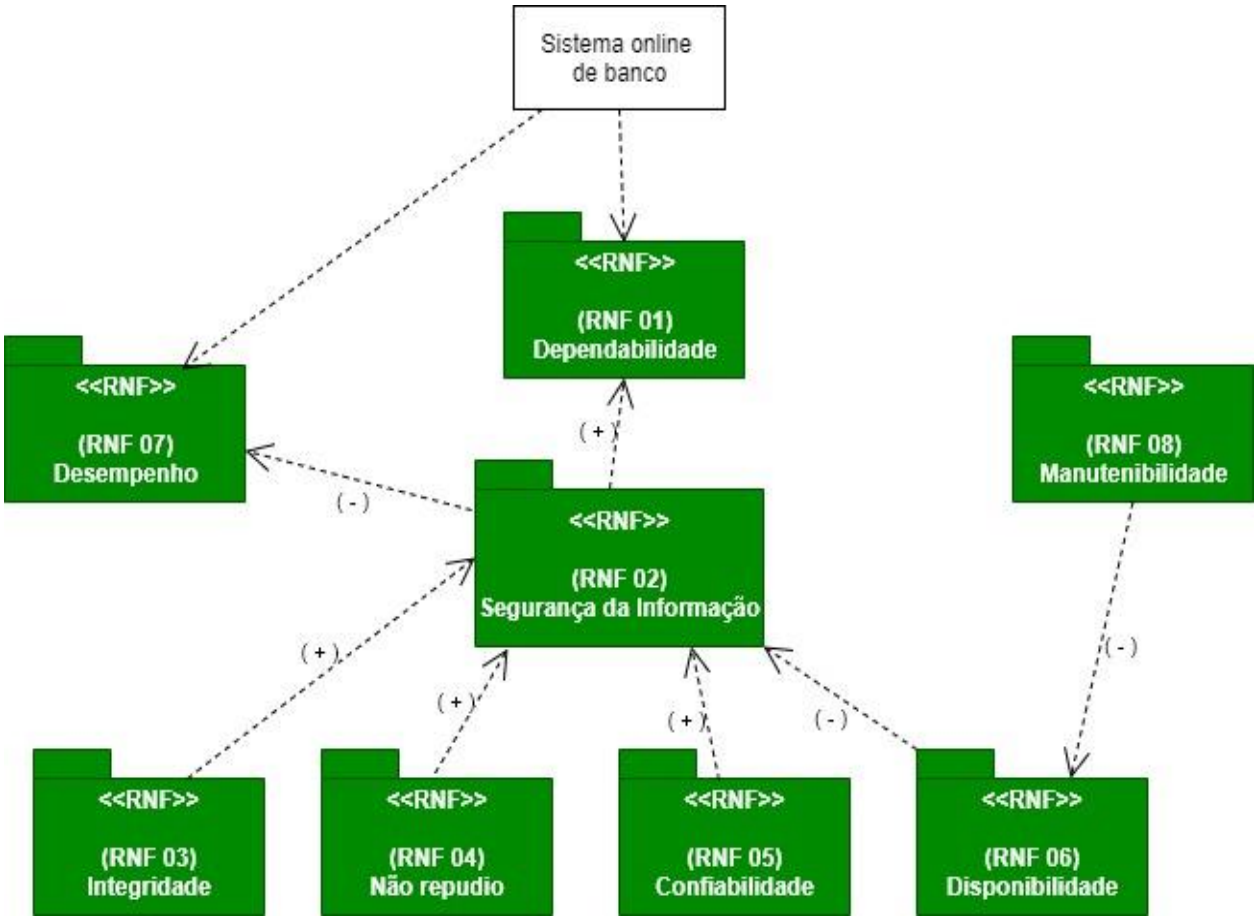


Figura 28 – Grafo do RNFCase aplicado ao sistema de Aplicação Bancária



O grafo da figura 28 mostra todos os requisitos não funcionais elicitados para o software. O objeto sistema é representado por um retângulo branco com sua descrição em seu interior, os RNF's são relacionados por associações de dependência com o objeto sistema e entre si podendo estas associações serem consideradas positivas ou negativas. No exemplo acima, a estrutura do grafo assumiu uma forma hierárquica a partir das características de cada requisito e tivemos associações de dependência consideradas positivas como os requisitos de Integridade, Não Repudio e Confiabilidade que contribuem para a promoção do requisito de Segurança da Informação, como também associações consideradas negativas como o requisito de Segurança da Informação que gera conflito com o requisito de Desempenho e também sofre um impacto negativo da Disponibilidade que pode gerar conflito.

Tabela 9 – Especificação do Caso de Uso Fazer Transferência

Nº do Caso de Uso	03
Nome do Caso de Uso	Fazer Transferência
Origem (Autor) do Caso de Uso	Marcelo, representante do cliente
Prioridade	Alta
Objetivo	Fazer transferência de recursos financeiros entre contas
Ator Principal	Cliente
Atores Secundários	
Pré-condições	Deve existir saldo positivo na conta na data da transferência
Pós-condições	
Fluxo Principal	
Ações do Usuário	**Ações do Sistema**
1. Escolher opção de transferência para contas do mesmo banco	
2. Escolher de qual conta do usuário será feito o débito	
3. Fornecer número da agência e conta destino	
4. Selecionar data da transferência	
5. Fornecer valor da transferência	
	6. Validar informações de transferência

	7. Retornar ao cliente resumo das informações passadas
8. Informar senha da conta	
9. Confirmar transferência	
	10. Emitir comprovante parcial da operação
	11. Confirmar saldo existente na data da transferência
	12. Fazer débito da conta do usuário e creditar na conta destino
	13. Emitir comprovante definitivo
Fluxo Alternativo	
Ações do Usuário	**Ações do Sistema**
1. Escolher opção de transferência para contas de outros bancos	
2. Escolher tipo de transferência (DOC ou TED) e escolher finalidade	
3. Escolher de qual conta do usuário será feito o débito	
4. Fornecer nome do banco de destino ou seu código	
	5. Validar existência de banco e emitir lista com nomes que correspondem ao informado pelo usuário

6. Escolher banco destino conforme lista e fornecer informações da conta destino (Agência, conta, CPF ou CNPJ)	
7. Selecionar data da transferência	
8. Fornecer valor da transferência	
	9. Validar informações de transferência
	10. Retornar ao cliente resumo das informações passadas
11. Informar senha da conta	
12. Confirmar transferência	
	13. Emitir comprovante parcial da operação
	14. Confirmar saldo existente na data da transferência
	15. Fazer débito da conta do usuário e creditar na conta destino
	16. Emitir comprovante definitivo
Fluxo de Exceção	
Ações do Usuário	**Ações do Sistema**
	1. Verificação de saldo inexistente em conta
	2. Emitir alerta ao usuário de cancelamento de operação
Requisitos Não Funcionais Associados	RNF 01 - Dependabilidade

Riscos Associados	
Discriminação	**Categoria**
R01 – Integridade dos Dados	Catastrófico
R02 - Indisponibilidade	Catastrófico
R03 – Acesso não autorizado	Crítico

Tabela 10 - Especificação dos Requisitos Não Funcionais do Sistema de Aplicação Bancária

Nome do RNF:	RNF01 - Dependabilidade
Descrição	
O requisito implica em oferecer um serviço no qual possa-se depositar confiança, através do gerenciamento da confiabilidade, disponibilidade, segurança, integridade e confidencialidade, que são características essenciais para os casos de uso, pois lidam com informações e recursos financeiros dos clientes e da empresa.	
RF(s) relacionado(s) ao RNF: Fazer Transferência, Fazer Pagamento, Fazer Empréstimo e Realizar Investimento	

Nome do RNF:	RNF02 - Segurança da Informação
Descrição	
Como as funções do sistema utilizam informações sigilosas, deve-se proteger as informações da empresa, dos usuários e suas transações através da gestão de sua confidencialidade, integridade e disponibilidade.	
RF(s) relacionado(s) ao RNF: Fazer Transferência, Fazer Pagamento, Fazer Empréstimo e Realizar Investimento	

Nome do RNF:	RNF03 - Integridade
Descrição	
Todas as transações bancárias devem ter seus registros salvos no servidor em sua forma original e sem erros, preservando a precisão, consistência e confiabilidade das informações.	
RF(s) relacionado(s) ao RNF: Fazer Transferência, Fazer Pagamento, Fazer Empréstimo e Realizar Investimento	

Nome do RNF:	RNF04 - Não repudio
Descrição	
O requisito se relaciona aos casos de uso**,** pois deve-se ter o registro de todas as transações realizadas, com comprovação de sua origem, integridade e autenticidade, como forma de proteção dos clientes e da empresa em relação a todas as atividades realizadas.	
RF(s) relacionado(s) ao RNF: Fazer Transferência, Fazer Pagamento, Fazer Empréstimo e Realizar Investimento	

Nome do RNF:	RNF05 - Confiabilidade
Descrição	
O sistema deve realizar as suas operações da maneira que foram especificadas nos requisitos do sistema, durante todo o tempo que for utilizado. Para que isso seja feito, deve-se calcular uma probabilidade de uso sem falhas, e utilizar estratégias de tolerância a falhas e recuperabilidade.	
RF(s) relacionado(s) ao RNF: Fazer Transferência, Fazer Pagamento, Fazer Empréstimo e Realizar Investimento	

Nome do RNF:	RNF06 - Disponibilidade
Descrição	
Como a utilização das funções do sistema pelos clientes se relacionam diretamente com lucro a qual a empresa irá obter, deve-se implementar mecanismos para que o sistema esteja sempre disponível ao cliente.	
RF(s) relacionado(s) ao RNF: Fazer Transferência, Fazer Pagamento, Fazer Empréstimo e Realizar Investimento	

Nome do RNF:	RNF07 - Desempenho
Descrição	
O sistema deve operar conforme especificado. As operações não devem ultrapassar o tempo máximo para sua conclusão.	
RF(s) relacionado(s) ao RNF: Fazer Transferência, Fazer Pagamento, Fazer Empréstimo e Realizar Investimento	

Nome do RNF:	RNF08 - Manutenibilidade
Descrição	
O software deve ser escrito de forma que possa evoluir para atender as necessidades do cliente, portanto deve-se utilizar estratégias a fim de que as manutenções a serem realizadas no sistema sejam feitas de forma facilitada e sem impacto negativo nos demais componentes.	
RF(s) relacionado(s) ao RNF: Fazer Transferência, Fazer Pagamento, Fazer Empréstimo e Realizar Investimento	

CAPÍTULO 5 - CONCLUSÃO

Este trabalho se propôs a desenvolver uma extensão do modelo de Casos de Uso em que fosse possível suprir as necessidades de se relacionar os requisitos funcionais aos requisitos não funcionais, como também considerar os riscos envolvidos a eles.

Para que isso fosse possível foi feito um levantamento bibliográfico para descobrir quais os tipos de modelagem que são utilizadas para o levantamento de requisitos, entre esses modelos estavam os Casos de Uso, a Árvore de Características, os Mapas Conceituais, o NFR Framework e Modelos como o de Withall.

Devido a flexibilidade e fácil entendimento do diagrama Casos de Uso, achou-se viável fazer uma adaptação deste para o nosso propósito através da utilização de seus elementos nativos e adicionando outros elementos já pertencentes ao UML, a fim de integrar conceitos considerados importantes como os Requisitos Não funcionais a partir de um elemento chamado RNF utilizando o formato de pacote com o estereótipo “RNF”, como também o elemento chamado Risco utilizando igualmente o formato de pacote e o estereótipo “Risco”. Além desses elementos também adicionamos associações entre os requisitos as quais podem direcionar o entendimento para tomada de decisões de projeto posteriores ao designar as associações como positivas (ou seja, os requisitos contribuem positivamente entre si) ou negativas (os requisitos contribuem negativamente entre si, convergindo em conflito).

Ao propormos essa nova forma de modelagem, a comparamos com as outras abordagens já existentes na área de requisitos e podemos perceber uma adição positiva para a engenharia de requisitos, haja vista acrescentar uma nova visão para os requisitos que pode contribuir para um bom entendimento dos requisitos entre as partes interessadas, ajudando a identificar as relações entre os tipos de requisitos dos sistemas, como também visualizar os riscos inerentes a essas necessidades do cliente, de uma forma gráfica com elementos para facilitar o entendimento, e então poder respaldar decisões de projeto.

5.1 Limitações da Proposta

O RNFCase se propõe a ser um novo modelo para explorar alguns aspectos dos requisitos de software, como os relacionamentos entre eles, riscos e rastreabilidade, todavia não se destina a ser um modelo completo, sem necessidades de complementação de outros modelos ou documentos.

Além disso, a proposta também se limita em alguns critérios, tais como:

- Os riscos dos requisitos são apenas abordados qualitativamente e não quantitativamente, logo não há um cálculo para a abordagem da probabilidade.
- Há a necessidade de criação de dois elementos gráficos como o diagrama e o grafo para uma abordagem completa dos requisitos não funcionais.

Contudo, o RNFCase se constitui em uma ferramenta simples para ajudar no entendimento dos stakeholders quanto ao problema e aos cenários a serem explorados, agregando conhecimento entre as partes para um bom planejamento e desenvolvimento de uma solução.

5.2 Trabalhos Futuros

Em razão do curto período de desenvolvimento desta proposta de documentação não foi possível avalia-la dentro de projetos em empresas de desenvolvimento de software, portanto, como trabalhos futuros pode-se inferir um estudo através da aplicação desta proposta em projetos de empresas a fim de obter um feedback entre os analistas de sistema, desenvolvedores e usuários sobre o RNFCase aplicando-se entrevistas ou questionários de opinião para que ele possa ser aperfeiçoado.

REFERÊNCIAS

BARDIN, L. **Análise de conteúdo.** Ed. Revista e Ampliada. São Paulo: Edições 70, 2011, 229p.

BARROS, A. J. P. LEHEFELD, N. A. S. **Projeto de Pesquisa:** proposta metodológicas. Rio de Janeiro: Vozes, 1990.

BARROS, A. J. P. LEHEFELD, N. A. S. **Fundamentos de Metodologia Científica.** São Paulo: McGraw-Hill, 1986.

BENITTI,F. B. V.; SILVA, R. C. Padrões de Escrita de Requisitos: um mapeamento sistemático da literatura. In: Workshop em Engenharia de Requisitos, 2011, Rio de Janeiro. **Anais** [...]. Rio de Janeiro: PUC, 2011. p. 160. Disponível em: http://www.inf.pucrio.br/wer/WERpapers/artigos/artigos_WER11/silva.pdf. Acesso em: 20 out. 2019.

CALAZANS, A.T.S. et al. Requisitos de Qualidade de Usabilidade: Análise da Utilização em Sistemas de uma Instituição Financeira. In: Workshop em Engenharia de Requisitos, 2018, Rio de Janeiro. **Anais** [...]. Rio de Janeiro: PUC, 2018. Disponível em: http://www.inf.pucrio.br/wer/WERpapers/artigos/artigos_WER11/silva.pdf. Acesso em: 20 dez. 2019.

CHAVES, I.G.; AGUIAR, Y.P.C.. Product Experience e Requisitos Não Funcionais: um framework de design e sua contribuição à Engenharia de Requisitos. **Blucher Design Proceedings**, v. 2, n. 9, p. 1382-1394, 2016. Disponível em: https://www.proceedings.blucher.com.br/article-details/product-experience-e-requisitos-no-funcionais-um-framework-de-design-e-sua-contribuio-engenharia-de-requisitos-24353. Acesso em: 10 dez.2019.

FRANCH, X., *et al*. PABRE: Pattern-Based Requirements Elicitation, Third International Conference on Research Challenges in Information Science, RCIS 2009 (2009b)

GALIANO, A. J. **O método científico**: Teoria e Prática. São Paulo: Harbra, 1979.

GASTALDO, D.L.; MIDORIKAWA, E.T. Processo de Engenharia de Requisitos Aplicado a Requisitos Não-Funcionais de Desempenho–Um Estudo de Caso. In: Workshop em Engenharia de Requisitos, 2003, Piracicaba. **Anais** [...]. Piracicaba: PUC, 2003.p. 302-316. Disponível em: http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER03/denise_gastaldo.pdf. Acesso em: 10 jan. 2020.

GARVEY, P. R.; LANSDOWNE, Z. F. Risk Matrix: An Approach for Identifying, Assessment and Ranking Program Risks. **Air Force Journal of Logistics**. v. XXII, n. 1, p. 18-22, 1998.

GIL, A. C. **Como elaborar projetos de pesquisa.** 4 ed. São Paulo: Atlas, 2008.

GUEDES, G. T. A. **UML**: Uma abordagem prática. 3 ed. São Paulo: Novatec, 2018.

IEEE. **IEEE Recommended Pratice for Requirements Specifications**. New York: The Institute of Electrical and Electronics Engineers, 1998.

JACOBSON, I., et *al*. **Object-Oriented Software Engineering, A Use Case Driven Approach**. Inglaterra: Addison Wesley, 2 ed. 1992.

LEOPOLDINO, C. B. **Avaliação de Riscos em Desenvolvimento de Software**. 2004. Dissertação (Mestrado em Administração) – Universidade Federal do Rio Grande do Sul, Porto Alegre, 2004. Disponível em: https://www.lume.ufrgs.br/bitstream/handle/10183/4616/000413617.pdf?sequence=1. Acesso em: 15 jan.2020.

MACHADO, F.N.R. **Análise e Gestão de Requisitos de Software – Onde nascem os sistemas**. 3 ed. São Paulo: Érica, 2016. Edição do Kindle.

MAGALHÃES, A.L.C. A Garantia da qualidade e o SQA: sujeito que ajuda e sujeito que atrapalha. **ProQualiti – Qualidade na produção de software**, v. 2, n.2, p. 9-14, 2006. Disponível em: http://www.lbd.dcc.ufmg.br/colecoes/wamps/2006/001.pdf. Acesso em: 20 out. 2019.

NUNES, R. C; FERREIRA, R. N. **Ciência e Tecnologia:** O conhecimento pela independência do Brasil. Goiânia: Vieira, 2003.

OLIVEIRA, C.C. *et al*. Árvore de Características de Software Educativo: Uma Proposta para Elicitação de Requisitos pelo Usuário. In: XXI Simpósio Brasileiro de Informática na Educação,2010, São Luís. **Anais** [...]. São Luis:UFAM, 2010. Disponível em: https://www.br-ie.org/pub/index.php/sbie/article/view/1517. Acesso em: 20 dez. 2019.

FILHO, W.P.P. **Engenharia de Software**: fundamentos, métodos e padrões. 3 ed. Rio de Janeiro: LTC, 2013.

PRADO, P. F.; PRADO, E.P.V.; ALBUQUERQUE, J.P. **Análise de Risco em Projetos de Software**. Disponível em: https://www.researchgate.net/publication/231703381_Analise_de_Riscos_em_Projetos_de_Software. Acesso em: 20 out. 2019.

PRESSMAN, R.S. **Engenharia de Software**: uma abordagem profissional. 7. ed. Porto Alegre: AMGH, 2011.

RAUEN, F. J. **Elementos de iniciação a pesquisa científica**. Rio Grande do Sul: Nova Era, 1999.

ROBINSON, G. *et al*. Modelando Requisitos Especificados com Mapas conceituais através da UML-MC. In: XXV Simpósio Brasileiro de Informática na Educação,2014, São Luís. **Anais** [...]. São Luis:UFAM, 2014. Disponível em: https://www.researchgate.net/publication/237538177_Modelando_Requisitos_Especificados_com_Mapas_conceituas_atraves_da_UML-MC. Acesso em: 20 dez. 2019.

ROLIM, L.H.M.L. Utilização de Mapas Conceituais em Engenharia de Software: Projetando uma ferramenta case. com Mapas conceituais através da UML-MC. In: Anais do 12º Encontro de Iniciação Científica e Pós-Graduação do ITA,2006, São Luís. **Anais** [...]. São Luís: UFAM, 2006. Disponível em: https://www.researchgate.net/publication/237538177_Modelando_Requisitos_Especificados_com_Mapas_conceituas_atraves_da_UML-MC. Acesso em: 20 dez. 2019.

SEVERINO, A. J. **Metodologia do Trabalho Científico.** 22 ed. São Paulo: Cortez Editora, 2002.

SOMMERVILLE, I. **Engenharia de Software**. 9 ed. São Paulo: Pearson Prentice Hall, 2011.

SOMMERVILLE, I. **Engenharia de Software**. 10 ed. São Paulo: Pearson Prentice Hall, 2018.

STANDISH GROUP INTERNATIONAL. **The chaos report**. United States of America, 2015.

SWEBOK**. Guide to the Software Engineering Body of Knowledge**. 2004. Disponível em: http://www.swebok.org. Acesso em: 20 dez 2019.

TORO, A. D., et al. A Requirements Elicitation Approach Based in Templates and Patterns. In: Proceedings 2nd Workshop on Requirements Engineering, 1999, [S.l]. **Anais** [...].

U.S. AIR FORCE.Software Risk Abatement. September 30, 1998.

VAZQUEZ, C.E.; SIMÕES, G.S. **Engenharia de Requisitos**: software orientado ao negócio.1. ed. Rio de Janeiro: BRASPORT, 2016.

WAZLAWICK, R.S. Uma reflexão sobre a pesquisa em ciência da computação à luz da classificação das ciências e do método científico. **Revista de Sistemas de Informação da FSMA**, v. 6, p. 3-10, 2010. Disponível em: https://www.researchgate.net/publication/216546082_Uma_Reflexao_sobre_a_Pesquisa_em_Ciencia_da_Computacao_a_Luz_da_Classificacao_das_Ciencias_e_do_Metodo_Cientifico. Acesso em: 10 jan. 2019.

WIEGERS, K. **Software Requirements (Developer Best Practices)** . Washington: Pearson Education:, 2013.

WITHALL, S. Software Requirement Patterns. Washington: Microsoft Press.Appendix: Springer-Author Discount, 2007.

XAVIER, L. **Integração de Requisitos não-funcionais a processos de negócio: integrando BPMN e RNF**. Recife: UFPE, 2009. Disponível em: https://www.semanticscholar.org/paper/Integra%C3%A7%C3%A3o-de-Requisitos-N%C3%A3o-Funcionais-a-Processos-XavierAlencar/5d364743c75491ba9d9a745a4ada256f51246b68. Acesso em: 10 jan. 2020.