UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE QUIXADÁ
CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

ANDRÉ HENRIQUE CONJO

PROPOSTA DE UMA FERRAMENTA *WEB* PARA ASSISTÊNCIA À MODELAGEM DE REQUISITOS NÃO FUNCIONAIS DE *SOFTWARE*

QUIXADÁ
2022

ANDRÉ HENRIQUE CONJO

PROPOSTA DE UMA FERRAMENTA *WEB* PARA ASSISTÊNCIA À MODELAGEM DE REQUISITOS NÃO FUNCIONAIS DE *SOFTWARE*

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientador: Prof. Dr. Antônio Joel Ramiro de Castro.

QUIXADÁ

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C74p Conjo, André Henrique.
Proposta de uma ferramenta web para assistência à modelagem de requisitos não funcionais de software / André Henrique Conjo. – 2022.
53 f. : il.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2022.
Orientação: Prof. Dr. Antônio Joel Ramiro de Castro.

1. Modelagem de software. 2. Requisitos não funcionais (Engenharia de sistemas). 3. Projetos de software. I. Título.

CDD 005.1

ANDRÉ HENRIQUE CONJO

PROPOSTA DE UMA FERRAMENTA *WEB* PARA ASSISTÊNCIA À MODELAGEM DE REQUISITOS NÃO-FUNCIONAIS DE *SOFTWARE*

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. Dr. Antônio Joel Ramiro de Castro (Orientador)
Universidade Federal do Ceará (UFC)

_____

Prof. Dr. Alberto Sampaio Lima
Universidade Federal do Ceará (UFC)

_____

Prof. Me. Thiago Werlley Bandeira da Silva
Universidade Federal do Ceará (UFC)

_____

Prof. Me. Leonardo Torres Marques
Universidade Federal do Ceará (UFC)

AGRADECIMENTOS

Agradeço aos meus professores e colegas, que mudaram minha vida desde que atravessei pela primeira vez os portões da UFC.

Agradeço também a minha família, principalmente a meu pai Celio Monteiro Machado, que esteve comigo durante essa trajetória.

RESUMO

Um projeto de desenvolvimento de *software* deve satisfazer objetivos (qualidade, performance, ambiente e outras) que, normalmente, são modelados como Requisitos Não Funcionais (RNF). Os RNFs não estão relacionados com as funções específicas oferecidos pelo *software*, mas sim com suas propriedades, como confiabilidade e tempo de resposta. O objetivo geral da monografia foi desenvolver uma ferramenta *web,* denominada SIGSoft 1.0, para auxiliar um projetista de *software* a modelar seus RNFs através de *softgoals* (SIGs) inspirado no *NFR-Framework*, ajudando-os a prever se haverá influências entre eles durante o processo de modelagem. Dessa forma, pretendeu-se com o presente trabalho solucionar as principais lacunas existentes em ferramentas envolvendo os RNFs disponíveis no mercado e fornecer uma biblioteca de modelos para reutilização. A partir de tais objetivos, juntamente com a elaboração da ferramenta, foram realizados períodos de testes com estudantes de TI (tecnologia de informação), com a intencionalidade de desenvolver de forma eficiente a ferramenta proposta. Referente a base teórica desta pesquisa sobre Engenharia de Requisitos, Requisitos Funcionais, Requisitos Não Funcionais e Modelagem de Sistema, utilizamos os estudos de Sommerville (2011), Novais (2019), Pressman (2011), Andrade (2019), além de outros estudiosos. Assim, ao trazer para o centro das discussões essas problemáticas, proporciona-se uma avaliação do cenário de mercado atual referente as ferramentas de modelagem de Requisitos Não Funcionais. Permite, também, que novas ferramentas utilizadas na modelagem de RNFs através de SIGs possam ser desenvolvidas, utilizando como base a ferramenta proposta nesse trabalho.

Palavras-chave: Requisitos Não Funcionais. Ferramenta. Modelagem de requisitos.

ABSTRACT

A software development project must satisfy objectives (quality, performance, environment and others) that are usually modeled as Non-Functional Requirements (NFR). NFRs are not related to the specific functions offered by the software, but to its properties, such as reliability and response time. The general objective was to develop a web tool, called SIGSoft 1.0, to help a software designer to model their NFRs through softgoals (SIGs) inspired by the NFR-Framework, helping them to predict if there will be influences between them during the modeling process. In this way, the present work was intended to solve the main gaps in tools involving the NFRs available on the market and to provide a library of models for reuse. Based on these objectives, together with the elaboration of the tool, test periods were carried out with TI students (information technology), with the intention of efficiently developing the proposed tool. Regarding the theoretical basis of this research on Requirements Engineering, Functional Requirements, Non-Functional Requirements and System Modeling, we used the studies of Sommerville (2011), Novais (2019), Pressman (2011), Andrade (2019), in addition to other scholars. Thus, by bringing these issues to the center of discussions, it provides an assessment of the current market scenario regarding Non-Functional Requirements modeling tools. It also allows new tools used in the modeling of NFRs through SIG to be developed, using the tool proposed in this work as a basis.

Keywords: Non-Functional Requirements. Tool. requirements modeling

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

RE *Requirements Engineering* (Engenharia de Requisitos)

RNF Requisito Não Funcional

SIG *Softgoal Interdependency Graph*

SUMÁRIO

1 INTRODUÇÃO

Essencialmente, a utilidade de um sistema é determinada por seus requisitos (funcionais e não funcionais) como usabilidade, flexibilidade, desempenho, interoperabilidade e segurança. Ambos devem ser levados em consideração no desenvolvimento de um sistema de *software* de qualidade, tendo em vista que o conceito de qualidade é fundamental para a engenharia de *software* (CHUNG, *et al*. 2012).

Geralmente os projetos de *software*s tem processos de desenvolvimento orientados a Requisitos Funcionais, ou seja, os desenvolvedores gastam suas energias na implementação de funcionalidades da aplicação. Toda via, assim como afirma Chung *et al*. (2012), a maioria dos problemas de sistemas são comumente relacionados aos Requisitos Não Funcionais (apresentados no trabalho como RNF) do que aos Requisitos Funcionais: por exemplo, a baixa produtividade, o processamento lento, o alto custo e a baixa qualidade; o que leva a um cliente insatisfeito.

Como apresentado por Sommerville (2011), os RNFs de um *software* são aqueles que não estão relacionados com as funções específicos oferecidos pelo *software* (o que o *software* faz), mas sim com as propriedades do *software*, como confiabilidade e tempo de resposta (como o *software* faz). Dessa forma, um projeto de desenvolvimento de *software* deve satisfazer objetivos (qualidade, performance, ambiente e outras) que, normalmente, são modeladas como Requisitos Não Funcionais (WIEGERS; BEATTY, 2013). De acordo com Wiegers e Beatty (2013) os RNFs são de extrema importância para um projeto que exige qualidade (sendo considerados atributos de qualidade), já que eles estão diretamente ligados a arquitetura do sistema. Como é algo intrínseco ao projeto, não é proveitoso levá-los em consideração apenas no final do projeto, tendo em vista que o trabalho de implementação de um atributo de qualidade requerer uma série de Requisitos Funcionais (CHUNG, *et al.* 2012).

Em alguns casos, há choque de interesses entre diferentes RNFs de um sistema. Suponha-se que um RNF A pode influenciar negativamente em um RNF B, por exemplo: Segurança influencia negativamente em Desempenho, já que foi adicionado uma ou mais camadas de validação. Tal ocorrência poderia prejudicar profundamente a construção do projeto.

Atualmente, existem algumas ferramentas destinadas a modelagem dos RNFs no mercado. A Star-UML permitia a criação de diagramas em UML (Linguagem de Modelagem Unificada, do inglês *Unified Modeling Language*). Desta forma, projetava modelos rápido e facilmente no padrão UML, disponibilizando recursos como a abordagem básica dos

diagramas, reutilização de modelos, entre outros. A NDR-Tool, outra ferramenta, disponibilizava ao usuário a montagem de gráfico SIG partindo de catálogos que já estão dispostos na ferramenta, catálogos de RNFs para consulta por categorias (como Usabilidade e Segurança). Essas ferramentas foram utilizadas como fonte de comparação durante o desenvolvimento da ferramenta do presente trabalho, utilizando-se dos pontos positivos e aperfeiçoando os pontos negativos.

Com isso, o objetivo principal do presente trabalho foi desenvolver uma ferramenta *web*, denominada SIGSoft 1.0, para auxiliar o projetista durante o processo de modelagem de RNFs através de *softgoals* (SIGs). Também possibilita uma avaliação do cenário de mercado atual referente as ferramentas de modelagem de RNFs, realizando uma análise entre modelos de ferramentas, expondo uma comparação entre as principais características identificadas e assinalando os pontos positivos e negativos de cada abordagem. O *NFR-Framework*, utilizado para representar e registrar o processo de projeto e raciocínio em grafos (SIGs), serviu como inspiração no processo de construção da ferramenta proposta nesta monografia. Assim, uma ferramenta *web* mais intuitiva para realizar a modelagem de RNFs através de SIGs foi desenvolvida e disponibilizada para todos os que realizam a modelagem da arquitetura de uma aplicação utilizando RNFs.

Desta forma, foi elaborada um período teste da SIGSoft 1.0 com estudantes de TI (tecnologia de informação), com a intencionalidade de desenvolver de forma eficiente a ferramenta proposta partindo dos pontos levantados por eles (a usabilidade, eficiência, entre outros).

O trabalho está organizado em seis (6) capítulos. O primeiro apresenta a introdução da pesquisa, realizando breves aspectos teóricos, objetivos, hipóteses e questões presente na pesquisa. No segundo capítulo apresenta-se a fundamentação teórica sobre Engenharia de Requisitos, Requisitos Funcionais, Requisitos Não Funcionais e Modelagem de Sistema. O terceiro capítulo se encontra as análises e comparações deste com outros trabalhos. Em seguida, no quarto capítulo, é desenvolvido os procedimentos metodológicos e seus devidos arranjos, materiais e técnicas. No quinto capítulo, é a apresentação da ferramenta, juntamente com o teste desenvolvido com os estudantes de cursos da área de TI (tecnologia de informação). O último capítulo, por sua vez, apresenta as considerações finais da pesquisa.

2 FUNDAMENTAÇÃO TEÓRICA

Capítulo dedicado a apresentação das descrições e definições dos conceitos utilizados para este trabalho. Apresenta-se ao decorrer do capítulo os tópicos sobre Engenharia de Requisitos, assim como também sobre os Requisitos Funcionais e os Requisitos Não Funcionais. Realiza-se uma breve apresentação do *NFR-Framework* e os *Softgoals*, o conceito de Modelagem de Sistema e as ferramentas destinadas a modelagem dos RNFs utilizados no trabalho.

2.1 Engenharia de Requisitos

Os requisitos de um sistema, de acordo com Sommerville (2011), são as descrições do que o sistema deve fazer, os serviços que oferecem e as restrições a seu funcionamento. Esses requisitos refletem as necessidades dos clientes para um sistema que serve a uma finalidade determinada. Essas necessidades variam entre controlar um dispositivo, colocar um pedido ou encontrar informações. O processo de analisar, descobrir, verificar e documentar esses serviços e restrições é chamado Engenharia de Requisitos (RE, do inglês *Requirements Engineering*).

A RE pretende sistematizar o processo de definição de requisitos. “Essa sistematização é necessária porque a complexidade dos sistemas exige que se preste mais atenção ao correto entendimento do problema antes do comprometimento de uma solução” (CYSNEIROS, s.d, p.16). Assim, a Engenharia de Requisitos estrutura e auxilia na documentação dos requisitos de um sistema.

De acordo com Novais (2019), a RE é a disciplina de engenharia que busca compreender e demonstrar o que deverá ser feito, antes de definir como será feito. De acordo com o autor, “a medida primária do sucesso de um *software* ou sistema corresponde ao quanto ele atinge o *propósito* para o qual foi concebido” (NOVAIS, 2019, p. 29, grifo do autor). Novais também afirma que a representação de requisito pode ser estruturada de diversas formas, e que sua maioria está restrita a lugares específicos.

> A representação de requisitos pode ser realizada de diversas formas, como por exemplo: texto, tabelas, modelos gráficos, linguagens formais de especificação de requisitos e até mesmo formulações matemáticas. Algumas destas representações estão restritas ao meio acadêmico ou a indústrias muito especificas. Na maioria das indústrias, a especificação de requisitos é tradicionalmente realizada de forma

> sentencial, ou seja, feita em linguagem textual ou por meio de tabelas contendo o texto (tabular) (NOVAIS, 2019, p. 31).

Pressman (2011, p. 127) reitera que a RE "é uma ação de engenharia de *software* importante que se inicia durante a atividade de comunicação e continua na de modelagem. Ela deve ser adaptada às necessidades do processo, do projeto, do produto e das pessoas que estão realizando o trabalho". A RE desenvolve uma base para o projeto e a construção. Essa base possibilita que as necessidades dos clientes sejam atendidas mais positivamente.

> A engenharia de requisitos constrói uma ponte para o projeto e para a construção [...], a jornada através da ponte nos leva bem à frente no projeto, permitindo que examinemos o contexto do trabalho de *software* a ser realizado; as necessidades específicas que o projeto e a construção devem atender; as prioridades que orientam a ordem na qual o trabalho deve ser completado e as informações, funções e comportamentos que terão um impacto profundo no projeto resultante (PRESSMAN, 2011, p. 127)

De acordo com Pressman (2011), a RE abrange sete (7) etapas distintas: concepção, levantamento, elaboração, negociação, especificação, validação e gestão. "É importante notar que algumas delas ocorrem em paralelo e todas são adaptadas às necessidades do projeto" (PRESSMAN, 2011, p. 127). Esses requisitos são utilizados para compreender o que o cliente deseja, as necessidades, viabilidade, e até possíveis ambiguidades, negociando soluções razoáveis à medida que o sistema operacional é desenvolvido.

A concepção seria o início do projeto. Em termos gerais, a grande maioria dos projetos iniciam-se quando é determinada a necessidade do negócio. Na concepção do projeto, "estabelecemos um entendimento básico do problema, as pessoas que querem uma solução, a natureza da solução desejada e a eficácia da comunicação e colaboração preliminares entre os demais interessados e a equipe de *software*" (PRESSMAN, 2011, p. 128).

O levantamento seriam os objetivos que devem ser alcançados, como o sistema ou produto é/deve ser utilizado, como ele atende as necessidades da empresa, etc.

A elaboração é a etapa em que as informações passadas pelos clientes durante a primeira etapa são polidas e melhoradas. "A elaboração é guiada pela criação e refinamento de cenários de usuários que descrevem como o usuário final (e outros atores) irão interagir com o sistema" (PRESSMAN, 2011, p. 128).

Na negociação, assim como o nome sugere, a etapa para definir os requisitos e os termos de prioridade entre os clientes e usuários. "Usando uma abordagem interativa que priorize os requisitos, avalie seus custos e riscos, bem como trate dos conflitos internos,

requisitos são eliminados, combinados e/ou modificados, de modo que cada parte atinja certo nível de satisfação" (PRESSMAN, 2011, p. 128).

A especificação, de Acordo com Pressman (2011, p.128), "pode ser um documento por escrito, um conjunto de modelos gráficos, um modelo matemático formal, um conjunto de cenários de uso, um protótipo ou qualquer combinação dos fatores citados".

No processo de validação, os artefatos produzidos são analisados. Durante esse processo, os requisitos de *software* são examinados para garantir que eles tenham sidos desenvolvidos de forma não ambígua. Esse processo é necessário para que os possíveis erros sejam encontrados e corrigidos, e que eles estejam de acordo com os padrões estabelecidos para o produto, processo e projeto. "A equipe [...] examina a especificação em busca de erros no conteúdo ou na interpretação, áreas que sejam necessários esclarecimentos, informações faltantes, inconsistências, requisitos conflitantes ou requisitos irreais (inatingíveis)" (PRESSMAN, 2011, p. 130).

A gestão, por sua vez, ocorre durante a vida de um sistema. Gestão de requisitos é "um conjunto de atividades que ajuda a equipe de projeto a identificar, controlar e acompanhar as necessidades e suas mudanças a qualquer momento enquanto o projeto prossegue" (PRESSMAN, 2011, p. 130).

De acordo com Sommerville (2011) fazer uma clara separação entre os diferentes níveis de descrição é de suma importância para não ocorrer falhas. Para exemplificar e facilitar a descrição detalhada do que o sistema deve fazer, esses requisitos são definidos entre "requisitos de usuário" e "requisitos de sistema".

Requisitos de usuário são declarações, em uma linguagem natural com diagramas, de quais serviços o sistema deverá fornecer a seus usuários e as restrições com as quais este deve operar. Requisitos de sistema são descrições mais detalhadas das funções, serviços e restrições operacionais do sistema de *software*. O documento de requisitos do sistema (às vezes, chamado especificação funcional) deve definir exatamente o que deve ser implementado. Pode ser parte do contrato entre o comprador do sistema e os desenvolvedores de *software.*

Os requisitos de sistema retratam informações mais específicas sobre os serviços juntamente com as funções do sistema que podem e/ou devem ser implementadas, enquanto os requisitos de usuários são informações mais gerais.

Utilizando-se do Sistema de Gerenciamento da Saúde Mental de Pacientes (MHC-PMS, do inglês *Mental Health Care Patient Management System*), utilizado como exemplo por Sommerville (2011), pode-se compreender como um requisito pode ser expandido em

diversos requisitos de sistemas. Na figura 1, evidência uma diferença inerente entre os requisitos de sistema (fornecem informações especificas sobre os serviços e funções do sistema que devem ser implementados) e de usuário (mais gerais).

Figura 1 – Requisitos de usuário e de sistema

Definição de requisitos de usuário

1. O MHC-PMS deve gerar relatórios gerenciais mensais que mostrem o custo dos medicamentos prescritos por cada clínica durante aquele mês.

Especificação de requisitos de sistema

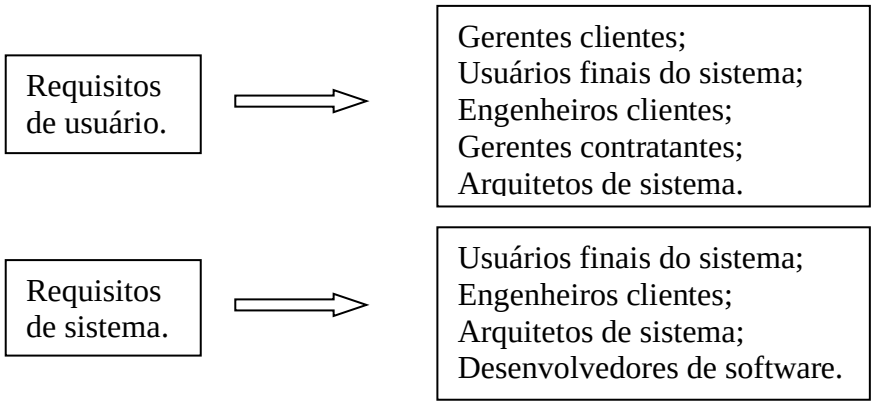
1.1 No último dia útil de cada mês deve ser gerado um resumo dos medicamentos prescritos, seus custos e as prescrições de cada clínica; 1.2 Após 17:30h do último dia útil do mês, o sistema deve gerar automaticamente o relatório para impressão; 1.3 Um relatório será criado para cada clínica, listando os nomes dos medicamentos, o número total de prescrições, o número de doses prescritas e o custo total dos medicamentos prescritos; 1.4 Se os medicamentos estão disponíveis em diferentes unidades de dosagem (por exemplo, 10mg, 20mg), devem ser criados relatórios separados para cada unidade; 1.5 O acesso aos relatórios de custos deve ser restrito a usuários autorizados por uma lista de controle de gerenciamento de acesso.

Fonte: Sommerville (2011).

As variedades de requisitos existentes dos sistemas de *software* necessitam que os requisitos sejam escritos em diferentes níveis de detalhamento. Para Sommerville (2011, p.74) os diversos níveis de detalhamento é preciso para que "diferentes leitores possam usá-los de diversas maneiras" e, consequentemente, transmitirem informações sobre o sistema de forma ampla para diferentes níveis de clientes.

Na figura 2, também apresentado por Sommerville (2011), apresenta os possíveis leitores dos requisitos de usuários e de sistema. De acordo com o autor, os leitores dos requisitos de usuário normalmente não se preocupam em como o sistema será implementado; como gerentes que não estão interessados nos recursos detalhados do sistema.

Figura 2 – Leitores de diferentes tipos de especificação de requisitos



Fonte: Sommerville (2011).

Diferente dos requisitos de usuário, os leitores dos requisitos de sistema "precisam saber mais detalhadamente o que o sistema fará, porque estão interessados em como ele apoiará os processos dos negócios ou porque estão envolvidos na implementação do sistema" (SOMMERVILLE, 2011, p. 74). Assim como afirma Pressman (2011), o fato de existir leitores diversos para o sistema permite que esses requisitos sejam explorados de formas diferentes e de pontos de vistas diversos, e cada um desses grupos contribui com informações durante o processo da RE. Como por exemplo,

> O grupo de marketing está interessado nas funções e recursos que irão suscitar o mercado potencial, facilitando a venda do novo sistema. Os gerentes comerciais estão interessados em um conjunto de recursos que pode ser construído dentro do orçamento e que estarão prontos para atender às oportunidades definidas de ingresso no mercado. Os usuários finais podem querer recursos que sejam familiares a eles e que sejam fáceis de aprender e usar. Os engenheiros de software podem estar preocupados com funções invisíveis aos interessados não técnicos, mas que possibilitam uma infraestrutura que dê suporte a um maior número de funções e recursos comercializáveis. Os engenheiros de suporte talvez enfoquem a facilidade de manutenção do software (PRESSMAN, R. S. 2011, p. 130).

As informações serão coletadas, e devido à alta variedade, poderá ocorrer conflito entre requisitos emergentes. Pressman (2011, p. 130) afirma que, dessa forma, "deve-se classificar as informações de todos os interessados (inclusive os requisitos inconsistentes e conflitantes) de maneira que permita aos tomadores de decisão escolher um conjunto internamente consistente de requisitos para o sistema". Dessa forma, os requisitos de *software* variam de diferentes tipos de acordo com os problemas e soluções estabelecidas entre os usuários. Esses requisitos podem ser definidos em dois (2) tipos: os funcionais e os não funcionais, explicados mais detalhadamente a seguir.

2.2 Requisitos Funcionais e Não Funcionais

De acordo com Sommerville (2011), os requisitos de *software* – explicados anteriormente – são frequentemente classificados como Requisitos Funcionais e Requisitos Não Funcionais. Eles se diferem da seguinte forma: Os Requisitos Funcionais 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. Em alguns casos, os Requisitos Funcionais também podem explicitar o que o sistema não deve fazer; os Requisitos Não Funcionais, por sua vez, 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. Como afirma Cysneiros (s.d, p. 22), em grosso modo, "[...] podemos dizer que um Requisito Funcional expressa algum tipo de transformação que tem lugar no *software*, enquanto um Requisito Não Funcional expressa como essa transformação irá se comportar ou que qualidades específicas ela deverá possuir".

Nesse contexto, os Requisitos Funcionais descrevem "o que" o sistema deve fazer, o modo em que o sistema deve reagir de acordo com entradas específicas, assim como também deve mostrar como o sistema deve se comportar em determinadas situações. Já os Requisitos Não Funcionais estabelecem algumas restrições sobre "como" os Requisitos Funcionais serão implementados, ou seja, "restringem *como* o sistema deve realizar o *o que*, e inclui restrições de custos, performance, portabilidade, robustez e outros" (ANDRADE et al. 2019, p. 190, grifo do autor).

Para o guia PMBOK (Guia do Conhecimento em Gerenciamento de Projetos), os requisitos descrevem funções, características e atributos do produto esperados pelas partes interessadas. O guia define os dois (2) requisitos da seguinte forma:

> *Requisitos funcionais:* os requisitos funcionais descrevem os comportamentos do produto. Exemplos incluem ações, processos, dados e interações que o produto deve executar;
> *Requisitos não funcionais:* os requisitos não funcionais complementam os requisitos funcionais e descrevem as condições ou qualidades ambientais requeridas para que o produto seja eficaz. Exemplos incluem: confiabilidade, proteção, desempenho, segurança, nível de serviço, suportabilidade, retenção/descarte, etc. (UM GUIA DO CONHECIMENTO EM GERENCIAMENTO DE PROJETOS, 2017, p. 185, grifo do autor).

Sommerville (2011, p. 75), também afirma que “os requisitos de sistema não apenas especificam os serviços ou as características necessárias ao sistema, mas também a funcionalidade necessária para garantir que esses serviços/características sejam entregues corretamente”. Para o autor, distinguir entre os diferentes tipos de requisitos, na realidade, não é tão claro como as definições simples sugerem, tendo em vista que por vezes, ambos os requisitos estão interligados.

> Um requisito de usuário relacionado com a proteção, tal como uma declaração de limitação de acesso a usuários autorizados, pode parecer um requisito não funcional. No entanto, quando desenvolvido em mais detalhes, esse requisito pode gerar outros requisitos, claramente funcionais, como a necessidade de incluir recursos de autenticação de usuário no sistema. Isso mostra que os requisitos não são independentes e que muitas vezes geram ou restringem outros requisitos (SOMMERVILLE, 2011, p. 75).

A seguir será apresentado uma visão mais especifica das diferenças entre os Requisitos Funcionais e Requisitos Não Funcionais. Os Requisitos Não Funcionais, por sua vez, são especificados mais profundamente, tendo em vista que é o foco do presente trabalho.

2.2.1 Os Requisitos Funcionais

Assim como mostrando anteriormente, os Requisitos Funcionais de um sistema descrevem o que ele deve fazer. Eles possuem um grau de dependência do tipo de *software* que está sendo desenvolvido, pela abordagem adotada pela organização referente aos requisitos e aos possíveis usuários.

Para Sommerville,

> “Quando expressos como requisitos de usuário, os requisitos funcionais são normalmente descritos de forma abstrata, para serem compreendidos pelos usuários do sistema. No entanto, requisitos de sistema funcionais mais específicos descrevem em detalhes as funções do sistema, suas entradas e saídas, exceções etc” (SOMMERVILLE, 2011, p. 75).

Então, os Requisitos Funcionais podem variar entre os requisitos gerais, abrangendo assim o que o sistema deve executar, indo até os requisitos mais específicos, que remetem as formas e ao sistema de trabalho em uma organização.

Utilizando como exemplo os Requisitos Funcionais do sistema MHC-PMS[1], citado por Sommerville (2011) em seu estudo, o sistema deve então possibilitar o

[1] Sistema de Gerenciamento da Saúde Mental de Pacientes (MHC-PMS, do inglês *Mental Health Care Patient Management System*)

armazenamento sobre os pacientes em tratamento por problemas de saúde mental da seguinte forma, apresentando os recursos específicos que devem ser oferecidos pelo sistema, em diferentes níveis de detalhamento:

1. Um usuário deve ser capaz de pesquisar as listas de agendamentos para todas as clínicas;
2. O sistema deve gerar a cada dia, para cada clínica, a lista dos pacientes para as consultas daquele dia;
3. Cada membro da equipe que usa o sistema deve ser identificado apenas por seu número de oito (8) dígitos.

A imprecisão relacionada aos requisitos é a causa de muitos problemas em *softwares*. Pedidos ambíguos compreendidos de forma equivocada pelo desenvolvedor podem simplificar sua implementação no sistema, o que pode não ser a preferência do cliente, sendo necessário estabelecer novos requisitos e novas modificações no sistema, que causaria atraso na entrega e aumento de custo.

Como exemplo, utilizando-se do primeiro ponto dos requisitos funcionais apresentado acima. O membro da equipe, ao se deparar com o “pesquisar” apresentado no requisito, pode esperar que ao colocar o nome de um paciente a busca seja de todas as clínicas relacionadas ao sistema MHC-PMS, vendo todos os agendamentos. Porém, isso não está explícito no requisito. Os desenvolvedores podem interpretar que, primeiramente eles devem identificar a clínica e, em seguida, pesquisar o nome do paciente e os agendamentos. O segundo método necessitaria mais entradas de usuário e mais tempo gasto no projeto.

Assim, os Requisitos Funcionais são focados no que será realizado com o sistema. Como afirma Sommerville, deve ser completa e consciente.

2.2.2 Os Requisitos Não Funcionais

Os Requisitos Não Funcionais (RNF), por sua vez, como o nome sugere, diferenciam-se dos Requisitos Funcionais pois são requisitos que não estão diretamente relacionados com os serviços específicos oferecidos aos seus usuários pelo sistema. Para Sommerville,

> Eles podem estar relacionados às propriedades emergentes do sistema, como confiabilidade, tempo de resposta e ocupação de área. Uma alternativa a esse cenário seria os requisitos definirem restrições sobre a implementação do sistema, como as capacidades dos dispositivos de E/S ou as representações de dados usadas nas interfaces com outros sistemas. Os requisitos não funcionais, como desempenho,

> proteção ou disponibilidade, normalmente especificam ou restringem as características do sistema como um todo (SOMMERVILLE, 2011, p. 76).

Como também cita Cysneiros, a multiplicidade de um *software* é, em parte, determinada por sua funcionalidade, ou seja, "o que o sistema faz, e em parte por requisitos gerais que fazem parte do desenvolvimento do *software* como custo, performance, confiabilidade, manutenibilidade, portabilidade, custos operacionais entre outros" (CYSNEIROS, s.d, p. 18). Ele ainda ressalta a importância que os Requisitos Não Funcionais executam no sistema, tendo em vista que os RNFs desempenham um "papel crítico" durante o desenvolvimento de sistemas, e "erros devido a não elicitação ou a elicitação incorreta destes estão entre os mais caros e difíceis de corrigir, uma vez que um sistema tenha sido implementado" (CYSNEIROS, s.d, p. 19).

Para Sommervile, assim como para Cysneiros, os Requisitos Não Funcionais são regularmente mais delicados que Requisitos Funcionais individuais. Uma falha em um RNF pode acarretar na falha do sistema em completo, impossibilitando a utilização por parte do usuário.

> 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. Por exemplo, se um sistema de aeronaves não cumprir seus requisitos de confiabilidade, não será certificado como um sistema seguro para operar; se um sistema de controle embutido não atender aos requisitos de desempenho, as funções de controle não funcionarão corretamente (SOMMERVILLE, 2011, p. 76).

Os Requisitos Não Funcionais, então, podem influenciar a estrutura geral do sistema e não somente componentes individuais. Como exemplo,

> [...] para assegurar que sejam cumpridos os requisitos de desempenho, será necessário organizar o sistema para minimizar a comunicação entre os componentes. Um único requisito não funcional, tal como um requisito de proteção, pode gerar uma série de requisitos funcionais relacionados que definam os serviços necessários no novo sistema. Além disso, também podem gerar requisitos que restrinjam requisitos existentes (SOMMERVILLE, 2011, p. 76).

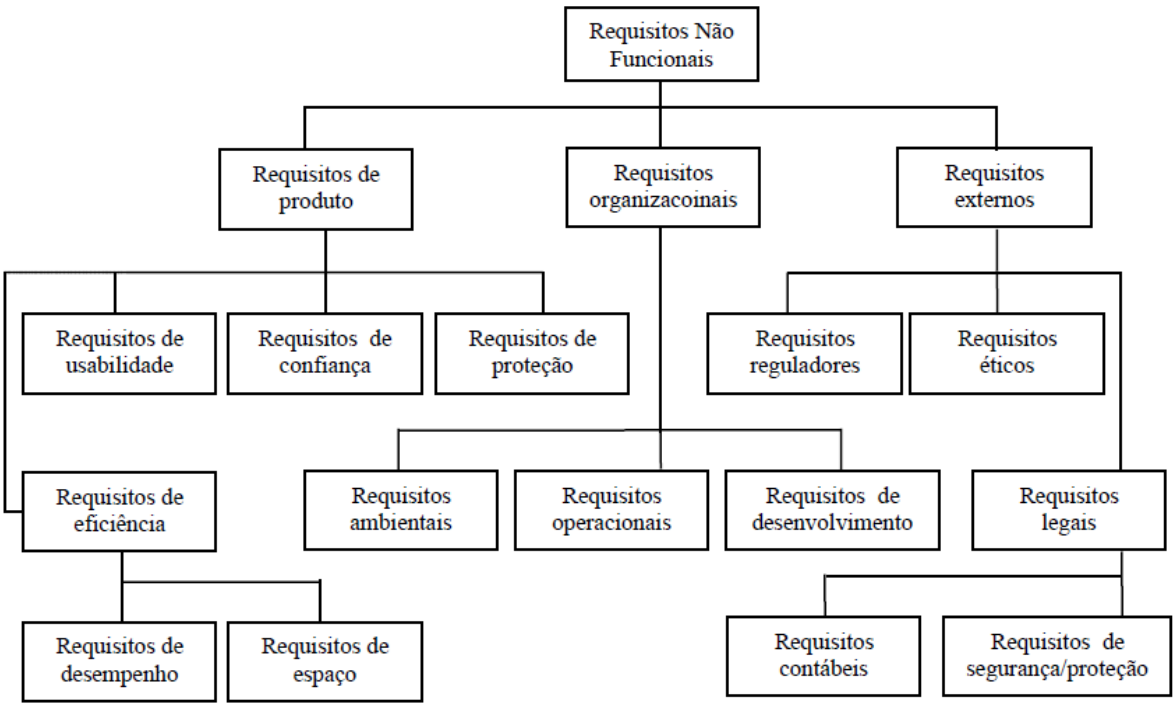
De acordo com Cysneiros, frequentemente, quando um RNF é adicionado a uma especificação de requisitos, outros RFNs poderão ser afetados, seja positiva ou negativamente, visualizado como interdependência entre os RNFs. Os RNFs são, então, de extrema importância para o desenvolvimento do *software*. Em um projeto, se por acaso algum Requisito Funcional não estiver implementado como deveria, o usuário pode ainda desenvolver um modo de substituir sua ausência. A ausência de um RNF, por sua vez, pode

comprometer o funcionamento de todo o sistema. Por serem geralmente subjetivos, os RNFs podem ser interpretados e definidos de formas variadas por diferentes pontos de vistas. É por essas razões, como afirma Cyneiros (s.d, p. 22), que os "RNFs são difíceis de se lidar e vitais de serem tratados para que possamos obter *softwares* de qualidade".

> Requisitos não funcionais vêm sendo citados em vários processos de desenvolvimento de software, dentre outras formas como restrições e condições de contorno, porém sempre de maneira quando muito secundária e altamente informal do ponto de vista da elicitação de requisitos. Este tipo de tratamento faz com que, quando tratados, estes requisitos sejam frequentemente contraditórios, difíceis de serem considerados durante o desenvolvimento de software e difíceis de serem validados. O fato destes requisitos terem sido mal elicitados ou não elicitados tem causado uma série de histórias de insucessos, incluindo a desativação de sistemas pouco após terem sido implantados (CYSNEIROS, s.d, p. 9).

Apresenta-se na figura 3 uma classificação de Requisitos Não Funcionais. De acordo com Sommerville (2011), a imagem mostra que os RNFs podem ser oriundos das características requeridas para o *software* (requisitos de produto), da organização que desenvolve o *software* (requisitos organizacionais) ou de fontes externas. Afirmando assim, que requisitos podem gerar outros, e que um equívoco em uma parte por interferir em toda a estrutura.

Figura 3 – Classificação de Requisitos Não Funcionais



Fonte: Sommerville (2011).

Os requisitos de produto categorizam ou delimitam o comportamento do *software.* Exemplos incluem “os requisitos de desempenho quanto à rapidez com que o sistema deve executar e quanta memória ele requer, os requisitos de confiabilidade que estabelecem a taxa aceitável de falhas, os requisitos de proteção e os requisitos de usabilidade” (SOMMERVILLE, 2011, p. 77).

Os requisitos organizacionais são mais gerais do sistema. Ele vem dos procedimentos de organização entre o cliente e o desenvolvedor. Exemplos dos requisitos organizacionais incluem os requisitos do processo operacional, que determinam o modo em que o sistema será utilizado, o ambiente de desenvolvimento ou normas de processo a serem usadas, os requisitos do processo de desenvolvimento que classificam a linguagem de programação, bem como os requisitos ambientais que caracterizam o ambiente operacional do sistema (SOMMERVILLE, 2011).

Os requisitos externos, por sua vez, abrangem todos os requisitos que descendem de fatores externos ao sistema e seu processo de desenvolvimento. Pode-se integrar requisitos reguladores, que determina o que deve ser executado para que o sistema seja admitido para uso, “por um regulador, tal como um banco central; requisitos legais, que devem ser seguidos para garantir que o sistema opere dentro da lei; e requisitos éticos, que asseguram que o sistema será aceitável para seus usuários e o público em geral” (SOMMERVILLE, 2011, p. 77).

Os Requisitos Não Funcionais podem surgir por vários meios: das necessidades dos usuários, necessidade de interoperabilidade com outros sistemas de *software* ou *hardware*, políticas organizacionais, devido a restrições de orçamento ou a partir de fatores externos, como legislações de privacidade ou regulamentos de segurança (SOMMERVILLE, 2011).

Um problema encontrado no desenvolvimento é que os Requisitos Não Funcionais recorrentemente costumam ser requisitados pelos usuários ou clientes como metas gerais, tendo em vista a facilidade de uso, da velocidade das respostas do usuário e da capacidade do sistema de se recuperar de possíveis falhas. “Metas estabelecem boas intenções, mas podem causar problemas para os desenvolvedores do sistema, uma vez que deixam margem para interpretação e, consequentemente, para disputas, quando da entrega do sistema” (SOMMERVILLE, 2011, p. 78).

> RNFs abordam importantes aspectos relacionados à qualidade de softwares. Eles são essenciais para que softwares sejam bem sucedidos. A não observância de RNFs pode resultar em: softwares com inconsistência e de baixa qualidade; clientes e desenvolvedores insatisfeitos; tempo e custo de desenvolvimento além dos previstos

> devido à necessidade de se consertar softwares que não foram desenvolvidos sob a ótica da utilização de RNFs (CYSNEIROS, s.d, p. 21).

Sommerville (2011) exemplificou uma meta de sistema típica de como um gerente pode expressar requisitos de usabilidade, ainda utilizando o sistema MHC-PMS como exemplo:

Figura 4 – Exemplificação geral do sistema MHC-PMS

O sistema deve ser de fácil uso pelo pessoal médico e deve ser organizado de tal maneira que os erros dos usuários sejam minimizados.

Fonte: Sommerville (2011).

Essa meta poderia ser expressa como um Requisitos Não Funcionais e "testável". Ao seguir essa descrição acima, é quase impossível determinar objetivamente qual a finalidade do sistema. Porém, na descrição abaixo também proposta por Sommerville (2011), já é possível incluir pelo menos a instrumentação de *software* para mostrar os erros cometidos pelos usuários quando estão testando o sistema:

Figura 5 – Exemplificação específica do sistema MHC-PMS

A equipe médica deve ser capaz de usar todas as funções do sistema após quatro horas de treinamento. Após esse treinamento, o número médio de erros cometidos por usuários experientes não deve exceder dois por hora de uso do sistema.

Fonte: Sommerville (2011).

Frequentemente acontecem interações - ou até mesmo conflitos - entre os Requisitos Não Funcionais e outros Requisitos Funcionais. Utilizando o exemplo proposto por Sommerville (2011), o requisito de autenticação na figura 6 a seguir, certamente requer a instalação de um leitor de cada cartão em cada computador conectado ao sistema, como também um outro requisito que necessite acesso móvel ao sistema pelos *laptops* dos médicos ou das enfermeiras. Nesse caso, esses não são normalmente utilizados leitores de cartão, sendo assim necessário utilizar outros modos.

Figura 6 – Exemplos de Requisitos Não Funcionais no MHC-PMS

Requisito de produto
O MHC-PMS deve estar disponível para todas as clínicas durante as horas normais de trabalho (segunda a sexta-feira, 8h30 às 17h30). Períodos de não operação dentro do horário normal de trabalho não podem exceder cinco segundos em um dia.
Requisito organizacional
Usuários do sistema MHC-PMS devem se autenticar com seus cartões de identificação da autoridade da saúde.
Requisito externo
O sistema deve implementar as disposições de privacidade dos pacientes, tal como estabelecido no HStan-03-2006-priv.

Fonte: Sommerville (2011).

Como apresenta Cysneiros, a crescente complexidade dos sistemas de *software* e das exigências de qualidade requisitadas pelos clientes incentiva a produção de *softwares* que abarquem tanto os aspectos funcionais, que são os comumente exigidos pelos clientes, e os não funcionais como: confiabilidade, custo, portabilidade, segurança, performance, entre outros. Esses aspectos devem ser implementados desde o início de todo o desenvolvimento do *software*. De acordo com Sommerville (2011, p. 78), "sempre que possível, os Requisitos Não Funcionais devem ser escritos quantitativamente, para que possam ser objetivamente testados".

> Na prática, os clientes de um sistema geralmente consideram difícil traduzir suas metas em requisitos mensuráveis. Para algumas metas, como manutenibilidade, não existem métricas que possam ser usadas. Em outros casos, mesmo quando a especificação quantitativa é possível, os clientes podem não ser capazes de relacionar suas necessidades com essas especificações. Eles não entendem o que significa um número definindo a confiabilidade necessária (por exemplo), em termos de sua experiência cotidiana com os sistemas computacionais. Além disso, o custo de verificar requisitos objetivamente não funcionais mensuráveis pode ser muito elevado, e os clientes, que pagam pelo sistema, podem não achar que os custos sejam justificados. (SOMMERVILLE, 2011, p. 78).

De acordo com Cysneiros, da perspectiva da engenharia de *software*, o processo de implementação de requisitos "é talvez a mais crucial parte do processo de desenvolvimento de *software* [...] quando só detectados depois do *software* implementado, erros em requisitos de *software* são até vinte (20) vezes mais caros de se corrigir que qualquer outro tipo de erro" (CYSNEIROS, s.d, p. 16).

Assim, os RNFs são, em suma, focados em como serão realizadas as funções do sistema, tornando-se de extrema importância para a funcionalidade no todo.

2.3 Modelagem de sistema

A modelagem de sistema é usada durante o processo de Engenharia de Requisitos para possibilitar a extração dos requisitos do sistema. “Durante o processo de projeto, são usados para descrever o sistema para os engenheiros que o implementam; e, após isso, são usados para documentar a estrutura e a operação do sistema” (SOMMERVILLE, 2011, p. 82) Como afirmam Cysneiros e Leite (s.d.), a modelagem tem por objetivo possibilitar uma noção preliminar de todos os dados do futuro sistema.

> Modelagem de sistema é o processo de desenvolvimento de modelos abstratos de um sistema, em que cada modelo apresenta uma visão ou perspectiva, diferente do sistema. A modelagem de sistema geralmente representa o sistema com algum tipo de notação gráfica, que, atualmente, quase sempre é baseada em notações de UML (SOMMERVILLE, 2011, p. 82).

De acordo com Sommerville, pode-se desenvolver dois tipos de modelos: os modelos do sistema existente e os modelos do novo sistema.

> 1. Modelos do sistema existente são usados durante a Engenharia de Requisitos. Eles ajudam a esclarecer o que o sistema existente faz e podem ser usados como ponto de partida para discutir seus pontos fortes e fracos. Levam, então, os requisitos para o novo sistema;
> 2. Modelos do novo sistema são usados durante a Engenharia de Requisitos para ajudar a explicar os requisitos propostos para outros *stakeholders*[2] do sistema. Os engenheiros usam esses modelos para discutir propostas de projeto e documentar o sistema para a implementação. Em um processo de engenharia dirigida a modelos, é possível gerar uma implementação completa ou parcial do sistema a partir do modelo de sistema (SOMMERVILLE, 2011, p. 82).

O modelo, então, é a abstração do sistema a ser elaborado e estudado. Ele deve, idealmente, ser uma execução de todas as informações de um sistema da entidade representada. “O aspecto mais importante de um modelo de sistema é que ele deixa de fora os detalhes [...] uma abstração deliberadamente simplifica e seleciona as características mais salientes” (SOMMERVILLE, 2011, p. 83). Para Sakurada e Miyake (2009, p. 27) o processo de modelagem deve “inicialmente buscar uma clara compreensão da estrutura e dinâmica do sistema real a ser simulado e somente então avançar para a derivação dos procedimentos experimentais que possibilitarão analisar seu comportamento”. Em suma, permite um entendimento da estrutura do sistema e possibilita uma compreensão das formas em que o sistema é ordenado.

[2] *Stakeholders* é definido como qualquer organização ou indivíduo que, de algum modo, é afetado pelas ações de uma determinada empresa. Em uma tradução livre para o português, o termo significa “parte interessada”.

Utilizando-se como base a modelagem de Sommerville (2011), afirma que se pode desenvolver vários modelos para representar o sistema a partir de formas diferentes. Vários tipos de perspectivas podem ser desenvolvidos para a modelagem de sistema, porém tiramos como exemplo quatro (4) tipos principais:

1. Uma perspectiva externa, em que o contexto ou o ambiente do sistema é modelado;
2. Uma perspectiva de interação, em que as interações entre um sistema e seu ambiente, ou entre os componentes de um sistema são modelados;
3. Uma perspectiva estrutural, em que a organização de um sistema ou a estrutura dos dados processados pelo sistema são modelados;
4. Uma perspectiva comportamental, em que o comportamento dinâmico do sistema e como ele reage aos eventos são modelados.

De acordo com Sommerville (2011, p. 83), "os UML têm muitos tipos de diagramas e, dessa forma, apoia a criação de muitos tipos de diferentes modelos de sistema". No entanto, ele afirma que cinco (5) tipos de diagramas podem representar a essência de um sistema:

1. Diagramas de atividades: mostram as atividades envolvidas em um processo, seja ele no processamento de dados ou não;
2. Diagramas de casos de uso: mostram as interações entre um ambiente e seu sistema;
3. Diagramas de sequência: revelam as interações entre os atores e o componente e seus sistemas;
4. Diagramas de classe: mostram as classes de objeto no sistema e as associações entre elas;
5. Diagramas de estado: mostram como o sistema reage de acordo com os eventos internos e externos.

A modelagem será utilizada durante o trabalho justamente por sua capacidade de proporcionar uma demonstração abrangente do sistema de formas diversas, possibilitando assim uma abstração do sistema que será elaborado e estudado.

2.4 *NFR-Framework*

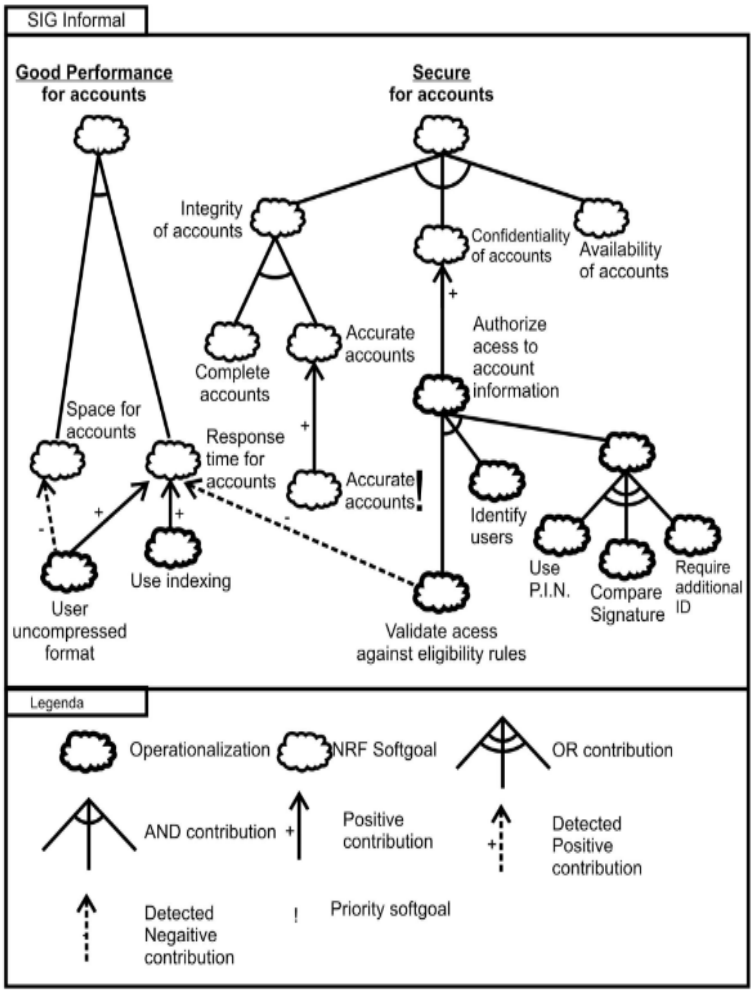
Como citado por Xavier (2009), os estudiosos Mylopoulos, Nixon e Chung desenvolveram alguns trabalhos sobre RNF. Esses trabalhos, por fim, resultaram em um

framework, onde os Requisitos Não Funcionais são “tratados como metas possivelmente conflitantes, devendo ser identificados em sua forma mais geral e refinados até que se chegue a um conjunto de requisitos que satisfaçam ao requisito geral” (XAVIER, 2009, p. 39). Assim, o *NFR-Framework* dispõe de uma estrutura para representar e registrar o processo de projeto e raciocínio em grafos, chamados de *Softgoal Interdependency Graph* (SIG).

Para Cysneiros, o *framework* apresentado por Chung em seus três trabalhos (1993, 1995, 2000) é “a abordagem mais completa até o momento e procura tratar de todos os tipos de Requisitos Não Funcionais desde as primeiras etapas do processo de desenvolvimento de *software*” (CYSNEIROS, s.d. p.11).

Como apresentado inicialmente, esse conjunto de estratégias pode ser visualizado em termos de construção incremental e interativa, elaboração, análise e revisão do SIG. Os *softgoals* são mostrados como nuvens, e os principais são mostrados no topo de um gráfico. Eles são conectados por *links* de interdependência, que são mostrados como linhas, geralmente com pontas de seta. Eles têm rótulos associados que são usados para ajudar o processo de raciocínio durante o projeto.

Figura 7 – Exemplo de SIG.



Fonte: Chung *et al.* (2000)

Para Xavier (2009, p. 40), "*softgoals* são todos os objetivos não funcionais que são difíceis de ser avaliados". Para determinar se os *softgoals* foram alcançados, um procedimento de avaliação (algoritmo de rotulagem) é usado, que considera rótulos e contribuições e, mais importante, decisões do desenvolvedor. É importante observar que o desenvolvedor tem controle sobre quais metas flexíveis são declaradas, como e até que ponto elas são refinadas.

A escolha na utilização da *NFR-Framework* foi a capacidade de prever conflitos de interesse de *software*. Em outras palavras, quando um requisito interfere no outro. Com isso, a ferramenta SIGSoft pretende utilizar-se desse grande ponto positivo como alicerce, realizando assim uma ferramenta mais eficiente e intuitiva.

2.5 Ferramentas destinadas a modelagem das RNFs existentes no mercado

Duas ferramentas foram utilizadas para o desenvolvimento do trabalho: A Star-UML e a NDR-Tool. A seguir, foi apresentado tais ferramentas de modelagem de RNFs, juntamente com seus pontos negativos e positivos, utilizados como embasamento no desenvolvimento da ferramenta proposta no trabalho. A NDR-Tool, por sua vez, não será utilizada ativamente no trabalho devido a perca do suporte por parte dos desenvolvedores, mas sim como fonte de comparação para o desenvolvimento da nova ferramenta proposta.

2.5.1 *Start-UML* e *NDR-Tool*

A Star-UML é uma ferramenta disponível em várias plataformas (incluindo *MacOS, Windows* e *Linux*) para criar diagramas em UML. Ela permite projetar modelos rápido e facilmente no padrão UML, disponibilizando recursos como a abordagem básica dos diagramas, reutilização de modelos, entre outros.

Neste trabalho foi utilizado a versão 1.0 da ferramenta Star-UML que, diferente das versões mais atuais, somente está disponível para a plataforma *Windows.* Faz-se necessário o uso da versão mais antiga pois utilizaremos um *plug-in*[3] compatível somente com a versão 1.0, o NDR-Tool.

A NDR-Tool era uma extensão que pode ser utilizada na ferramenta Star-UML/RE-Tool Designer. A ferramenta disponibilizava ao usuário a montagem de gráfico SIG partindo de catálogos que já estão dispostos na ferramenta. Ela dispõe de catálogos de RNFs para consulta por categorias (como usabilidade e segurança), disponível somente para *desktop*. Dessa forma, o engenheiro de *software* realiza buscas nos catálogos, integra os requisitos identificados na busca à modelagem atual e, em seguida, incluir em sua modelagem novos NRFs, estando ciente das possíveis implicações que as novas inclusões podem acarretar no *software*.

Pontos negativos da NDR-Tool era a falta do armazenamento de catálogos e a incompatibilidade com outros sistemas operacionais. Desse modo, a SIGSoft disponibiliza um armazenamento em nuvem, do modo em que todos os catálogos ficam salvos no servidor da aplicação, ou seja, não é preciso está com o projeto no seu computador para acessá-los, diferente do NDR-Tool. Ela também possui compatibilidade em todos os sistemas operacionais, pois só dependem do navegador. Diferente do NDR-Tool, que só funciona em uma versão especifica do Start-UML no *Windows*.

[3] *Plug-in* é usado para acrescentar funções em outros programas. Como o nome sugere, um *plug-in* serve de encaixe: ele adiciona recursos aos *softwares* principais.

3 TRABALHOS RELACIONADOS

Existe inúmeros trabalhos na literatura acadêmica que retratam o problema de modelagem de RNFs utilizando diversas abordagens. Parte significativa desses trabalhos decorrem sobre a forma de como realizar o tratamento da modelagem dos RNFs, assim como, ferramentas que servem para auxiliar a validação dos RNFs e da melhor visualização do problema modelado.

3.1 Um processo de validação de requisitos não funcionais baseado no *NFR-Framework*

O trabalho proposto por Anselmo de Araujo Couto e Luiz Eduardo Galvão Martins, de 2009, decorre sobre a importância dos RNFs como autenticação do procedimento da implementação de um *software*, para isso é apresentado um processo de validação de requisitos baseado no *NFR-Framework*. O objetivo geral exposto é a disponibilização de um processo de validação de documento dos RNFs, buscando um aperfeiçoamento destes e uma consequente melhora de qualidade na documentação de requisitos original. Ele se desdobra explicando os conceitos de validação de um requisito e em seguida explora mais a fundo os processos presentes no *NFR-Framework*. Com isso, ele apresenta a proposta desse processo. As etapas principais desse processo são:

- Revisar RNFs Originais;
- Construir o Grafo SIG dos RNFs Originais;
- Reorganizar os RNF Originais Revisados;
- Produzir o Grafo SIG a partir da Reorganização;
- Elaborar Quadro Comparativo (quantitativos por tipos de *softgoal* e relacionamentos);
- Avaliar a Evolução dos RNFs (Originais) frente aos RNFs propostos a partir do *NFR- Framework*.

Com o processo definido os autores aplicam um estudo de caso, comparando cada etapa, verificando métricas qualitativas e quantitativas sobre a elaboração dos RNFs. O resultado mostrou que o processo sistemático proposto detectou algumas variantes entre o documento de RNFs, como omissões e defeitos presentes na elaboração deles a partir do processo proposto.

3.2 Método e linguagem para modelagem gráfica de requisitos de *software* e sistemas

O trabalho proposto por Paulo José Dantas Novaes, de 2019, é apresentado um método e uma linguagem de modelagem gráfica de requisitos de *software* e sistemas denominada RIMON (*Requirements and Interdependencies Modeling Notation*). A motivação de sua implementação foi realizar a representação de requisitos e suas interdependências de maneira sistemática com precisão e expressividade.

O objetivo principal é auxiliar na melhoria da qualidade da especificação de requisitos de *softwares.* A estratégia foi inspirada a partir de uma abordagem de elaboração de requisitos orientadas a notificações, mas com uma adaptação de cunho a atrair usos comerciais da ferramenta desenvolvida. O método proposto consiste em um ciclo interativo de atividades que vão desde a identificação e análise dos dados de entrada até a modelagem gráfica dos requisitos e suas interdependências.

A RIMON oferece características de suporte a modelagem dos Requisitos Funcionais e Não Funcionais do sistema ou do *software*, entidades, atributos, pré-condições, pós-condições e identificação de conflitos. Seu diferencial central é a modelagem de pré-condições dos requisitos. Isso possibilita representar restrições de relações a RNFs por meio condições e inferências lógicas.

O método é aplicado em três (3) experimentos de modelagem, tanto de requisitos de sistemas quanto de requisitos de *software*, para aprovar as capacidades que essa linguagem alcançou com resultados satisfatórios. O trabalho apresenta uma exploração vasta de comparações da abordagem proposta com a literatura acadêmica, identificando os pontos a serem absorvidos tanto na academia quanto na indústria para utilização no levantamento de requisitos de *software*.

3.3 Gestão de riscos em projetos de software: Uma abordagem baseada em Requisitos Não Funcionais

Por fim, o trabalho de Ana Cristina Da Silva Andrade, José Luis Braga, André Luiz De Castro Leal e Fernando Hadad Zaidan, de 2019, apresenta um estudo voltado a uma gestão de riscos em projetos de *software* com mais aplicabilidade utilizando uma abordagem baseada em RNFs, com o uso do *NFR-Framework*.

O artigo escoa com uma revisão de literatura sobre os desafios da gestão de riscos

e nas relações diretas que são percebidas em artigos dessa vertente com uma boa elaboração de Requisitos Não Funcionais. A seguir detalha melhor os RNFs assim como as ferramentas da sua modelagem como *NFR-Framework* e *Framework i**.

O método de implementação da gestão de riscos baseada em RNFs é exposto por meio da verificação das atribuições que cada área alcança com uma comparação dos *softgoals* utilizada para elaborar requisitos e a lista de fatores de risco, artefato muito disseminado na área de engenharia de *software*.

O resultado a partir disso é a elaboração de um SIG de riscos, que é produzido para representação dos principais riscos de um projeto de *software*, mantendo a ideia do gráfico de interdependência dos *softgoals* com entidades representativas dos riscos presentes na tomada de decisão sobre as questões inerentes ao *software* em questão.

A avaliação feita é que o SIG de riscos auxilia para facilitar o seu entendimento e aplicação prática em projetos. Os aspectos identificados são agrupados em um catálogo dinâmico e representa o primeiro passo para elaboração de um catálogo mais completo. O SIG de riscos permite uma forma de validar os requisitos de riscos através de uma análise de metas mais flexíveis.

Concluiu-se que os modelos e exemplos alcançados contribuem para os gestores de projetos identificarem e gerenciarem os riscos do projeto em fase inicial, alertando problemas de infraestrutura, possibilitando uma ação de controle.

3.4 Comparação entre os trabalhos relacionados e o proposto

O quadro 1 evidencia as diferenças e semelhanças entre os três trabalhos relacionados apresentados anteriormente e o atual trabalho proposto nesse projeto, ilustrando sobre a elaboração da ferramenta, o ambiente de execução, a quem foi destinado e a proposta principal.

Quadro 1 – Comparações deste trabalho com os trabalhos relacionados

Trabalhos	Elaboração de ferramenta	Ambiente de execução	A quem se destina	Proposta principal
Trabalho proposto	Sim	Web	Gerentes de projeto e equipes de desenvolvimento;	Criação de ferramenta web para modelagem de RNFs através de SIGs inspirado no NFR-Framework;
(COUTO; MARTINS, 2009)	Não	-	Gerentes de projeto e equipes de desenvolvimento;	Nova abordagem no processo de validação dos RNFs utilizando NFR-Framework;
(NOVAES *et al.*, 2019)	Sim	Desktop	Engenheiros de *software* e pesquisadores;	Criação de uma linguagem e ferramenta lógico-matemática para elicitação de RNFs através de um sistema orientadas a notificação;
(ANDRADE *et al.*, 2019)	Não	-	Gerentes de projeto e equipes de desenvolvimento.	Utilização de frameworks de modelagem de RNFs para criação de um SIG na gestão de riscos de *software.*

Fonte: Elaborada pelo autor (2020).

4 METODOLOGIA

O presente trabalho apresenta uma avaliação do cenário atual referente as ferramentas de modelagem de RNFs, realizando uma análise entre modelos de ferramentas já existentes, expondo as caraterísticas positivas e negativas, sugerindo uma possível solução das lacunas ou equívocos que nelas podem ser identificadas. Com isso, desenvolveu-se um *software* para auxiliar um projetista a modelar seus Requisitos Não Funcionais, ajudando-os a prever qual e que tipo de influência eles poderão gerar entre si, utilizando-se o *NFR-Framework* como alicerce. Dessa forma, utilizou-se da pesquisa de natureza descritiva, com a abordagem quali-quantitativa.

A partir de tais objetivos, juntamente com a elaboração da ferramenta, foram feitos períodos de testes com estudantes de cursos da área de TI (tecnologia de informação), para desenvolver de forma eficiente a ferramenta proposta, tendo em vista que deteriam maior conhecimento referente aos RNFs, utilizando de forma proveitosa a ferramenta proposta.

O questionário apresenta treze (13) questões objetivas, algumas de caráter pessoal para definir os participantes do teste, e outras direcionadas ao uso da SIGSoft 1.0. A pesquisa foi realizada com vinte (20) alunos. Dessa forma, possibilitou que os alunos apresentassem uma avaliação acerca da ferramenta proposta.

Em suma, o objetivo geral da monografia foi desenvolver uma ferramenta para auxiliar um projetista de *software* a modelar seus RNFs, ajudando-os a prever se haverá influências entre eles durante o processo de modelagem. Dessa forma, pretendeu-se solucionar os principais questionamentos existentes em ferramentas envolvendo os RNFs e fornecer uma biblioteca de modelos para reutilização.

4.1 Coleta de dados

A pesquisa de campo foi elaborada por meio de um questionário estruturado que funciona como uma espécie de *feedback* direto da ferramenta proposta. As perguntas são relacionadas a funcionalidade geral da ferramenta, juntamente com a experiência prévia que os participantes tinham com modelagem de requisitos. Dessa forma, o público alvo do estudo é formada por alunos e/ou profissionais de cursos da área de TI (tecnologia de informação). Uma amostra da população foi utilizada para a análise da pesquisa, escolhidos de forma aleatória.

O questionário apresenta treze (13) questões objetivas, aplicado através da *internet* por meio do *site Google Docs*[4] durante o período de março à maio de 2022, disponibilizado em um grupo de alunos da Universidade Federal do Ceará (UFC) por meio da rede social *facebook*. Um grupo de vinte (20) estudantes se disponibilizaram a participar de forma anônima do projeto. O único requisito estabelecido para a participação no questionário era que os entrevistados ainda estivessem em período de graduação em um curso de TI.

As perguntas elaboradas foram de caráter identificatório e específicas, sendo quatro (4) delas para definir os participantes (idade, gênero, curso e semestre) e outras nove (9) relacionadas a pesquisa (experiência profissional no desenvolvimento de *softwares*, utiliza/ou alguma ferramenta de modelagem de *software*, frequência você utiliza essa ferramenta, conhece alguma ferramenta de modelagem de *software goals*, o que você achou da proposta da ferramenta SIGSoft, se contribui para identificação de conflito de Requisitos Não Funcionais, se é ferramenta fácil de utilizar, se confia no sistema de alerta de conflitos e se utilizaria futuramente).

As respostas enviadas foram apresentadas por meio de gráficos construídos no próprio *Google Docs*, apresentados aqui para melhor visualização.

[4] *Google Docs*, plataforma do Google que permite a elaboração de documentos, planilhas, apresentações e formulários. Foi escolhido a utilização dessa plataforma pela praticidade e facilidade de envio aos possíveis participantes do questionário, assim como a rapidez e segurança da coleta dos resultados.

5 SIGSoft 1.0

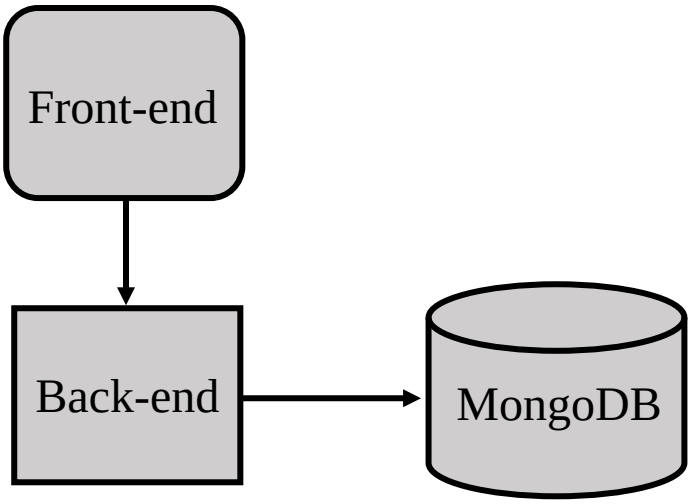
A SIGSoft 1.0, como já afirmado ao longo do trabalho, auxilia um projetista a modelar seus Requisitos Não Funcionais, ajudando-os a prever qual e que tipo de influência haverá entre os modelos.

5.1 Estrutura do sistema

A construção da ferramenta se deu através do código que se encontra na página do GitHub, sistema *online* para hospedagem de repositórios Git. Os *links* de acesso são: https://github.com/andreconjo/sig-soft, para a *front-end;* e https://github.com/andreconjo/sigsoft-api, para a *back-end*.

O sistema foi projetado para ter uma estrutura simples. A figura 8 representa o processo que a ferramenta desenvolve durante sua utilização. A ferramenta pode ser acessada por um computador *desktop* comum, por meio de uma página *web.*

Figura 8 – representação ilustrativa da arquitetura da ferramenta.



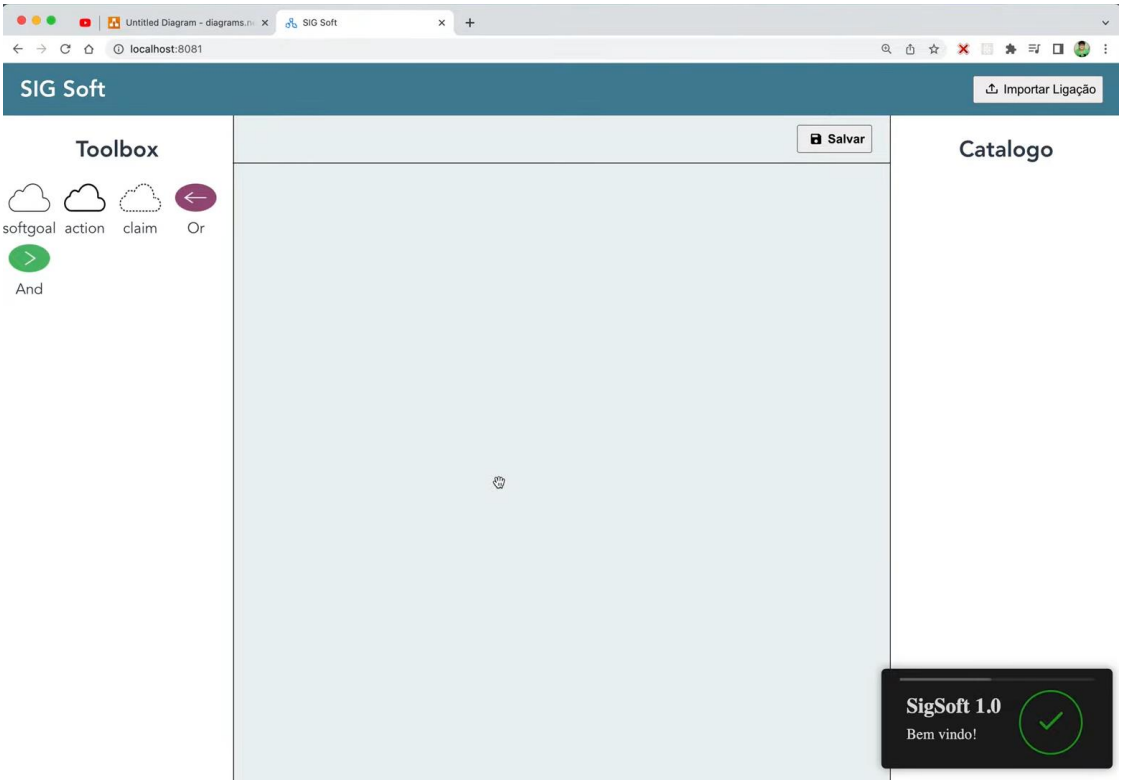
Fonte: elaborado pelo autor (2022).

O aplicativo foi desenvolvido utilizando a linguagem Java, e utiliza um banco de dados não relacional (MongoDB). De acordo com a ação requerida no *front-end* (a interface gráfica do site), o *back-end* (operação do sistema) se comunica com o MongoDB para salvar ou requisitar catálogos armazenados.

5.2 A ferramenta

Ela dispõe na barra lateral o *toolbox* com as opções *Softgoal*, *Action* e *Claim*: *Softgoal* representa o objetivo a ser alcançado na aplicação; *Action* é o que deve ser feito para implementar o *Softgoal*. Os itens da barra lateral podem ser arrastados para a área de trabalho.

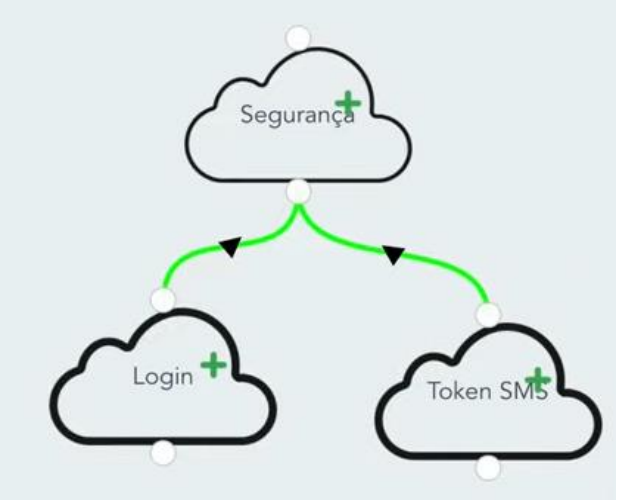
Figura 9 – *Front-end*.



Fonte: elaborado pelo autor (2022).

É possível fazer dois (2) tipos de ligações: A ligação *Or* e a *And*: a ligação *Or* (*ou*, em tradução livre do inglês), determina que pelo menos um dos *Actions* devem ser implementados; já a *And* (*e*, tradução livre do inglês), determina que todas as ligações devem ser implementadas para serem consideradas satisfeitas. Por exemplo, afim de implementar um *Softgoal* de Segurança, seria preciso implementar Login e Token SMS, dois (2) *Actions* com a ligação *And*. Caso os dois alcancem o objetivo, um símbolo positivo ao lado do *Softgoal* trabalhado é visível, como apresentado na figura 9. Caso não, um xis em vermelho seria visível.

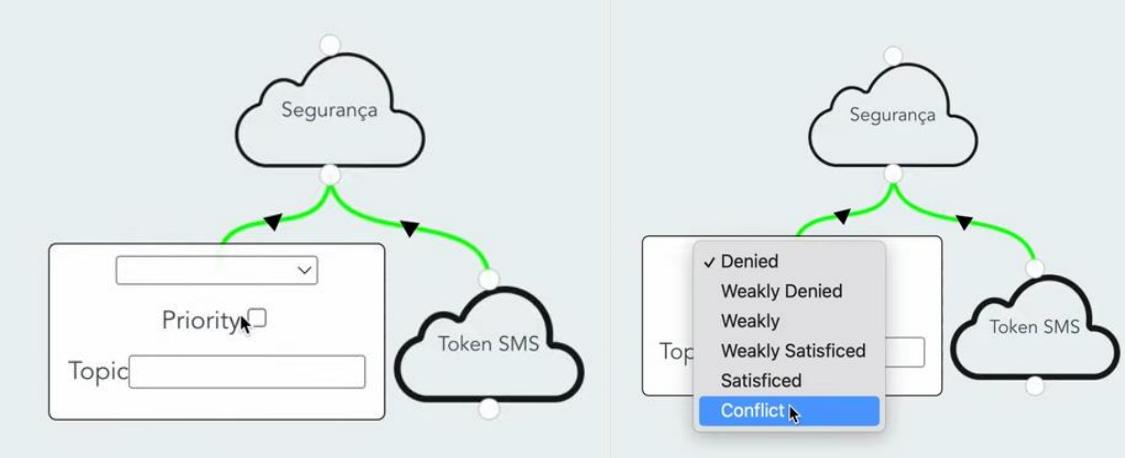
Figura 10 – *Softgoal* de Segurança com ligação *And*.



Fonte: elaborado pelo autor (2022).

É possível, manualmente, definir o nível de desenvolvimento. Ele pode se encontrar *denied*, *weakly denied*, *weakly*, *weakly satisficed*, *satisficed* e *conflict*. Também é possível definir o tópico e se ele se enquadra como prioridade.

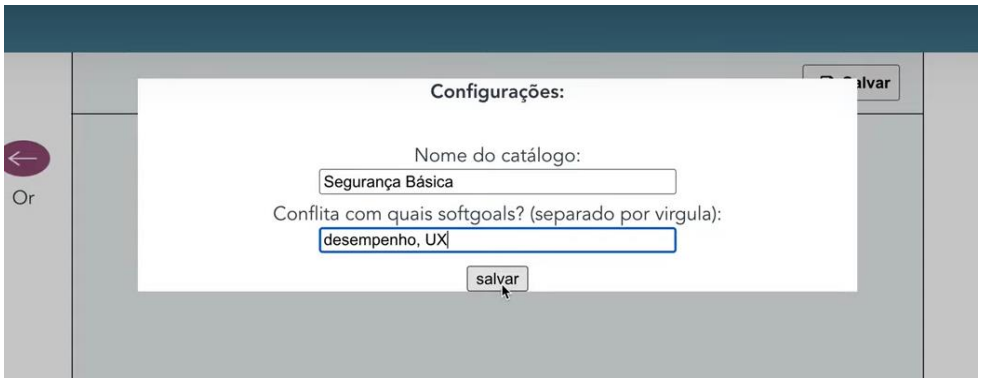
Figura 11 – Níveis de desenvolvimento.



Fonte: elaborado pelo autor (2022).

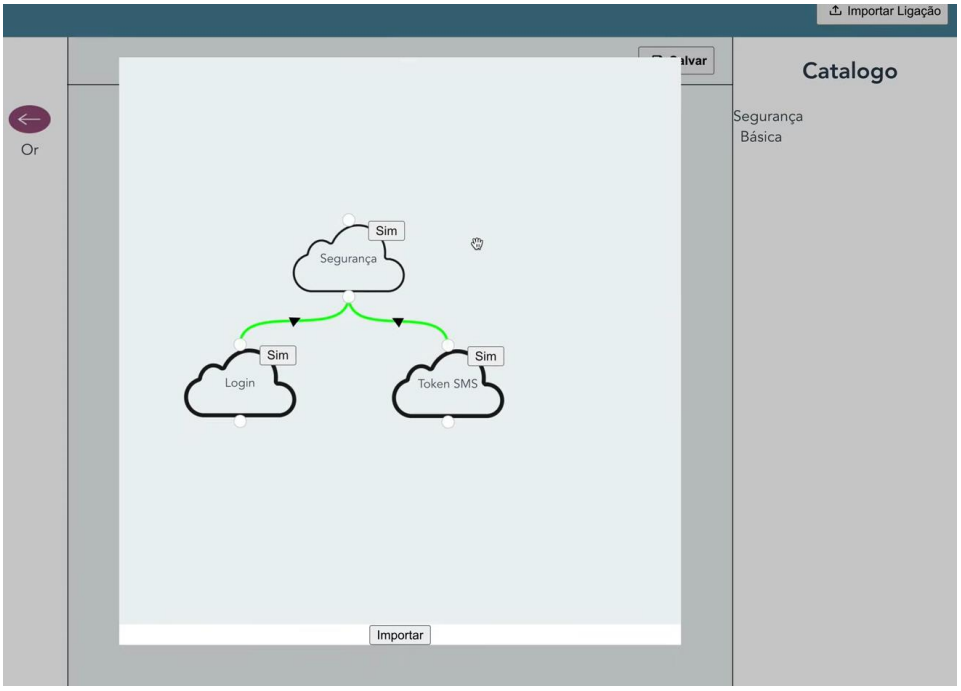
A SIGSoft 1.0 disponibiliza um armazenamento em nuvem, do modo em que todos os catálogos ficam salvos no servidor da aplicação, ou seja, não é preciso está com o projeto no seu computador para acessá-los, diferente do NDR-Tool. Ao salvar o catálogo, ele será publicado, possibilitando a importação do mesmo para outros usuários. É possível descrever o catálogo com informações acerca do projeto: tipo dos *Softgoals*¸ quais e de que modos foram afetados, entre outros.

Figura 12 – Salvamento do *softgoal*.



Fonte: elaborado pelo autor (2022).

Figura 13 – Catálogo.



Fonte: elaborado pelo autor (2022).

Caso haja interferência direta entre catálogos importados, um aviso é visível na tela. Por exemplo, ao adicionar outro objetivo, como Desempenho, em um projeto com o *Softgoal* de Segurança, apareceria um aviso sobre o conflito gerado entre os dois no canto inferior direito.

Figura 14 – Aviso de conflito.

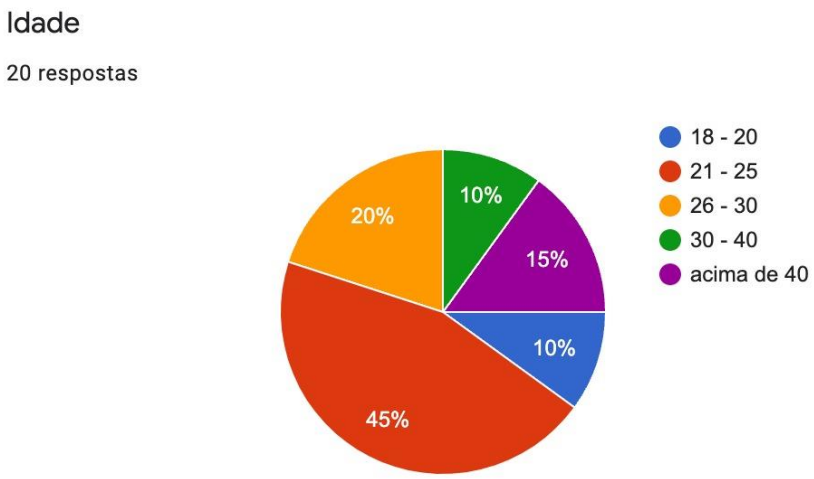
Fonte: elaborado pelo autor (2022).

Ela também possui compatibilidade em todos os sistemas operacionais, pois só depende do navegador. Diferente do NDR-Tool, que só funciona em uma versão especifica do Start-UML no *Windows*.

5.3 O questionário em prática

A primeira pergunta é referente a idade dos entrevistados. O gráfico 1 apresenta que 45% das respostas, a grande maioria, se encontra entre 21 e 25 anos, seguido por 20% entre 26 e 30 anos. 15% estão acima de 40 anos, enquanto 10% dos entrevistados estão entre 18 e 20 anos e também entre 30 e 40 anos. Embora a maioria dos estudantes sejam pessoas jovens adultas, há também uma gama de profissionais mais velhos na área.

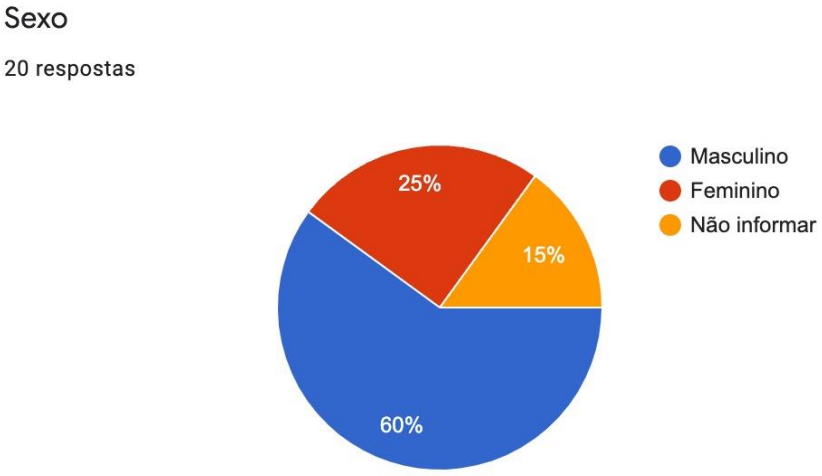
Gráfico 1 – Percentual dos entrevistados de acordo com a pergunta 01.



Fonte: dados da pesquisa.

A segunda pergunta é referente ao gênero dos entrevistados. O gráfico 2 apresenta que 60% dos participantes se identificam no gênero masculino, enquanto 25% se identificam com o gênero feminino. 15% preferem não informar.

Gráfico 2 – Percentual dos entrevistados de acordo com a pergunta 02.



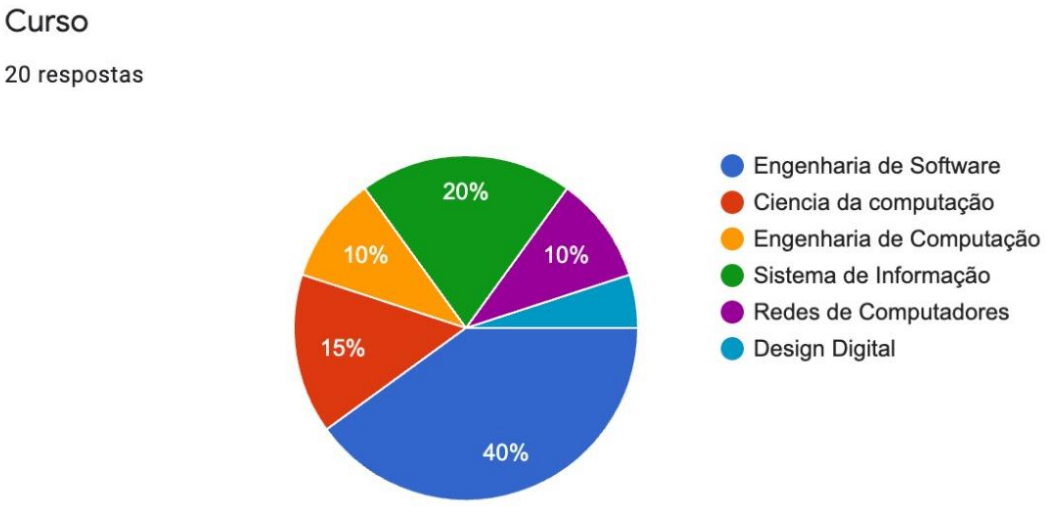
Fonte: dados da pesquisa

A terceira pergunta, apresentada no gráfico 3, é referente ao curso dos entrevistados: 40% fazem engenharia de *software;* 20% fazem sistemas de computação; 15% ciências da computação; 10% engenharia de computação; 10% redes de computadores; e 5% fazem design digital.

A quarta pergunta, apresentada no gráfico 4, é sobre o semestre em que os

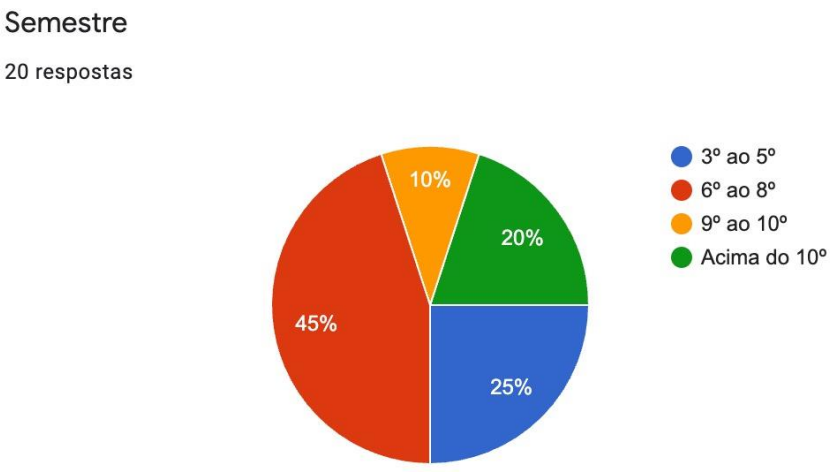
entrevistados estavam durante o período da pesquisa: 45% se encontram entre o 6º ao 8º semestre; 25% entre o 3º ao 5º semestre; 20% acima do 10º semestre; e 10% se encontra entre o 9º e o 10º semestre.

Gráfico 3 – Percentual dos entrevistados de acordo com a pergunta 03.



Fonte: dados da pesquisa.

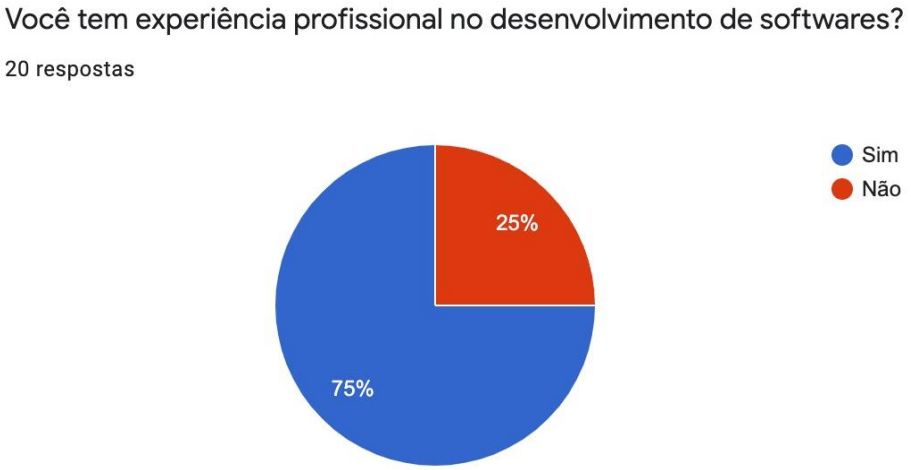
Gráfico 4 – Percentual dos entrevistados de acordo com a pergunta 04.



Fonte: dados da pesquisa.

A quinta pergunta, apresentada no gráfico 5, é relacionada a experiência profissional dos entrevistados no desenvolvimento de *softwares*: 75% das respostas, ou seja, a maioria dos entrevistados tem experiência profissional em desenvolvimento de *softwares*, enquanto 25% não.

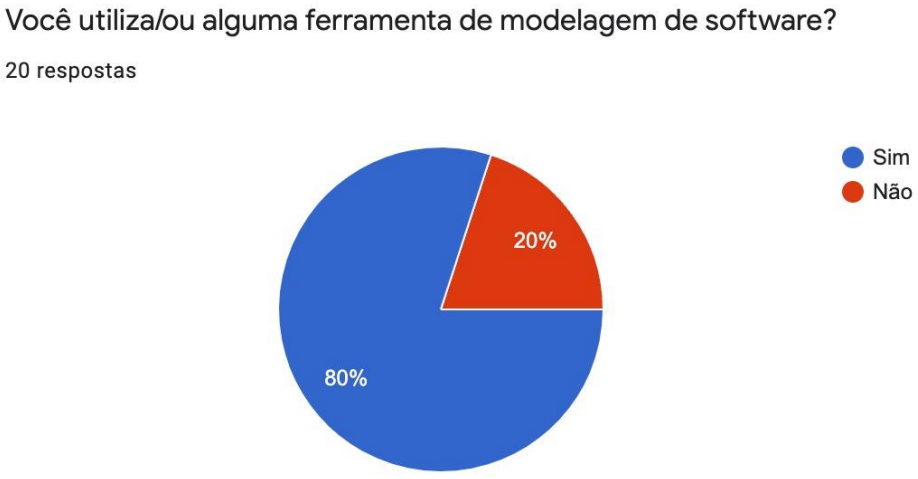
Gráfico 5 – Percentual dos entrevistados de acordo com a pergunta 05.



Fonte: dados da pesquisa.

A sexta pergunta, apresentada no gráfico 6, é sobre o uso prévio de ferramentas de modelagem de *software* dos entrevistados: 80% tinham utilizado previamente ferramentas de modelagem, enquanto 20% não.

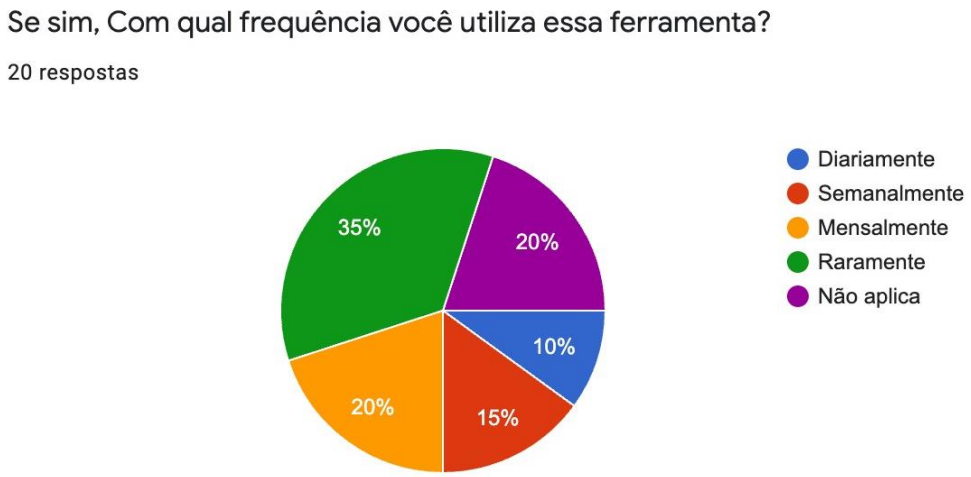
Gráfico 6 – Percentual dos entrevistados de acordo com a pergunta 06.



Fonte: dados da pesquisa.

A sétima pergunta, apresentada no gráfico 7, é referente a frequência de uso de ferramentas de modelagem de *software* dos entrevistados: 35% afirmaram que utilizam raramente uma ferramenta; 20% utilizam mensalmente; 15% afirmam que utilizam semanalmente; 10% utilizam diariamente; e 20% dos entrevistados não utilizam.

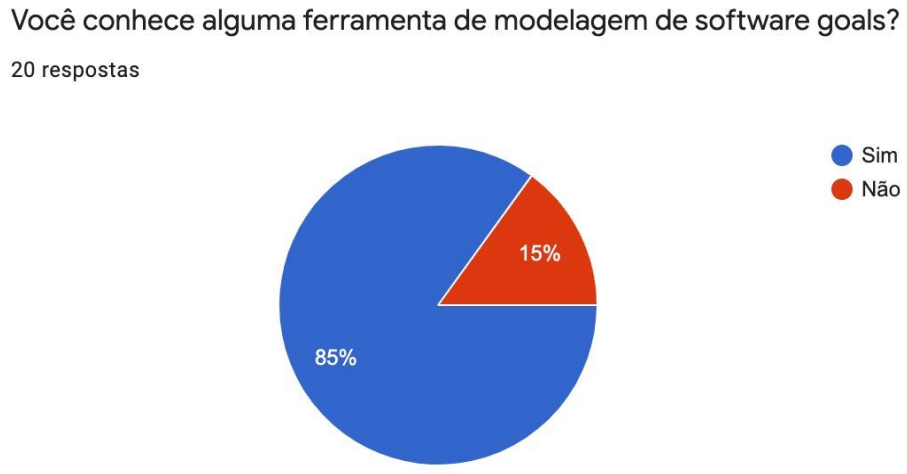
Gráfico 7 – Percentual dos entrevistados de acordo com a pergunta 07.



Fonte: dados da pesquisa.

A oitava pergunta, apresentada no gráfico 8, é se os entrevistados conheciam alguma ferramenta de modelagem de *software goals*: 85% dos entrevistados conheciam, enquanto 15% não.

Gráfico 8 – Percentual dos entrevistados de acordo com a pergunta 08.



Fonte: dados da pesquisa.

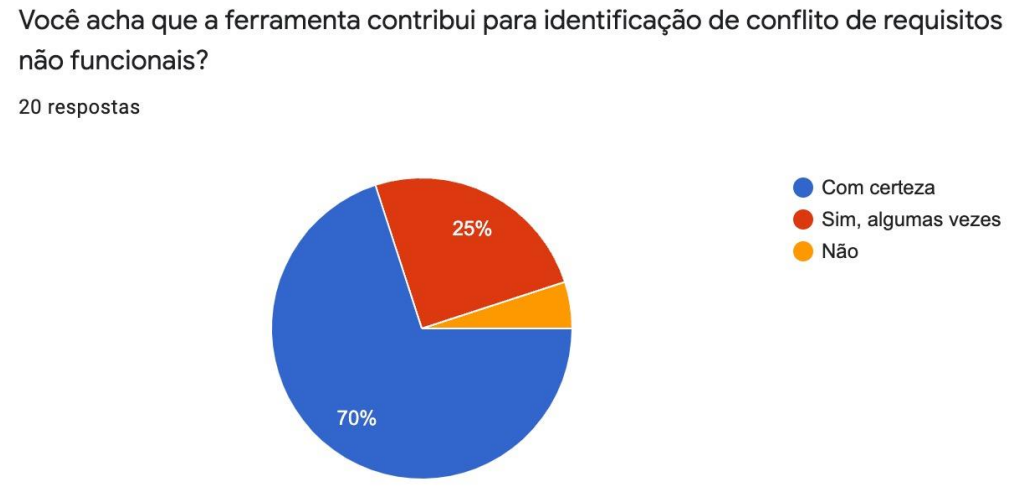
A nona pergunta é relacionada ao que os entrevistados acharam da proposta da ferramenta SIGSoft. De acordo com as respostas apresentada no gráfico 9, 65% dos entrevistados acharam a proposta da ferramenta muito boa, enquanto 25% acham boa. 10% das respostas acharam ruim.

Gráfico 9 – Percentual dos entrevistados de acordo com a pergunta 09.

Fonte: dados da pesquisa.

A décima pergunta, apresentada no gráfico 10, é se os entrevistados acharam que a ferramenta contribui para a identificação de conflito de RNFs: 70% acreditam sim; 25% acreditam que a ferramenta contribui algumas vezes; e 5% não acreditam que ele contribui para a identificação de conflitos.

Gráfico 10 – Percentual dos entrevistados de acordo com a pergunta 10.

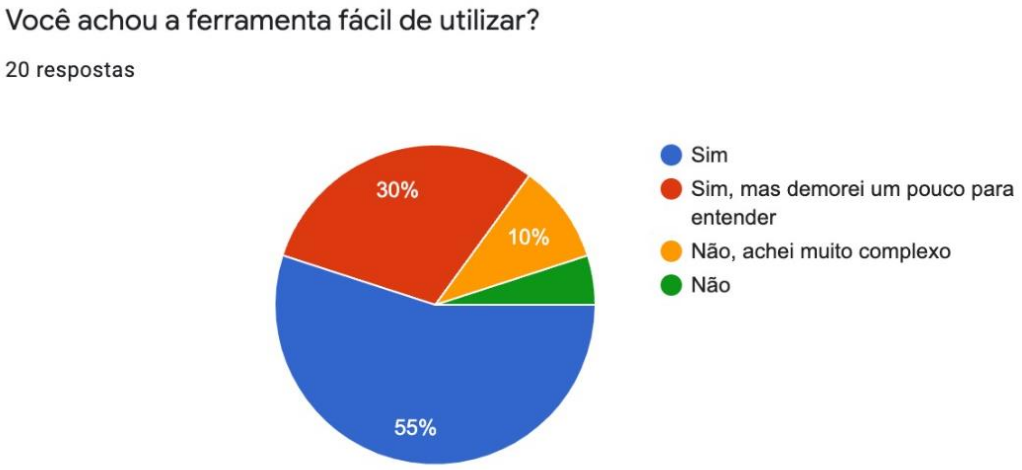


Fonte: dados da pesquisa.

A décima primeira pergunta é se os entrevistados acharam a ferramenta de fácil utilização. Como apresentada no gráfico 11, 55% acharam a ferramenta fácil, enquanto 30%

acharam boa, porém tiveram uma dificuldade para entender. 10% acharam muito complexa, enquanto 5% não acharam fácil de usar.

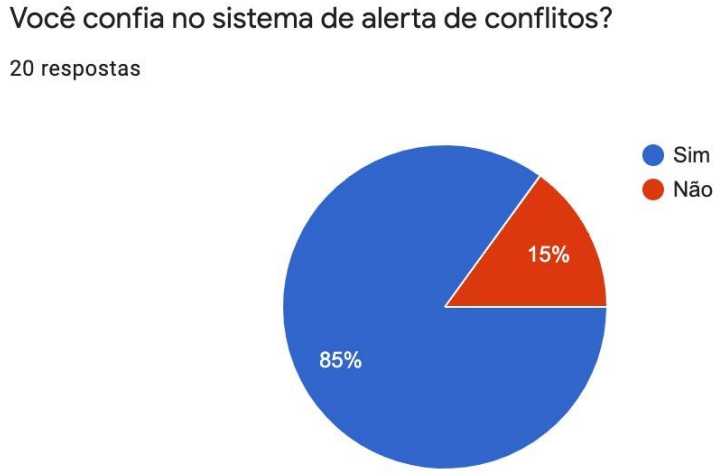
Gráfico 11 – Percentual dos entrevistados de acordo com a pergunta 11.



Fonte: dados da pesquisa.

A décima segunda pergunta, apresentada no gráfico 12, é se os entrevistados confiavam no sistema de alerta de conflitos: 85% dos entrevistados confiam no sistema, enquanto 15% não.

Gráfico 12 – Percentual dos entrevistados de acordo com a pergunta 12.



Fonte: dados da pesquisa.

A décima terceira pergunta, apresentada no gráfico 13, é se os entrevistados utilizariam a ferramenta futuramente: 75% confirmaram que utilizariam a ferramenta caso

disponível, enquanto 10% utilizariam para consulta de catálogos. 10% afirmaram que provavelmente não utilizariam por causa de sua complexidade, enquanto 5% apenas afirmaram que não utilizariam.

Gráfico 13 – Percentual dos entrevistados de acordo com a pergunta 13.

Fonte: dados da pesquisa.

Em suma, o objetivo geral da monografia foi desenvolver uma ferramenta para auxiliar um projetista de *software* a modelar seus RNFs, ajudando-os a prever se haverá influências entre eles durante o processo de modelagem. Dessa forma, a ferramenta conseguiu atingir seu objetivo, tendo em vista que a grande maioria dos entrevistados acharam a ferramenta útil e confiável, utilizando futuramente quando disponibilizada.

6 CONSIDERAÇÕES FINAIS

Em suma, este trabalho visou a criação de uma ferramenta para auxiliar um projetista de *software* a modelar seus RNFs, ajudando-os a prever influências geradas entre eles. Utilizando-se de um trabalho colaborativo, a ferramenta permite uma análise com diferentes catálogos para identificar possíveis conflitos entre *softgoals*, utilizando-se como alicerce as ferramentas para a modelagem de requisitos.

A SIGSoft 1.0 é uma ferramenta *web* mais intuitiva, onde é possível identificar previamente as influencias geradas entre *softgoals*. É compatível com todos os sistemas operacionais, pois só dependem do navegador, além de disponibilizar armazenamento em nuvem, do modo em que todos os catálogos ficam salvos no servidor da aplicação.

A ferramenta dispõe na barra lateral o *toolbox* com as opções *Softgoal*, *Action* e *Claim*: *Softgoal* representa o objetivo a ser alcançado na aplicação; *Action* é o que deve ser feito para implementar o *Softgoal*. Os itens da barra lateral podem ser arrastados para a área de trabalho. É possível fazer dois (2) tipos de ligações: A ligação *Or* e a *And*. É possível, manualmente, definir o nível de desenvolvimento (*denied, weakly denied, weakly, weakly satisficed, satisficed* e *conflict*), e também definir o tópico e se ele se enquadra como prioridade.

A SIGSoft 1.0 disponibiliza um armazenamento em nuvem, do modo em que todos os catálogos ficam salvos no servidor da aplicação. Ao salvar o catálogo, ele será publicado, possibilitando a importação do mesmo para outros usuários. Ela também possui compatibilidade em todos os sistemas operacionais, pois só depende do navegador. Diferente do NDR-Tool, que só funciona em uma versão especifica do Start-UML no *Windows.*

Com a intencionalidade de colocar a ferramenta em prática, foi realizado períodos testes com vinte (20) estudantes de cursos da área de TI (tecnologia de informação).

A grande maioria dos estudantes são jovens adultos, do curso de engenharia de *software*. A ferramenta se mostrou satisfatória, de acordo com as respostas disponibilizadas pelos participantes, apresentando que em sua grande maioria utilizariam futuramente quando disponibilizada, considerando a ferramenta útil e confiável.

Dessa forma, ao desenvolver a ferramenta, proporcionou a criação de um *software* de *softgoals*, e consequentemente, uma análise acerca das ferramentas de RNFs que existem atualmente. Permite o auxilio a projetistas que utilizem os RNFs, como também permite que aperfeiçoamentos da ferramenta sejam realizados.

REFERÊNCIAS

ANDRADE, A. C. S. *et al*. Gestão de risco sem projetos de *software*: uma abordagem baseada em requisitos não funcionais. **Sistemas & Gestão**, v. 14, n. 2, p. 188–196, 2019.

CHUNG, L. *et al*. **Non-functional requirements in *software* engineering**. [S.l.]: Springer Science & Business Media, 2012. v. 5.

CHUNG, L. NIXON, B. YU, E. MYLOPOULOS, J. **Non-Functional Requirements in Software Engineering.** Kluwer Academic Publishers, 1999.

COUTO, A.; MARTINS, L. E. Um processo de validação de requisitos não-funcionais baseado no NFR-framework. In: **Anais do WER09 - Workshop em Engenharia de Requisitos**. [S.l.: s.n.], 2009. p. 16–17.

CYSNEIROS, L. M. LEITE, J. C. S. P. **Utilizando Requisitos Não Funcionais para Análise de Modelos Orientados a Dados.** Rio de Janeiro. Departamento de Informática PUC. s.d.

CYSNEIROS, L. M. **Requisitos Não Funcionais:** Da Elicitação ao Modelo Conceitual. Tese (Doutorado em Ciências da Computação) - Pontifícia Universidade Católica Do Rio De Janeiro, Rio de Janeiro. s.d.

NOVAES, P. J. D. **Método e linguagem para modelagem gráfica de requisitos de *software* e sistemas**. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2019.

PRESSMAN, R. S. **Engenharia de *software*:** uma abordagem profissional. Tradução de Ariovaldo Griesi. 7. ed. Porto Alegre. AMGH Editora Ltda. 2011.

SAKURADA, N. MIYAKE, D. I. Aplicação de simuladores de eventos discretos no processo de modelagem de sistemas de operações de serviços. **Gest. Prod**., São Carlos, v. 16, n. 1, p. 25-43. 2009.

SOMMERVILLE, I. **Engenharia de *Software*. ed**. [S.l.]: São Paulo, SP: Pearson Prentice Hall, 2011.

Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK). (Texto e tradução) Project Management Institute. 6. ed. Pensilvânia. Project Management Institute, Inc. 2017.

WIEGERS, K.; BEATTY, J. ***Software* requirements**. [S.l.]: Pearson Education, 2013.

XAVIER, L. **Integração de Requisitos não Funcionais a Processos de Negócios:** Integrando BPMN e NFR. Dissertação (Mestrado), 2009.

APÊNDICE A – QUESTIONÁRIO UTILIZADO NA COLETA DE DADOS

PROPOSTA DE UMA FERRAMENTA PARA ASSISTÊNCIA À MODELAGEM DE REQUISITOS NÃO FUNCIONAIS DE SOFTWARE

1 - Idade

( ) 18 - 20

( ) 21 - 25

( ) 26 - 30

( ) 30 - 40

( ) acima de 40

2 - Sexo

( ) Masculino

( ) Feminino

( ) Não informar

3 - Curso

( ) Engenharia de Software

( ) Ciencia da computação

( ) Engenharia de Computação

( ) Sistema de Informação

( ) Redes de Computadores

( ) Design Digital

4 - Semestre

( ) 3o ao 5o

( ) 6o ao 8o

( ) 9o ao 10o

( ) Acima do 10o

5 - Você tem experiência profissional no desenvolvimento de softwares?

( ) Sim

( ) Não

6 - Você utiliza/ou alguma ferramenta de modelagem de software?

( ) Sim

( ) Não

7 - Se sim, Com qual frequência você utiliza essa ferramenta?

( ) Diariamente

( ) Semanalmente

( ) Mensalmente

( ) Raramente

( ) Não aplica

8 - Você conhece alguma ferramenta de modelagem de software goals?

( ) Sim

( ) Não

9 - O que você achou da proposta da ferramenta?

( ) Muito bom.

( ) Bom.

( ) Ruim.

( ) Muito ruim.

10 - Você acha que a ferramenta contribui para identificação de conflito de requisitos não funcionais?

( ) Com certeza.

( ) Sim, algumas vezes.

( ) Não.

11 - Você achou a ferramenta fácil de utilizar?

( ) Sim

( ) Sim, mas demorei um pouco para entender.

( ) Não, achei muito complexo.

( ) Não.

12 - Você confia no sistema de alerta de conflitos?

( ) Sim.

( ) Não.

13 - Caso a ferramenta seja disponibilizada, você utilizaria?

( ) Sim

( ) Sim, mas apenas para consulta de catálogos.

( ) Não, achei muito complexo.

( ) Não.