UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE QUIXADÁ

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

AURISLANIA PEREIRA BATISTA

DECOMPOSIÇÃO DE REQUISITOS DE CONFIABILIDADE PARA SISTEMAS AUTODAPTATIVOS UTILIZANDO O NFR FRAMEWORK

QUIXADÁ

2022

AURISLANIA PEREIRA BATISTA

DECOMPOSIÇÃO DE REQUISITOS DE CONFIABILIDADE PARA SISTEMAS AUTODAPTATIVOS UTILIZANDO O NFR FRAMEWORK

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

Orientadora: Profa. Dra. Carla Ilane Moreira Bezerra

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)

B336d Batista, Aurislania Pereira.
Decomposição de requisitos de confiabilidade para sistemas autodaptativos utilizando o NFR framework / Aurislania Pereira Batista. – 2022.
74 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Engenharia de Software, Quixadá, 2022.
Orientação: Profa. Dra. Carla Ilane Moreira Bezerra.

1. Sistema Autoadaptativo. 2. Requisitos não funcionais (Engenharia de sistemas). 3. Qualidade. 4. Confiabilidade. I. Título.

CDD 005.1

AURISLANIA PEREIRA BATISTA

DECOMPOSIÇÃO DE REQUISITOS DE CONFIABILIDADE PARA SISTEMAS AUTODAPTATIVOS UTILIZANDO O NFR FRAMEWORK

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

Aprovada em: ___/___/___

BANCA EXAMINADORA

Profa. Dra. Carla Ilane Moreira Bezerra (Orientadora)
Universidade Federal do Ceará (UFC) - Quixadá

Prof. Dr. Paulo Henrique Mendes Maia
Universidade Estadual do do Ceará (UECE)

Profa. Dra. Rainara Maia Carvalho
Universidade Federal do Ceará (UFC) - Quixadá

Prof. Dr. Enyo José Tavares Gonçalves
Universidade Federal do Ceará (UFC) - Quixadá

À minha família, em especial meus pais e meu namorado que sempre estiveram ao meu lado durante toda a caminhada me dando forças para continuar. Obrigado por sempre acreditarem em mim, sem vocês nada disso seria possível.

AGRADECIMENTOS

Á Profª, Drª Carla Ilane Moreira Bezerra por me orientar em meu trabalho, pela sua paciência e seus incentivos que me ajudaram a ir mais longe e me proporcionaram tanto aprendizado. Obrigado por tudo e por me orientar tão bem durante todo este período.

Ao Prof. Dr. Tobias Rafael Fernandes Neto, coordenador do Laboratório de Sistemas Motrizes (LAMOTRIZ) onde este *template* foi desenvolvido.

Ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

Ao aluno Thiago Nascimento do curso de ciência da computação da Universidade Estadual do Ceará que elaborou o *template* do qual este trabalho foi adaptado para Universidade Federal do Ceará.

Aos meus pais e irmãos, que nos momentos de minha ausência dedicados ao estudo superior, sempre me deram apoio e força para não desistir. À meu namorado, que está ao meu lado desde o início desta caminhada e sempre me ajudou e apoiou, mesmo com todas as adversidades que surgiram pelo caminho. Aos meus amigos, que nunca deixaram de me apoiar e acreditar em mim, mesmo distantes.

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender.

“Não existe triunfo sem perda, não há vitória sem sofrimento, não há liberdade sem sacrifício.”

(J. R. R. Tolkien)

RESUMO

Sistemas autoadaptativos (SAS) são capazes de alterar seu próprio comportamento em tempo de execução de acordo com suas necessidades e problemas, apresentando operações dinâmicas e complexas. Uma forma de ajudar a garantir a qualidade dos SAS é lidando corretamente com os requisitos não-funcionais (RNFs), como segurança, usabildiade e confiabilidade. Entretanto, lidar com RNFs é uma tarefa naturalmente difícil, visto que eles possuem uma série de problemas relacionados a seus relacionamentos com outros RNFs ou com suas subcaracterísticas. Além disso, eles são relativos e a interpretação acerca deles pode variar dependendo de muitos fatores. Dentro do escopo dos SASs, além desta dificuldade natural, existem também as dificuldades do próprio domínio. Os SASs possuem características próprias que estão diretamente ligadas aos NFRs ou características de qualidade e que devem ser mantidas corretamente para que o sistema funcione. Portanto, conhecer como essas características se relacionam com os RNFs se torna importante. Uma maneira de lidar com as problemáticas contidas ao lidar com RNFs é utilizando SIGs (*Softgoal Interdependency Graph*), unidades básicas que representam requisitos não funcionais ou metas genéricas e flexíveis. O conceito de SIG foi proposto por Chung *et al.* (2000) em conjunto com o NFR *framework*. Visando prover suporte à implantação dos RNFs nos SAS e, consequentemente, contribuir com a melhoria da garantia da qualidade desses sistemas, este trabalho tem o objetivo de propor um SIG de confiabilidade para sistemas autoadaptativos. Neste trabalho, foi realizado um mapeamento sistemático com o objetivo de fornecer uma visão da literatura sobre a utilização dos RNFs nos SASs. Após isso, foi iniciado o processo de construção do SIG, foi realizado o mapeamento de todas as interdependências encontradas entre as características envolvidas. Ao final do mapeamento, a característica confiabilidade foi identificada como uma das características mais recorrentes nos estudos selecionados. Além disso, foi identificado que a confiabilidade tem grande impacto no funcionamento dos SASs, o que motivou ainda mais a escolha desta característica. Em seguida, foi realizada a etapa de decomposição do RNF de confiabilidade, gerando como produto um SIG de confiabilidade para SAS.

Palavras-chave: Sistemas Autoadaptativos. Requisitos Não-Funcionais (Engenharia de sistemas). Qualidade. Confiabilidade.

ABSTRACT

Self-adaptive systems (SASs) can change their behavior at runtime according to their needs and problems, presenting dynamic and complex operations. One way to guarantee the quality of SAS is to correctly handle non-functional requirements (NFRs), such as security, usability and reliability. However, dealing with NFRs is naturally a difficult task, because they come into conflict and contradict each other. Other than that, NFRs are relative, and their interpretation can vary depending on many factors. Within the scope of SASs, there are also domain difficulties in addition to this natural difficulty. SASs have their characteristics directly linked to NFRs or quality characteristics that must be properly maintained for the system to work. Therefore, know-how these characteristics relate to NFRs becomes important. One way to deal with conflict problems within NFRs is to use SIGs (Softgoal Interdependency Graphs), basic units representing non-functional requirements or generic and flexible goals. The SIG concept was proposed by Chung *et al.* (2000) with the NFR framework concept. To support the implementation of NFRs in SAS and, consequently, contribute to improving the quality assurance of these systems, this work aims to propose a reliability SIG for SASs. In this work, a systematic mapping was carried out to provide an overview of the literature on the use of NFRs in SAS. At the end of the mapping, reliability was identified as one of the most recurrent characteristics in the selected studies. In addition, it was identified that reliability greatly impacts the functioning of SAS, which further motivated the choice of this characteristic. Then, the decomposition step of the reliability NFR was carried out, generating a reliability SIG for the SAS as a product. After that, the decomposition of the reliability NFR was carried out, generating as a product a reliability SIG for SASs.

Keywords: Self-Adaptative Systems. Non-Functional Requirements (Systems engineering). Quality. Reliability.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ER	Engenharia de requisitos
SAS	Sistemas Autoadaptativos
SIG	Softgoal interdependence graphic
IoT	Internet das coisas
RNF	Requisito não-funcional
GT	Grounded Theory
IHC	Interação humano computador

SUMÁRIO

1 INTRODUÇÃO

Segundo Wu *et al.* (2019) sistemas autoadaptativos (SAS) são sistemas que conseguem se modificar e realizar operações de melhoria ou incremento em tempo de execução, sendo ainda capazes de identificar falhas de operação e com isso disparar estratégias de configuração para continuar em operação sem deixar de oferecer um serviço. Em decorrência disto, esses sistemas geralmente precisam atender a uma série de aspectos voltados para sua qualidade (e.g., disponibilidade, segurança, eficiência e desempenho) (RAIBULET *et al.*, 2017).

SAS são cada vez mais utilizados em diversos domínios, tais como, cidades inteligentes, telecomunicações e prevenção de catástrofes (ADJOYAN; SERIAI, 2017). Sousa *et al.* (2019b) apresentam em seu trabalho que o número de estudos relacionados a sistemas dinâmicos com propriedades de autônomas e autonômicas, como os sistemas autoadaptativos tem sido ampliado, o que forte indicadores de um crescente interesse neste tipo de sistema.

Devido às suas características específicas, estes sistemas estão diretamente ligados a RNFs ou características de qualidade e para que estes sistemas funcionem corretamente, é necessário que estas características sejam implementadas de maneira correta (RAIBULET *et al.*, 2017). Desta forma, vê-se a necessidade de estudos voltados para o processo de ER (engenharia de requisitos) em SAS, com foco nos RNFs.

A Engenharia de Requisitos (ER) é um processo que viabiliza e engloba todas as atividades presentes durante a etapa de construção dos requisitos de um software. Nela é realizado todo o processo de descobrir, analisar, documentar e verificar as funções e restrições do sistema (PRESSMAN; MAXIM, 2016). Os requisitos não-funcionais (RNFs) dizem respeito às características do sistema e estão relacionados ao uso da aplicação em termos de desempenho, segurança, confiabilidade, dentre outros (HAMMANI, 2014).

Chung e Leite (2009) afirmam que a fase de ER possui fundamental importância para o desenvolvimento de um software e é uma das etapas mais críticas de todo o ciclo de vida dele. De acordo com os autores, isso decorre devido a ER possuir informações que podem ser lidas erroneamente e que podem sofrer modificações ao longo do projeto, o que pode vir a causar grandes problemas posteriormente.

Zhang e Wang (2019) relatam em seu trabalho que os RNFs estão diretamente ligados a questão da qualidade de software, eles tendem a possuir problemas entre suas interações e a se contradizer, e isso é reconhecido como um dos principais problemas ao lidar com eles. Além disso, os RNFs são relativos, e a interpretação acerca deles pode variar dependendo de muitos

fatores, como o contexto do sistema que está sendo desenvolvido e a extensão do envolvimento das partes interessadas (CHAZETTE; SCHNEIDER, 2020).

Segundo Zhang e Wang (2019), o impacto prático dos RNFs não foi apenas reconhecido pela comunidade de pesquisa, mas também foi documentado em muitos estudos realizados na indústria. Os autores citam que negligenciar RNFs durante o desenvolvimento de software é um dos dez maiores riscos do processo de desenvolvimento, os mais caros e difíceis de corrigir. Visto que eles são inevitáveis, a necessidade de lidar com essas problemáticas existentes é de fundamental importância.

Uma forma de visualizar as interações entre os RNFs é utilizando o NFR *framework* e o conceito de SIGs (*Softgoals Interdependence Graphics*). Seu objetivo é ajudar desenvolvedores na implementação de soluções personalizadas, levando em consideração as características do domínio e do sistema em questão (CHUNG *et al.*, 2000). Para isso, geralmente é utilizado o conceito de SIG, que descreve as dependências entre os *softgoals* e como eles são decompostos. *Softgoals* são unidades básicas que representam requisitos não funcionais ou metas genéricas e flexíveis, este conceito foi proposto por Chung *et al.* (2000) e geralmente é utilizado com o NFR *framework*.

Tendo em vista o estudo de Chazette e Schneider (2020), que citam o fato dos RNFs serem relativos, pode-se interpretar que as técnicas para o gerenciamento desses requisitos também são relativas e dependem do contexto do projeto. Por exemplo, Wiegers e Beatty (2013) demonstram em seu trabalho que os requisitos de eficiência têm relação negativa (conflito) com o requisito de usabilidade, mas de acordo com Egyed e Grunbacher (2004) esses dois tipos de RNFs tem relacionamento positivo (suporte). Quanto ao domínio dos SAS, estudos como o de Sousa *et al.* (2019a) mostram que a característica de confibilidade possui grande impacto nestes sistemas, visto que ela está interligada a muitas das características próprias que eles possuem, o que podem impactar diretamente no funcionamentos desses sistemas.

Portanto, este trabalho tem como objetivo apresentar um SIG que represente graficamente as interações do RNF de confiabiliade com suas subcaracterísticas e com as características específicas do domínio dos SAS. Para o desenvolvimento deste trabalho, inicialmente foi realizado um mapeamento sistemático da literatura para identificar os principais RNFs para sistemas autoadaptativos, através deste mapeamento o RNF de confiabilidade foi escolhido como RNF a ser decomposto, pois os dados retornaram que esta característca é muito importante para SAS e possui ligação direta com algumas das principais características próprias destes sistemas.

Posteriormente, foi utilizado o NFR *framework* para decompor o requisito escolhido, e a partir disto obter o SIG. Além disso, a metodologia do trabalho de Carvalho *et al.* (2018) serviu como apoio para o processo de criação do SIG. Ao final deste estudo, foi gerado um SIG que servirá de apoio aos desenvolvedores de sistemas desse domínio, bem como aos pesquisadores da área, a fim de evitar retrabalho e os demais problemas que surgem nesta etapa do ciclo de vida do software.

Este trabalho está estruturado da seguinte forma: no Capítulo 2 são apresentados os conceitos principais sobre o tema deste trabalho; o Capítulo 3 apresenta os trabalhos que possuem relação com este; no Capítulo 4 são descritos as etapas da metodologia do estudo; no Capítulo 5 é detalhado o mapeamento sistemático que foi realizado; no Capítulo 6 detalha-se o processo de decomposição do RNF e criação do SIG, bem como o seu processo de avaliação; O Capítulo 7 especifica as ameças à validade deste trabalho; e finalmente no Capítulo 8 são apresentadas as conclusões do trabalho e trabalhos futuros que podem se beneficiar desta pesquisa.

1.1 Objetivos

O objetivo principal deste trabalho é construir um SIG de um RNF de importância e impacto para SAS de forma a auxiliar o processo de desenvolvimento desses sistemas e melhorar sua qualidade. Para alcançar o objetivo geral deste trabalho, foram definidos objetivos específicos listados a seguir:

1. Investigar RNFs ou características e sub-características usados na avaliação da qualidade de sistemas autoadaptativos a partir de um mapeamento sistemático da literatura.
2. Selecionar e decompor o RNF selecionado.
3. Elaborar um SIG desse RNF utilizando o NFR *framework*.
4. Avaliar o SIG com um especialista do domínio dos SAS.

2 FUNDAMENTAÇÃO TEÓRICA

Essa capítulo apresenta a fundamentação teórica deste trabalho. A Seção 2.1 discorre sobre as definições de sistemas autoadaptativos. A Seção 2.2 apresenta uma visão geral sobre o processo da Engenharia de Requisitos. A Seção 2.3 apresenta o modelo de qualidade da ISO 25000. Por fim, a Seção 2.4 apresenta o conceito do NFR *framework*, mostrando como este *framework* funciona e como ele pode ser utilizado.

2.1 Sistemas autoadaptativos

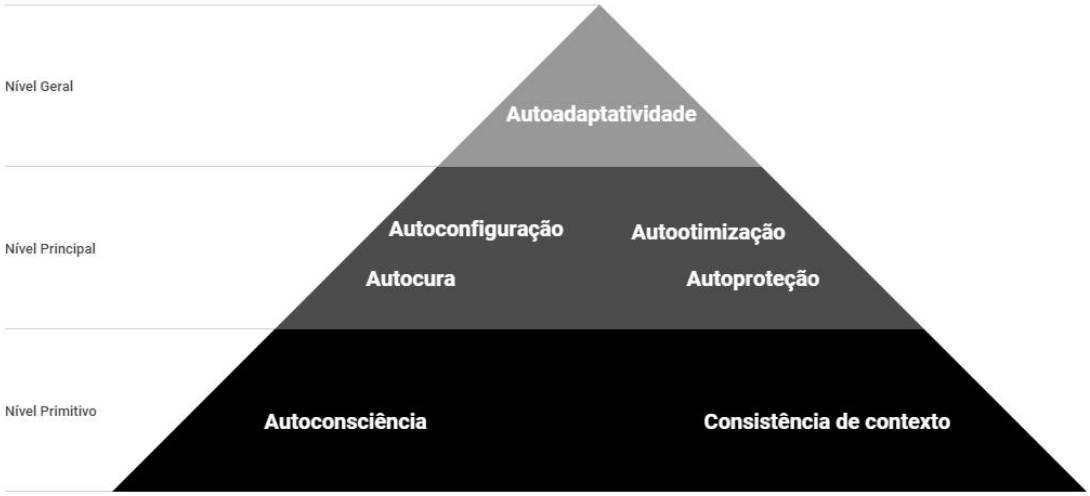
Com a complexidade que os softwares atuais estão tomando, as propriedades Self* (ou seja, autoadaptáveis) são cada vez mais importantes para fornecer bons mecanismos de verificação e validação a fim de garantir a qualidade do software (RAIBULET *et al.*, 2017). Wu *et al.* (2019) definem SAS como softwares capazes de avaliar seu próprio comportamento e assim, alterá-lo caso sua avaliação indique que é necessário, com o mínimo de intervenção humana. Como exemplo disso, existem os casos onde é possível após essa avaliação obter melhores funcionalidades, melhorar seu próprio desempenho ou até mesmo modificar algo que ele esteja realizando de forma errônea (LADDAGA; ROBERTSON, 2004). Dessa forma, esses sistemas têm a capacidade de adaptar-se automaticamente em resposta às mudanças no ambiente em que estão inseridos, sem que ocorra interação humana (SEN *et al.*, 2015).

A principal razão para precisarmos de softwares adaptativos é o custo ao precisar lidar com a complexidade dos sistemas de software para atingir seus objetivos (ROBERTSON *et al.*, 2000). Entre esses objetivos, alguns lidam com gestão complexa, robustez no tratamento de condições inesperadas (por exemplo, falha), mudança de prioridades, requisitos incertos e políticas que regem as metas e condições de mudança (por exemplo, no contexto de mobilidade). Uma parte significativa das pesquisas sobre o manuseio de complexidade e como atingir metas de qualidade tem se concentrado no desenvolvimento de software e seus atributos de qualidade internos (como no modelo de qualidade da ISO 25010 (ISO, 2011). Porém, nos últimos anos, tem havido uma demanda crescente para lidar com esses problemas em tempo de operação (tempo de execução). As principais causas para essa tendência incluem um aumento no nível de heterogeneidade dos componentes de software, mudanças mais frequentes no contexto ou objetivos dos requisitos durante o tempo de execução e necessidades de segurança mais altas (SALEHIE; TAHVILDARI, 2009).

*2.1.1 Propriedades self-**

As propriedades de software adaptável são freqüentemente conhecidas como propriedades self-*. Um dos primeiros conjuntos conhecidos de propriedades self-* foi introduzido pela IBM e inclui oito propriedades (IBM-AC, 2001). A Figura 1 ilustra uma hierarquia dessas propriedades em três níveis. Nesta hierarquia, autoadaptatividade e autoorganização são propriedades gerais, que são decompostas em propriedades principais e primitivas em dois níveis diferentes. O resto desta seção elabora detalhadamente cada nível desta hierarquia.

Figura 1 – Hierarquia de propriedades dos sistemas Self*



Fonte: Adaptado de Salehie e Tahvildari (2009).

O nível geral destacado na Figura 1 contém as propriedades globais de softwares autoadaptativos. Um sistema com propriedades desse nível tipicamente possuem muitos elementos capazes de interação, mas que não possuem inconsistências no total ou parcialmente ao visualizar o sistema como um todo (SALEHIE; TAHVILDARI, 2009).

O nível principal possui propriedades definidas para os mecanismos de autoadaptação. Curiosamente, estas propriedades foram definidas de acordo com mecanismos de autoadaptação biológica (KEPHART; CHESS, 2003). Como exemplo disso, o trabalho de Salehie e Tahvildari (2009) cita que o corpo humano tem propriedades semelhantes, a fim de se adaptar às mudanças em seu contexto (e.g., mudança de temperatura no ambiente) ou no próprio corpo (e.g.,uma lesão ou falha em um dos órgãos internos). As propriedades desse nível são detalhadas a seguir (SALEHIE; TAHVILDARI, 2009):

– **Autoconfiguração:** É a capacidade de se reconfigurar automaticamente e dinamicamente

em resposta às mudanças instalando, atualizando, integrando e compondo criação ou decomposição de entidades de software.

- **Autocura:** É a capacidade de descobrir, diagnosticar e reagir às interrupções. Também pode antecipar possíveis problemas e, portanto, tomar as medidas adequadas para evitar uma falha.
- **Autootimização:** É a capacidade de gerenciar o desempenho e a alocação de recursos, a fim de satisfazer os requisitos de diferentes usuários. Tempo de resposta ponta a ponta, rendimento, utilização e carga de trabalho são exemplos de preocupações importantes relacionadas a esta propriedade.
- **Autoproteção:** É a capacidade de detectar violações de segurança e se recuperar de seus efeitos. Tem dois aspectos, defender o sistema contra ataques maliciosos e antecipar problemas, tomando ações para evitá-los ou para mitigar seus efeitos.

O nível primitivo é composto pelas propriedades de autoconsciência e consciência de contexto. As propriedades são descritas a seguir:

- **Autoconsciência:** Significa que o sistema está ciente de seus estados e comportamentos próprios. Esta propriedade é baseada em automonitoramento que reflete o que é monitorado.
- **Consistência de contexto:** Significa que o sistema está ciente do seu contexto, o seu ambiente operacional.

As características de sistemas autoadaptativos possuem relacionamento direto com os RNFs/características de qualidade, a característica de autocura por exemplo, se relaciona fortemente com a confiabilidade (SOUSA *et al.*, 2019a). Por conta disto, para cada uma das propriedades que um sistema autoadaptativo possui é importante verificar se as características de qualidade relacionadas a ele estão corretas.

2.2 Engenharia de requisitos

A Engenharia de Requisitos (ER) é amplamente reconhecida como a primeira fase do processo de engenharia de software e é considerada como a principal tarefa do desenvolvimento de software (WAHONO, 2003). Por ser um processo com uma grande diversidade de termos e definições, não há uma única descrição capaz de defini-la, elas vão depender dos autores e do momento em que foram escritas.

Pressman e Maxim (2016), por exemplo, dizem que a ER é o meio pelo qual é

possível entender o que o cliente quer, o que deve ser feito e quais suas necessidades. Através dela também é possível negociar soluções razoáveis que sejam boas para ambos os envolvidos, bem como realizar o processo de especificação, gerenciamento e validação dos requisitos conforme eles vão sendo transformados em funcionalidades do sistema.

Para Nuseibeh e Easterbrook (2000), a ER é um processo de descoberta do propósito, identificando as partes interessadas e suas necessidades, documentando-as de uma forma que seja passível de análise, comunicação e posterior implementação. Independente disso, o processo de ER possui um conjunto de atividades sucintas, utilizadas para documentar, analisar, revisar, manter e validar requisitos para um sistema, são elas: elicitação de requisitos, análise de requisitos, documentação de requisitos, verificação e validação de requisitos (CHEMUTURI, 2012).

Segundo Lamsweerde *et al.* (1998), a fase de ER é de fundamental importância para o desenvolvimento de um software e também é uma das etapas mais críticas de todo o ciclo de vida dele. De acordo com o autor, isso decorre devido a ER possuir processos difíceis de mensurar, bem como de obter informações totalmente precisas e imutáveis. Além disso, ainda existe a possibilidade de entender e elicitar tais informações erroneamente, o que pode causar grandes problemas posteriores. Portanto, esta fase do ciclo de vida dos softwares requer bastante atenção e cuidado ao ser executada.

2.2.1 Requisitos não-funcionais

Hammani (2014) define requisitos não-funcionais (Requisito não-funcional (RNF)s) como características do sistema que estão relacionados ao uso da aplicação em termos de desempenho, segurança, confiabilidade, dentre outros. Os RNFs não expressam o que uma funcionalidade deve fazer, mas sim quais comportamentos ou restrições o software deve atender. Estes requisitos estão contidos no processo de Engenharia de Requisitos e impactam diretamente na execução e qualidade do mesmo.

Segundo Cysneiros e Leite (1998), muitos dos RNFs que são esquecidos podem aumentar muito o custo final de um sistema e até mesmo inviabilizar a implantação do mesmo. Isso ocorre pois os RNFs são especialmente mais difíceis de se implementar, como mostram Zhang e Wang (2019) em seu trabalho. Eles tendem a entrar em conflito e se contradizer, e isso é reconhecido como uma das principais características deles. Além disso, os autores falam que os RNFs são relativos, e a interpretação acerca deles pode variar dependendo de muitos fatores,

como o contexto do sistema que está sendo desenvolvido e a extensão do envolvimento das partes interessadas.

Existem diferentes motivos para os requisitos entrarem em conflito em si, o trabalho de Robinson *et al.* (2003) mostra uma boa categorização para identificar as razões dos conflitos, eles as classificam em razões técnicas e razões sociais. Razões técnicas são causado pelas seguintes dificuldades:

- Grande quantidade de requisitos pode levar a conflitos entre eles, pois mudanças nos requisitos podem ocorrer após a adição de novos requisitos, ou após a atualização de antigos.
- Um sistema com domínio muito complexo pode levar a requisitos mal entendidos e portanto, conflitantes.

Considerando que o trabalho também cita, as dificuldades sociais como razão para os conflitos, é mostrado o seguinte:

- Sistemas com diferentes partes interessadas e com diversos interesses diferentes geralmente causam conflitos;
- Mudanças nas partes interessadas do sistema, adicionando novas partes interessadas, com necessidades diferentes ou solicitações que mudam constantemente, também tendem a causar muitos conflitos;

Não há uma só abordagem para encontrar e identificar conflitos entre requisitos. No geral, essas abordagens vão depender do contexto, software em questão, domínio, dentre outros aspectos. Também não existe uma única forma de realizar a classificação desses conflitos. Cada trabalho oferece uma classificação diferente após a descoberta, a escolha é feita com base na técnica usada para detectar os conflitos (ROBINSON *et al.*, 2003).

2.2.2 Processo de ER em SAS

A atividade de ER voltada para SAS é inerente e parcialmente incompleta (MORANDINI *et al.*, 2017). Levando em consideração a variabilidade nos requisitos para sistemas adaptativos e as várias soluções alternativas resultantes disso, muitas decisões que são tradicionalmente feitas de acordo com os requisitos e a necessidade de tempo de projeto, tendem a ser deslocadas para o tempo de execução (NITTO *et al.*, 2008).

De acordo com Salehie e Tahvildari (2009), uma maneira plausível de realizar a elicitação de requisitos em softwares autoadaptativos é com a ajuda dos seis homens honestos.

Eles explicam em seu trabalho que essas seis questões são muito importantes na elicitação de requisitos para SAS. Robertson *et al.* (2000) usa uma ideia semelhante para abordar parcialmente esses requisitos. As questões abaixo são uma versão modificada por Salehie e Tahvildari (2009) das perguntas de Robertson *et al.* (2000) para eliciar os requisitos essenciais de autoadaptação:

- **Onde:** Este conjunto de perguntas está relacionado com a necessidade de mudança. Quais artefatos serão mudados, em qual camada (por exemplo, middleware)? Esse tipo de pergunta é utilizada para localizar o problema que precisa ser resolvido por conta da adaptação.
- **Quando:** Essas perguntas estão relacionadas ao tempo. Quando uma mudança precisa ser aplicada e quando é possível realizá-la? Pode ser aplicado a qualquer momento que o sistema requer, ou existem restrições que limitam alguns tipos de mudança? Qual a frequência de alteração do sistema? Dentre outras.
- **O que:** Essas perguntas identificam quais atributos ou artefatos do sistema serão mudados por meio de ações de adaptação, e o que precisa ser mudado em cada situação. Eles podem variar de parâmetros e métodos a componentes, estilo de arquitetura e recursos do sistema.
- **Por quê:** Este conjunto de perguntas trata das motivações para a construção do sistema autoadaptativo. Essas questões estão preocupadas com os objetivos atendidos pelo sistema (por exemplo, robustez). Se há um objetivo em utilizar certa abordagem de engenharia de requisitos para realizar a elicitação, este conjunto de perguntas identifica os objetivos do sistema de software autoadaptativo.
- **Quem:** Essas perguntas abordam o nível de automação e integração humana envolvido no software autoadaptativo. Com relação à automação, espera-se que haja intervenção humana mínima, enquanto uma interação eficaz com os proprietários e gerentes do sistema também é necessária em alguns aspectos específicos.
- **Como:** Um dos requisitos importantes para adaptação é determinar como os artefatos adaptáveis podem ser alterados e quais ações de adaptação podem ser adequadas para ser aplicado em uma determinada condição.

Durante o processo de desenvolvimento do sistema autoadaptativo, as perguntas acima devem ser respondidas em duas fases: 1) na fase de desenvolvimento, que trata do desenvolvimento e construção do software autoadaptável do zero ou por meio da reengenharia de um sistema legado, 2) na fase operacional, que gerencia as questões operacionais para responder adequadamente às mudanças no contexto Self da aplicação, a fim de tentar diminuir os problemas

advindos dessa etapa de construção do software (SALEHIE; TAHVILDARI, 2009).

2.3 ISO 25000

A norma ISO/IEC 25000 (ISO/IEC, 2014) consiste em um conjunto de normas para avaliação da qualidade de produtos de software. A norma SQuaRE possui as seguintes divisões (ISO/IEC, 2014): ISO/IEC 2500n (Divisão da Gestão da Qualidade), ISO/IEC 2501n (Divisão do Modelo de Qualidade), ISO/IEC 2502n (Divisão de Métricas de Qualidade), ISO/IEC 2503n (Divisão de Requisitos de Qualidade), ISO/IEC 2504n (Divisão da Avaliação da Qualidade) e ISO/IEC 25050 – ISO/IEC 25099 (Extensão de Padrões SQuaRE).

Segundo a norma SQuaRE (ISO/IEC, 2011), um modelo de qualidade é composto de características, subcaracterísticas e atributos de qualidade. Característica de qualidade é um conjunto de propriedades de um produto de software pela qual sua qualidade pode ser definida e avaliada (ISO/IEC, 2001). Uma característica pode ser dividida em várias subcaracterísticas. Atributos são propriedades mensuráveis, físicas ou abstratas, de uma entidade. As medidas de qualidade são utilizadas para medir, de forma direta ou indireta, os atributos, as subcaracterísticas e as características de qualidade em um produto de software. Uma medida é um mapeamento de uma entidade para um número ou símbolo, a fim de caracterizar uma propriedade da entidade (ISO/IEC, 2011). As medidas são essenciais na avaliação da qualidade de um produto, e estas podem compor um modelo de qualidade.

O modelo SQuaRE possui oito características e trinta e uma subcaracterísticas relacionadas à qualidade interna e externa, conforme ilustrado na Tabela 1. Como apenas a característica de confiabilidade será abordada neste trabalho, a seguir são descritas sua definição e suas subcaracterísticas (ISO/IEC, 2011):

- **Confiabilidade**: grau em que um sistema, produto ou componente mantém, ao longo do tempo, um comportamento consistente com o esperado, sob as condições especificadas.
  - Maturidade: grau no qual um sistema, produto ou componente satisfaz às necessidades de confiabilidade em sua operação normal.
  - Tolerância a falhas: grau em que um sistema, produto ou componente opera como pretendido, apesar da presença de falhas de hardware ou software.
  - Recuperabilidade: grau em que, em caso de interrupção ou falha, um produto ou sistema pode recuperar os dados diretamente afetados e re-estabelecer o estado desejado do sistema.

Tabela 1 – Características e atributos de qualidade da ISO/IEC 25010 (ISO/IEC, 2011)

Características de Qualidade	**Atributos de Qualidade**
Adequação Funcional	Completude Funcional
	Corretude Funcional
	Funcionalidade apropriada
Confiabilidade	Maturidade
	Tolerância a Falhas
	Recuperabilidade
	Disponibilidade
Usabilidade	Conhecimento Adequado
	Apreensibilidade
	Operabilidade
	Acessibilidade
	Proteção de Erro de Usuário
	Estética de Interface com o Usuário
Eficiência de Desempenho	Comportamento em Relação ao Tempo
	Comportamento em Relação aos Recursos
	Capacidade
Manutenibilidade	Analisabilidade
	Modificabilidade
	Modularidade
	Testabilidade
	Reusabilidade
Portabilidade	Adaptabilidade
	Instabilidade
	Substituibilidade
Segurança	Confidencialidade
	Integridade
	Não-Repúdio
	Responsabilização
	Autenticidade
Compatibilidade	Coexistência
	Interoperabilidade

Fonte: Adaptada de ISO (2011)

– Disponibilidade: grau em que um sistema, produto ou componente está operacional e acessível quando requisitado para uso.

2.4 NFR *framework*

O NFR *framework* foi desenvolvido por Chung *et al.* (2000), na *University of Toronto*. Nele os RNFs são tratados como metas flexíveis, ou seja, metas que precisam ser alcançadas, não totalmente mas ao menos em uma boa quantidade (CHUNG *et al.*, 2000). Ele oferece uma estrutura para representar e registrar o processo de projeto e raciocínio em grafos, chamados grafos de interdependência entre softgoals (SIGs, este conceito será apresentado na próxima seção). Sua linguagem de fácil entendimento oferece a possibilidade do uso dos grafos que são construídos para documentação de requisitos não-funcionais.

O trabalho de Chung *et al.* (2000) introduz a noção de *satisficing* e com base nisso,

um *softgoal* é utilizado para satisfazer outro. *Softgoals* representam uma meta sem definição ou critérios claros sobre como atingir sua satisfação e foram um conceito chave para a criação do processo. Nesta abordagem o profissional tem a possibilidade de fazer o refinamento entre RNFs adicionando operacionalizações ao grafo. Operacionalização é uma estratégia real que o desenvolvedor pode utilizar para satisfazer um determinado *softgoal*.

2.4.1 Softgoal Interdependency Graph (SIG)

O termo SIG foi desenvolvido e proposto por (CHUNG *et al.*, 2000), eles são grafos de softgoals interconectados um ao outro onde cada um representa um RNF para o sistema que está sendo desenvolvido. Ele pode ser utilizado para representar e analisar RNFs, tal notação consegue apresentar um nível mais específico de como interações podem ocorrer, apoiando de forma mais concreta os desenvolvedores.

Eles são utilizados para representar os RNFs (ou atributos de qualidade) pois é muito difícil conseguir satisfazer totalmente uma característica. Um SIG é composto de vários *softgoals* e ligações entre eles, metas flexíveis de RNFs ou metas flexíveis de operacionalização. Ambas atuam como restrições para o sistema, e são representadas por uma nuvem. Elas podem ser refinadas em mais metas flexíveis ou mais específicas, como mostrado na Figura 17.

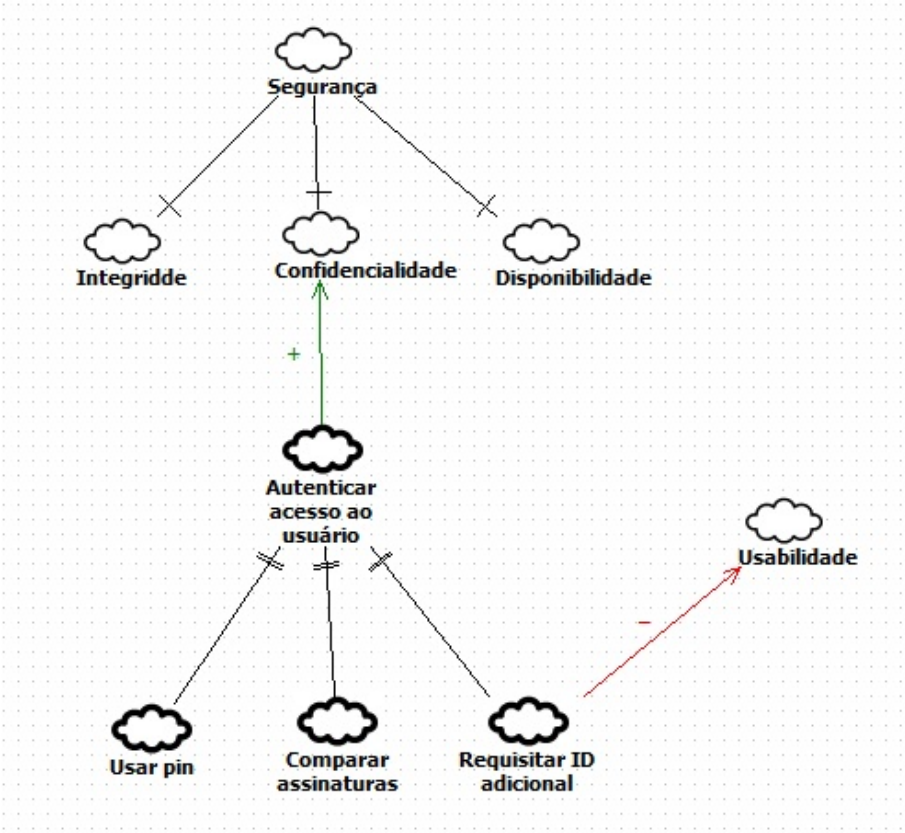
Existem vários tipos diferentes de contribuições para satisfação ou não de um *softgoal*. Alguns termos como MAKE, HELP, HURT e BREAK são utilizados, bem como AND e OR. MAKE e HELP costumam representar *softgoals* que satisfazem o outro positivamente, enquanto BREAK e HURT representam um *softgoal* que possui interação negativa com o outro.

São trabalhados graus de satisfação para cada *softgoal*, seja no quesito positivo ou negativo. Além disso, cada *softgoal* pode também ser associado a um valor de criticidade. Ao utilizar o RNF *framework* os RNFs serão transformados em *softgoals* a serem atendidos, assim se inicia um processo interativo e intercalado, a fim de que a satisfação seja encontrada através de decomposições AND /OR.

No exemplo abaixo, o desenvolvedor foi decompondo os softgoals principais em outros softgoals, até o nível que foi necessário para cada RNF. Note que alguns dos softgoals contidos neste grafo são operacionalizações adicionadas pelo desenvolvedor, por exemplo: "Usar PIN".

Este trabalho utilizará e terá como base catálogos desenvolvidos neste formato e a próxima seção mostrará um processo que utiliza o NFR *framework* bem como os SIGs para seu

Figura 2 – Exemplo de SIG



Fonte: Retirada de Chung *et al.* (2012)

desenvolvimento.

3 TRABALHOS RELACIONADOS

Esse capítulo apresenta os trabalhos relacionados ao tema em estudo. A Seção 3.1 discorre sobre o trabalho de Martinez *et al.* (2019), que propõe um modelo para identificação de conflitos e deficiências em RFNs através da utilização de casos de uso e cenários. A Seção 3.2 apresenta o trabalho de Mairiza e Zowghi (2010) que tem como objetivo principal gerar um catálogo que contenha os conflitos entre RFNs para diferentes domínios. A Seção 3.3 discorre sobre o trabalho de Abdul *et al.* (2010) que desenvolve um modo de identificar conflitos entre RFNs utilizando um mapeamento de matrizes. A Seção 3.4 apresenta o trabalho de Carvalho *et al.* (2018) que propõe um catálogo de invisibilidade para sistemas UbiComp e IoT. A Seção 3.5 apresenta uma Tabela que compara cada trabalho de acordo com os parâmetros estabelecidos. E por fim, a Seção 3.6 que apresenta as relações existentes entre cada trabalho apresentado.

3.1 *A model for detecting conflicts and dependencies in non-functional requirements using scenarios and use case*

Neste trabalho proposto por Martinez *et al.* (2019), casos de uso e cenários são utilizados para a realização de um modelo para detecção de problemas e conflitos entre os RNFs. O modelo proposto possui 3 etapas para execução, que se complementam e são necessárias para a execução da solução.

A primeira etapa consiste em extrair um considerável número de RNFs. Para isso, a fase de elicitação é a primeira a ser realizada. Nela um conjunto de atividades é desenvolvido a fim de obter inicialmente os RFNs que serão analisados. Na segunda etapa se inicia a análise dos requisitos coletados. Aqui, as expectativas de qualidade são decompostas em um nível apropriado de detalhe, estabelecendo relações e categorizando entre eles. Finalmente, os RNFs são especificados de forma estruturada e, em seguida, armazenados de forma persistente e bem organizada. Ao final da etapa, foram obtidos RNFs, cenários e casos de uso definidos inteiramente.

O segundo estágio consiste em, com os RNFs em mãos e com os cenários e casos de uso bem definidos, iniciar o processo de detecção de conflitos e dependências dos mesmos, fazendo uso de ferramentas como: ElasticSearch, Kibana e MAICD. A forma como os dados e as ferramentas interagem entre si é dada da seguinte forma: (i) por meio do aplicativo MAICD os RFNs, (ii) cenários e, (iii) casos de uso são inseridos no banco de dados. Com o *Entity*

Framework, as diferentes operações de inserção, remoção, atualização ou alteração (CRUD) são controladas. O MAICD tem controle sobre algumas ferramentas como a ElasticSearch, e por conta disso, é possível criar 4 índices: (i) para RNFs, (ii) para cenários, (iii) para conflitos e, (iv) para as dependências. A partir disso, um procedimento de análise de texto é realizado para obter conflitos e dependências, e então eles são mostrados no MAICD. Os RNFs e os cenários são atribuídos aos 2 primeiros índices e os conflitos e dependências aos 2 últimos, utilizando a ferramenta Kibana.

O último estágio envolve o processo de visualização de dados, onde se obtém a matriz de conflitos e dependências entre os RNFs no MAICD. Além da matriz, o Kibana pode fornecer uma representação gráfica de conflitos e dependências. Como parte da última etapa, um gráfico de conflitos é gerado mostrando as relações de conflito entre os RNFs com uma linha cinza. Se esta linha for mais espessa, significa que o nível de conflito é maior do que uma linha mais fina.

3.2 *Constructing a Catalogue of Conflicts among Non-functional Requirements*

Este trabalho proposto por Mairiza e Zowghi (2010), apresenta o resultado de uma investigação sistemática da literatura existente sobre uma maior noção de RNFs e os conflitos entre eles, resultando na construção de um catálogo de conflitos de RNFs voltado às características relativas deles. As relações de conflitos podem mudar dependendo do significado dos RNFs no sistema que está sendo desenvolvido. Consequentemente, os modelos de conflitos potenciais existentes permanecem em desacordo uns com os outros.

A primeira etapa desse processo foi a investigação da literatura, conduzida explorando os artigos acadêmicos, recursos e documentos da indústria de desenvolvimento de software. Quatro tipos gerais de fontes de informação foram identificadas para a realização do trabalho: (1) artigos de periódicos; (2) conferência de procedimentos; (3) livros; e, (4) documentos da indústria de software. Este trabalho examinou 182 fontes de informação. Todos são literaturas dentro da disciplina de engenharia de software. Eles cobrem várias questões de RNFs e conflitos entre eles.

A segunda etapa foi a criação de um catálogo inicial dos conflitos entre os RNFs. Nesta fase, as relações de conflito entre os RNFs foram usadas como critérios para desenvolver o catálogo. Esta etapa foi iniciada identificando as interdependências entre os vários tipos de RFNs. Essa interdependências representam as relações entre um determinado tipo de RNF e

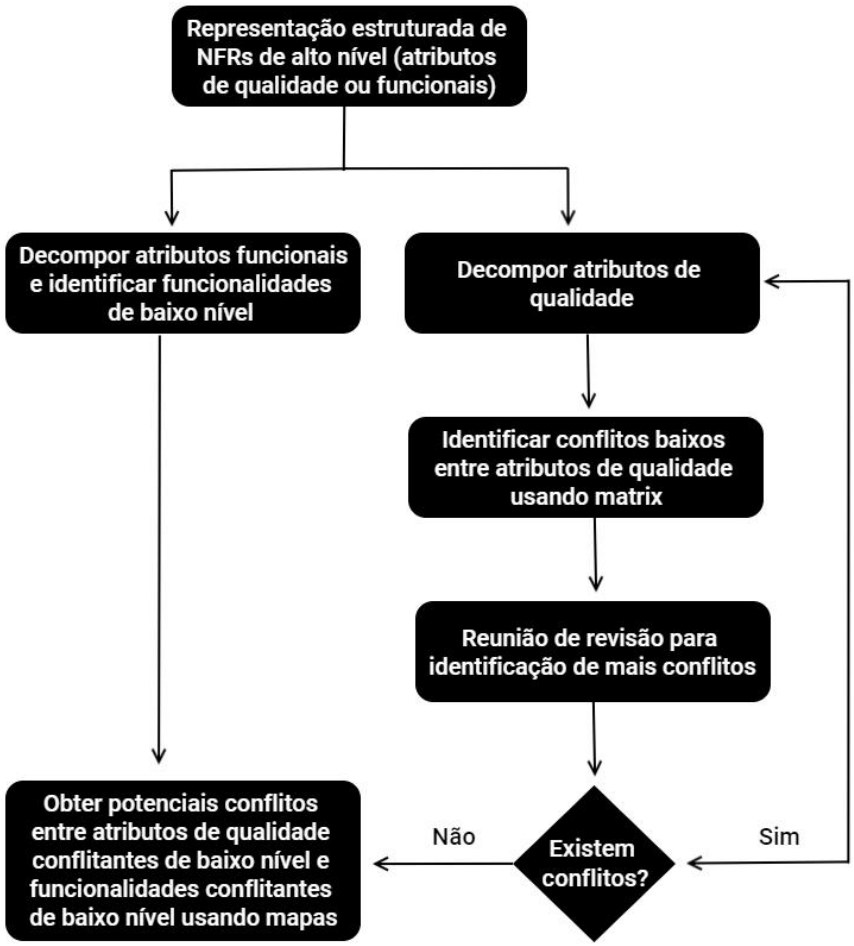
outro. O primeiro catálogo gerado nessa etapa apresenta conflitos entre 26 RNFs.

A etapa posterior foi a tentativa de normalizar esses RNFs encontrados e inseridos no catálogo. A normalização é o processo de remoção dos RNFs irrelevantes, ou seja, os tipos de RNFs que não têm definição ou atributos, do catálogo inicial. O objetivo é produzir um catálogo de conflitos dos tipos de RNFs bem definidos.

3.3 *Conflicts Identification among Non-functional Requirements using Matrix Maps*

O trabalho de Abdul *et al.* (2010) apresenta um modo de identificar conflitos entre RNFs através do mapeamento de matrizes. O modelo proposto por Abdul não apenas identifica os conflitos entre os RNFs, como também realiza a análise destes, mapeando os atributos de qualidade conflitante utilizando matrizes.

Figura 3 – Etapas do trabalho



Fonte: Adaptada de Abdul *et al.* (2010)

A Figura 3, mostra o modelo proposto e suas etapas. O modelo se inicia com uma

representação estruturada de RNFs de alto nível, então a identificação de conflitos é realizada para atributos de qualidade de alto nível, para assim conseguir identificar os conflitos de baixo nível. Feito isso, os atributos conflitantes de baixo nível são então mapeados para funcionalidades de baixo nível em ordem para identificar os potenciais conflitos.

Após a decomposição das funcionalidades de alto nível e atributos de qualidade, os conflitos são identificados utilizando uma matriz de qualidade. A partir daí os atributos de qualidade de baixo nível, tendo como base os RNFs selecionados, são mapeados para funcionalidades de baixo nível durante o processo de hierarquia das funcionalidades. A partir daí são identificados os conflitos potenciais entre eles.

Como principais resultados deste trabalho foi identificado que o modelo proposto não apenas identifica corretamente os potenciais conflitos dos requisitos de produtos, mas também identifica os conflitos dos requisitos baseados no processo. Além disso, devido ao fato da identificação dos conflitos terem um problema com a questão dos falsos conflitos, o modelo também faz a identificação destes, por meio da análise mapeada das funcionalidades de baixo nível. Este modelo também leva menos sobrecarga de documentação se comparado a outros modelos propostos, visto que reduz um bom número de diagramas e tabelas, além de ser de simples implementação e manter uma boa transparência dos conflitos.

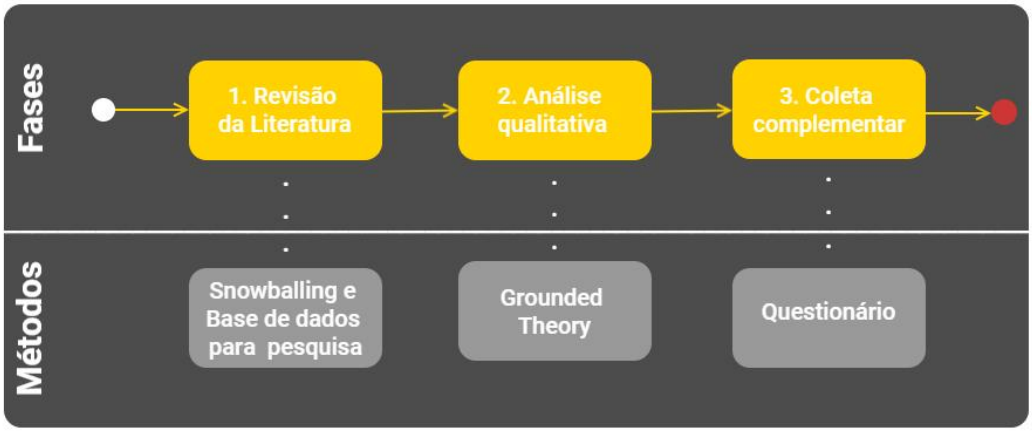
3.4 *Catalog of Invisibility Requirements for UbiComp and IoT Applications*

O trabalho de Carvalho *et al.* (2018) apresenta os requisitos de invisibilidade para sistemas UbiComp e IoT. As autoras utilizaram o NFR *framework*, bem como o conceito de SIG, para representá-los graficamente. O processo realizado durante este trabalho consiste em 3 principais fases ilutradas na Figura 4. As autoras utilizam métodos de pesquisa bem definidos para sistematizar a definição de Invisibilidade. Além de uma revisão da literatura, foi proposto o uso de um método de análise qualitativa de dados, *Grounded Theory* (Grounded Theory (GT)), para investigar os dados obtidos na literatura.

Esse método constrói teoria a partir de dados de forma sistemática, extraindo e agrupando conceitos derivados de dados, construindo um conhecimento pirâmide, que as autoras julgaram adequado para definir um SIG. Além do SIG, houve um complemento com soluções específicas extraídas de questionários enviados a desenvolvedores experientes de UbiComp e Aplicativos IoT.

Foram realizadas pesquisas para a identificação de catálogos existentes para RNFs de

Figura 4 – Metodologia utilizada



Fonte: Adaptada de Carvalho *et al.* (2018)

computação Ubíqua, porém as autoras identificaram a falta de precisão na descrição das etapas de ambos. A partir disto elas utilizaram sua própria metodologia para realizar a definição do SIG, baseadas nestes resultados. A seguir, as 3 fases mostradas na Figura 4 são explicadas com mais detalhes.

A primeira fase consiste em realizar uma revisão da literatura a fim de juntar o máximo de informações possíveis sobre o RNF, assim será possível realizar a decomposição deste requisito com o maior número de informações possíveis e com mais fidelidade. Foi utilizada a estratégia *Snowballing* para realizar esta coleta de dados. Para que este processo possa ser iniciado é necessário obter uma gama de artigos iniciais e a partir daí a coleta será iniciada, para isso foi escolhido o banco de pesquisa *Google Scholar*.

Na segunda fase uma análise qualitativa foi iniciada, utilizando o método *Grounded Theory* (GT). Esse método foi utilizado para a compreensão dos dados, extração e relacionamento de conceitos para que seja possível criar o SIG. Em geral, o GT é composto das seguintes etapas: planejamento, coleta de dados, codificação e relatórios de resultados. O planejamento visa identificar a área de interesse e a pesquisa em questão que irá impulsionar o trabalho. A etapa de coleta de dados visa coletar dados necessários para responder a pergunta de pesquisa. A etapa de codificação é a principal do GT, nela são extraídos os conceitos de dados brutos e é realizado o relacionamento deles, até chegar a um conceito central. E por último, a etapa de relatório dos resultados que consiste em, como o próprio nome diz, documentar os resultados obtidos, bem como as anotações e observações comentadas para um possível uso futuro.

No final desta segunda etapa as autoras enviaram os resultados para especialistas da Interação Humano–Computador (Interação humano computador (IHC)) para que os dados

fossem avaliados, assim é possível obter mais segurança e uma segunda visão sobre o quer foi coletado. Caso o pesquisador perceba a necessidade de melhorias, o processo de GT pode reiniciar.

A terceira fase consiste em uma coleta complementar, para refinar conceitos genéricos e amplos em metas flexiveis e mais específicas. Logo, o SIG gerado foi complementado com metas específicas de operacionalização, advindas da experiência de desenvolvedores de aplicativos IoT. Carvalho *et al.* (2018) optaram por utilizar a técnica de questionário para realizar esta coleta.

Como principais contribuições as autoras relatam que este catálogo gerado tem implicação direta na melhor compreensão e preenchimento da problemática da falta de taxonomia bem definida para a Invisibilidade. A metodologia utilizada neste trabalho também é citada como uma grande contribuição, visto que os estudos existentes não possuem detalhes claros sobre como é realizada a identificação e relacionamento entre os conceitos. Além disto, a metodologia utilizada para gerar o SIG é ampla, e foram realizadas também avaliações externas, logo o SIG resultante é considerado flexível o suficiente para ser refinado em trabalhos futuros.

3.5 Análise comparativa

Nesta Seção é apresentada uma comparação entre este trabalho e os trabalhos relacionados elencados neste capítulo. Foram utilizados 4 critérios para avaliar os trabalhos, como mostra o Quadro 1. O primeiro critério busca verificar se o artigo especifica um RNF (ou um conjunto deles) para trabalhar. Este critério está relacionado à etapa inicial de cada trabalho, quando é revisado como os autores chegaram a decisão de trabalhar com os requisitos citados no trabalho. O segundo critério identifica como o artigo detecta os conflitos abordados no trabalho, por meio do qual é avaliado se o autor deixa definido e bem detalhada a metodologia utilizada e quais conflitos foram obtidos ao final da etapa. O terceiro critério verifica se o artigo cataloga os resultados encontrados, verificando se o autor está realizando um catálogo ou algum tipo de documento especificando os conflitos encontrados entre os RFNs. Por fim, o quarto e último critério é busca identificar se o artigo pertence ao escopo dos SAS.

Quadro 1 – Comparativo entre os trabalhos relacionados e o trabalho proposto

Trabalhos	**Especifica um RNF (ou um conjunto deles) para trabalhar**	**Detecta as interações e possíveis conflitos**	**Pertence ao escopo dos SASs**
Trabalho Proposto	Sim	Sim	Sim
Martinez *et al.* (2019)	Sim	Sim	Não
Mairiza e Zowghi (2010)	Não	Sim	Não
Abdul *et al.* (2010)	Não	Sim	Não
Carvalho *et al.* (2018)	Sim	Não	Não

Fonte: Elaborado pela autora.

3.6 Relação entre os trabalhos

Nos trabalhos dos autores Mairiza e Zowghi (2010) e Abdul *et al.* (2010) foram encontradas limitações voltadas à metodologia de especificação dos RNFs trabalhados. Em ambos os trabalhos não ficou claro como foi realizada essa especificação ou quais RNFs foram trabalhados exatamente. No trabalho de Abdul *et al.* (2010), foram citados de forma sutil alguns RNFs trabalhados, porém de acordo com o descrito, outros requisitos além destes foram catalogados. Apesar disto, ambos os trabalhos mostram as metodologias utilizadas para o processo de detecção dos conflitos entre os RNFs, bem como o modo de catalogação dos resultados.

O trabalho de Martinez *et al.* (2019), ao contrário dos anteriores, especifica detalhadamente quais RNFs foram utilizados ao longo do trabalho. Foram utilizados 16 RNFs ao final de todo o processo de escolha, os autores deixam claro que no número não poderia ser maior visto que catalogar todos os RNFs existentes seria um trabalho exaustivo e praticamente impossível. Porém, este trabalho não cataloga os resultados obtidos, portanto é realizado um mapeamento de conflitos, mas não é gerado nenhum catálogo nem documento semelhante mostrando detalhadamente os resultados.

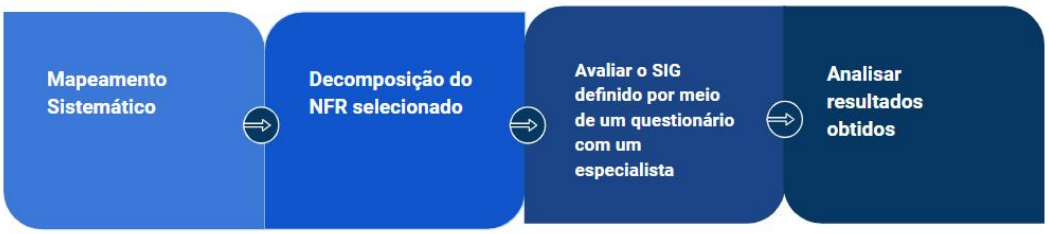
O trabalho de Carvalho *et al.* (2018) possui todos os processos e passos bem definidos e documentados. Desde a especificação até o processo de catalogação, todos os dados e metodologias utilizadas são bem especificados e este trabalha estuda um SIG de invisibildiade para aplicativos IoT e sistemas ubíquos. Há uma característica comum a todos os trabalhos: nenhum deles é do escopo dos SAS. Tendo isso em vista, o trabalho proposto tem como objetivo atingir este escopo. A seleção do RNF a ser trabalhado tomará como base um mapeamento sistemático que avaliará, do ponto de vista da literatura, qual RNF mais causa impacto nos

sistemas autoadaptativos, e avaliará também o estado da literatura a cerca do uso dos RNFs nestes sistemas. As fases de elaboração do SIG terão como base a metodologia do trabalho de Carvalho *et al.* (2018).

4 PROCEDIMENTOS METODOLÓGICOS

Esta seção apresenta os procedimentos metodológicos deste trabalho. As quatro seções a seguir representam as etapas de desenvolvimento. A Seção 4.1 discorre sobre os requisitos em SAS, relatando sobre o mapeamento sistemático realizado durante o processo. A Seção 4.2 apresenta a decomposição do RNF selecionado. A Seção 4.3 demonstra a avaliação do SIG definido por meio da aplicação de um questionário a um especialista do domínio dos SAS. Por fim, a Seção 4.4 discorre sobre os resultados que serão obtidos durante a avaliação do SIG. A Figura 5 demonstra graficamente essas atividades.

Figura 5 – Diagrama de fluxo dos Procedimentos Metodológicos.



Fonte: Retirada de Chung *et al.* (2012)

4.1 Mapeamento Sistemático

Nesta seção são descritas as etapas adotadas no mapeamento sistemático realizado, utilizado para explorar como os RNFs estão sendo trabalhados e desenvolvidos de acordo com os aspectos dinâmicos dos SAS. Para a realização do mapeamento, o trabalho de Keele *et al.* (2007) foi tomado como base. Este trabalho apresenta um mapeamento sistemático a fim de descobrir o estado da literatura envolvendo os RNFs no domínio dos SAS. Ao final deste mapeamento desenvolvido, o resultado obtido será um conjunto de RNFs pertinentes à SAS e a partir disso, será escolhido qual requisito será utilizado para decomposição.

4.2 Decompor o RNF selecionado

Nessa fase, iniciará o processo de decomposição do RNF que será selecionado após o fim do mapeamento. O trabalho de Carvalho *et al.* (2018) será utilizado como apoio para a realização desta etapa. A ISO 25010 (ISO, 2011) será utilizada para agrupar os conceitos e as subacaracterísticas do RNF que será decomposto, e o mapeamento sistemático realizado neste trabalhado será utilizado como base para agrupar informações acerca do RNF confiabilidade e

suas interações com as características próprias que os SASs possuem.

Após ter finalizado o processo de escolha do RNF a ser realizado neste trabalho, a decomposição dele será realizada, dando início a fase de construção do SIG. O trabalho utilizará o NFR *Framework* durante este processo, a ferramenta RE-tools[1] e o StarUML[2].

4.3 Avaliar o SIG

Essa seção discorre sobre a avaliação do SIG criado e apresentado na fase anterior. O objetivo dessa avaliação é verificar a viabilidade e aplicabilidade do SIG definido, bem como seus potenciais benefícios.

Para a realização da entrevista um especialista do domínio dos SAS foi escolhido para fazer parte de uma reunião de avaliação. Um *checklist* de 10 questões foi elaborado e ao longo da reunião essas perguntas foram feitas ao especialista. Para que a visualização dos dados obtidos seja feita de uma forma organizada e interessante graficamente, serão construídas planilhas para demonstrar as perguntas realizadas e as respostas obtidas. As conclusões obtidas com esta etapa serão relevantes, pois possibilitarão uma melhor visualização e entendimento de como o trabalho poderá ajudar, em quais aspectos especificos ele pode ser implementado e quais problemas irá sanar.

4.4 Analisar os resultados obtidos

Nessa seção, são apresentados os resultados obtidos por meio da avaliação realizada na fase anterior, onde um *checklist* com 10 perguntas foi enviado a um especialista do domínio dos SAS. As perguntas e suas respectivas respostas serão apresentadas e discutidas, bem como as sugestões ou comentários pertinentes realizados pelo avaliador. Os resultados obtidos através desta avaliação serão analisados e utilizados para alterar o SIG de acordo com as melhorias sugeridas. Vale ressaltar que os resultados obtidos não podem ser generalizados, uma vez que a coleta foi realizada em período curto de tempo e com um único especialista no domínio.

[1] https://personal.utdallas.edu/ chung/Sam$\mathit{supakkul}/RE-Tools/index.html$

[2] https://staruml.io/

5 MAPEAMENTO SISTEMÁTICO DE RNFS PARA SISTEMAS AUTOADAPTATIVOS

Neste capítulo será apresentado todo o precesso, execução e resultados do mapeamento sistemático realizado neste trabalho. A Seção 5.1 discorre sobre o conceito de mapemanto sistemático e como executá-lo. A Seção 5.2 apresenta todo o processo de planejamento do mapemaento. Dados como questões de pesquisa e critérios de inclusão e exclusão foram decididos nesta etapa. A Seção 5.3 discorre sobre a execução do mapeamento, todas as etapas planejadas foram apresentadas nesta seção. A Seção 5.4 apresenta uma análise dos resultados do mapeamento sistemático. E por último, a Seção 5.5 apresenta uma conclusão dos resultados do mapeamento sistemático.

5.1 Introdução

De acordo com Keele *et al.* (2007), mapeamentos sistemáticos são realizados com o objetivo de obter informações gerais sobre determinadas áreas ou tópicos de pesquisa. Dessa forma, é possível realizar identificação, análise, e interpretação dos dados coletados correspondentes às questões de pequisa que foram formuladas.

Os Mapeamentos Sistemáticos são compostos pelas fases de: Planejamento, Execução e Análise dos Resultados. Na fase de planejamento todos os elementos que serão importantes e terão um impacto direto na execução do mapeamento serão especificados. A fase de execução se concentra na busca e seleção dos estudos, bem como a extração dos dados. E por ultimo, a fase de análise dos resultados, que tem como objetivo sintetizar os resultados que foram obtidos (KEELE *et al.*, 2007)

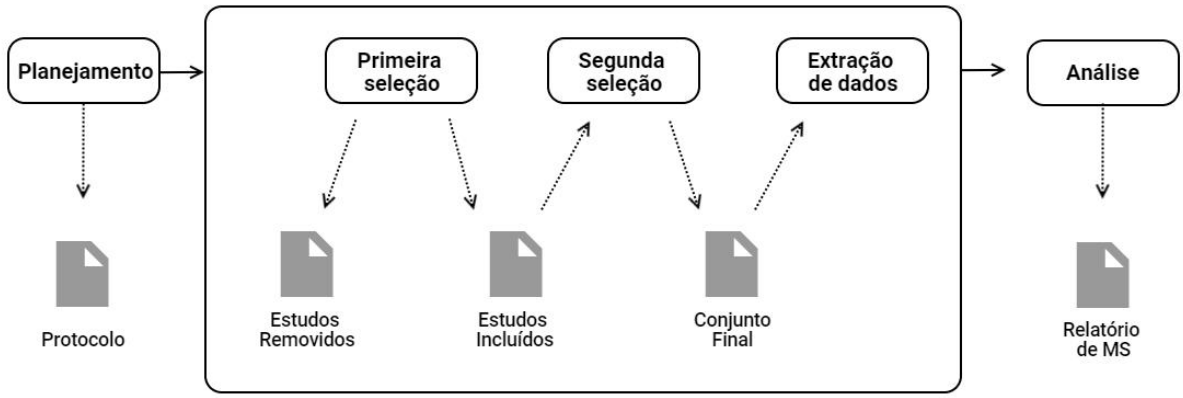
O mapeamento apresentado nesta seção foi adaptado do processo realizado por Keele *et al.* (2007) e foi conduzido dentro da ferramenta Parsifal[1]. A Figura 6 apresenta as etapas do processo adotado.

5.2 Planejamento

Nesta etapa, foram definidos os objetivos de pesquisa e o protocolo do mapeamento sistemático. No protocolo do mapeamento sistemático, foram definidos: objetivos e questões de pesquisa; estratégia de busca; critérios de seleção; procedimentos para a seleção de estudos; e,

[1] https://parsif.al/

Figura 6 – Etapas do mapeamento sistemático



Fonte: Adaptado de Keele *et al.* (2007)

extração e análise dos dados.

O objetivo deste mapeamento é fornecer uma visão da literatura acerca dos RNFs e do processo de requisitos para os sistemas autoadaptativos. Para extrair estas questões, foi utilizada a metodologia PICOC. A estratégia PICOC é um acrônimo para: *Population* (população), *Intervention* (intervenção), *Comparison* (comparação), *Outcome* (desfecho) e *Context* (contexto). Esses quatro componentes são os elementos fundamentais das questões de pesquisa e da construção da string para a busca bibliográfica de evidências (WYATT; GULY, 2002). Neste sentido, foram definidas as seguintes questões de pesquisa (QPs):

- **QP1:** Quais os desafios enfrentados ao implementar RNFs em SAS?
- **QP2:** Quais soluções tem sido propostas para lidar com os desafios ao implementar RNFs em SAS?
- **QP3:** Quais os RNFs, características ou subcaracterísticas de qualidade recorrentes em SAS?

Para conseguir responder às questões acima, foram considerados os seguintes tipos de estudos:

- Soluções que tratam dos RNFs, características ou subcaracterísticas de qualidade dentro do escopo dos SAS.
- Soluções que realizam a condução/identificação de metodologias para apoiar o processo de requisitos em SAS.

Ainda seguindo o raciocínio da estratégia PICOC, foram definidas as seguintes palavras chaves: “Sistemas Autoadaptativos”, “Características de Qualidade”, “Atributos de Qualidade” e “Requisitos Não Funcionais”. Além disso, sinônimos e termos relacionados a estas

palavras chaves foram considerados.

Os operadores OR e AND foram utilziados na construção de uma *string* de busca contendo as palavras chaves, bem como seus sinônimos e termos relacionados. Foi utilizada a língua inglesa, devido ao maior nível de conteúdo disponível nos meios de publicação. A seguinte *string* foi construída:

(“self-adaptive” OR “SAS” OR “software adaptive”) AND (“non-functional requirement” OR “quality characteristic” OR “quality attribute” OR “NRF”)*

Além disto, é necessário selecionar também as fontes de pesquisa apropriadas para aplicar as estratégias escolhidas. No presente trabalho, optou-se por realizar um mapeamento exaustivo, com o objetivo de obter uma grande quantidade de trabalhos. Por esse motivo, foram selecionadas 5 bases de dados: *ACM Library*, *SpringerLink*, *IEEE Xplorer*, *ScienceDirect* e *Wiley*.

Para algumas bases de dados foi necessário efetuar mudanças na *string* utilizada, pois os trabalhos retornados estavam fugindo do escopo requerido. Para a base de dados IEEE *Digital Library* a seguinte *string* foi utilizada:

(“self-adaptive” OR “SAS” OR “software adaptive”) AND (“non-functional requirement” OR “quality characteristic” OR “quality attribute” OR “NRF“) AND (software)*

Para as bases de dados *Science Direct* e *Springer Link*, a seguinte *string* foi utilizada:

(“self-adaptive” OR “SAS” OR “software adaptive”) AND (“non-functional requirement” OR “quality characteristic” OR “quality attribute” OR “NRF”) AND (software) em conjunto com os filtros *Research Articles* e *Computer Science*.

Nesta fase, também foram definidos os critérios de seleção dos artigos que foram coletados em cada base de dados. Esses critérios foram utilizados durante o período de aceitação ou recusa dos trabalhos coletados. Assim, foi possível incluir estudos importantes para responder às questões de pesquisa, bem como excluir estudos irrelevantes. Os critérios de inclusão e exclusão são citados a seguir:

- **INC1:** Artigos que descrevem abordagens de requisitos no escopo dos SAS.
- **INC2:** O resumo do trabalho cita o processo de engenharia de requisitos voltado para escopo dos SAS.
- **INC3:** O resumo do trabalho cita RNFs ou atributos de qualidade dentro do escopo dos SAS.

A especificação desses critérios de inclusão foi realizada com a intenção de capturar

trabalhos da literatura que propusessem ou estudassem a questão dos RNFs durante o período de desenvolvimento dos sistemas autoadaptativos. O período de tempo levado em consideração para filtrar os artigos foi de 1990 a 2021. Este filtro foi implementado com o intuito de agregar uma maior quantidade de trabalhos ao longo dos anos.

Quanto às questões de exclusão, as seguintes questões foram definidas:

- **EXC1:** O artigo está fora do escopo de RNFs/atributos de qualidade em sistemas autoadaptativos.
- **EXC2:** Artigos com menos de duas páginas.
- **EXC3:** Artigos que não estão em inglês.
- **EXC4:** Teses de doutorado ou dissertações de mestrado .
- **EXC5:** Estudos secundários.

A escolha destes critérios tiveram a finalidade de remover os trabalhos que não apresentam o escopo requerido, ou não estão dentro da área de estudo dos RNFs. Como resultado, foi obtido um conjunto de estudos possivelmente relacionados ao tópico de pesquisa. Foram lidos os títulos, resumos e palavras-chave e os critérios de inclusão e exclusão foram aplicados. Como resultado, um conjunto de estudos primários potencialmente relevantes foi definido.

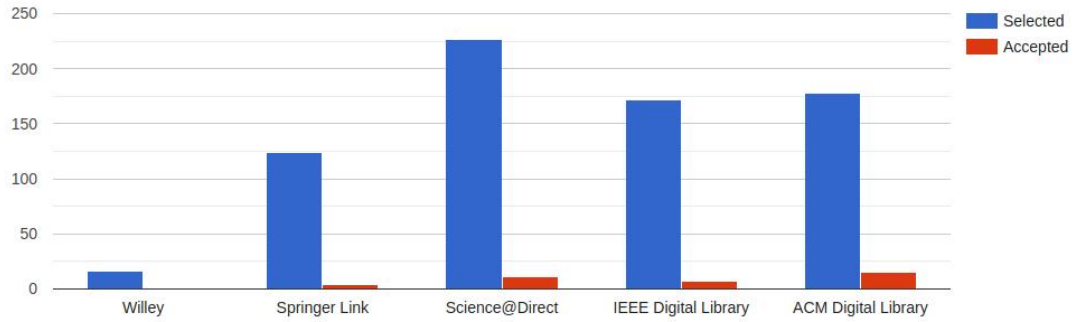
Posteriormente, os artigos selecionados foram enviados para a ferramenta *Parsifal*. Desta forma, foi possível realizar uma análise de todos os trabalhos coletados de uma maneira mais rápida e menos burocrática. Os critérios de inclusão e exclusão foram utilizados neste processo e inicialmente foi analisado o resumo dos trabalhos para realizar a aprovação ou exclusão. Se a decisão sobre a inclusão ou exclusão de determinado estudo não estava clara, o estudo era analisado completamente.

5.3 Execução

O mapemaento sistemático foi iniciado em 20 de Junho de 2021 e finalizado no dia 05 de dezembro. Na primeira seleção, foi utilizada a string de busca previamente adaptada para cada base de dados. As bases de dados utilizadas para obter os trabalhos foram: *ACM Library*, *SpringerLink*, *IEEE Xplorer*, *ScienceDirect* e *Wiley*. Ao todo 706 estudos foram retornados.

Nesse conjunto alguns documentos estavam duplicados, portanto, foram removidos nesta primeira seleção. Ao todo somaram 164 trabalhos duplicados. Após isso, sobraram 542 trabalhos para serem analisados.

Figura 7 – Estudos por base de dados



Fonte: Elaborado pela autora

Para apoiar o gerenciamento dos dados foi utilizada a ferramenta Parsifal, lá foram mantidas todas as informações dos artigos, bem como todo o processo de planejamento e execução do mapeamento. Como segundo resultado obtido nesta seleção, realizada de acordo com os critérios descritos nas questões de inclusão, foi obtido um conjunto de 81 artigos para uma inspeção mais detalhada.

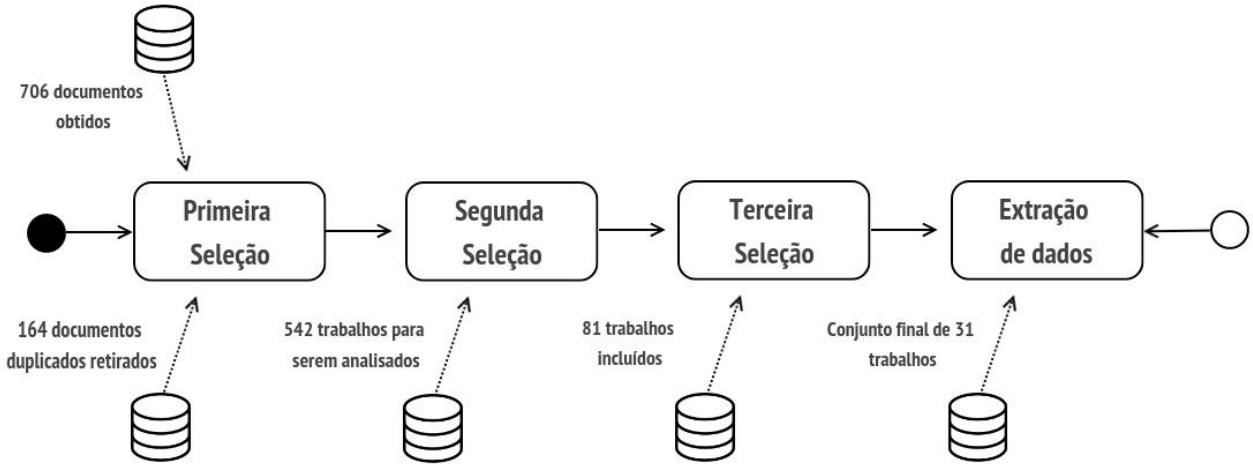
A exclusão dos documentos obtidos se deu por meio da utilização dos critérios de exclusão apresentados, e a quantidade de documentos removidos por cada critério é listada abaixo:

- **EXC1:** O artigo está fora do escopo de RNFs/atributos de qualidade em sistemas autoadaptativos: 404
- **EXC2:** Artigos com menos de duas páginas: 0
- **EXC3:** Artigos que não estão em inglês: 0
- **EXC4:** Teses de doutorado ou dissertações de mestrado: 0
- **EXC5:** Estudos secundários: 57

Após esta etapa, um processo de leitura dos artigos selecionados foi iniciado, e a partir disto mais artigos foram descartados caso o seu conteúdo não apresentasse as informaçoes necessárias. Com o final desta terceira etapa de exclusão sobraram 31 artigos para o processo de extração de dados.

Foi observado que o número de estudos foi reduzido em uma grande proporção, isto se deu pois foram obtidos muitos estudos relacionados ao tema, mas a grande maioria não trazia em sua contribuição nada relacionado a RNFs ou atributos de qualidade no escopo dos SAS.

Figura 8 – Processo de seleção dos estudos



Fonte: Elaborado pela autora

5.4 Análise dos resultados

Na fase de análise os resultados foram agrupados por questões de pesquisa, e foi realizada uma inspeção dos artigos com o objetivo de obter informações sobre o estado da literatura voltada para a problemática dos RNFs em SAS. O conjunto final de artigos é apresentado na Tabela 3, por limitação de espaço os nomes das bases foram abreviados: ACM Digital Library (ACM), SpringerLink (SL) IEEEXplorer (IEEE) e ScienceDirect (SD). O Apêndice A contém uma tabela que apresenta todos os artigos retornados neste mapeamento.

5.4.1 QP1 - Quais desafios enfrentados no desenvolvimento de requisitos em SAS?

Essa questão de pesquisa busca investigar quais os principais desafios encontradas ao lidar com RFNs em sistemas autoadaptativos. O Quadro 2 ilustra os principais desafios identificados e as referências dos artigos selecionados em que os desafios foram extraídos.

Quadro 2 – Conjunto Final de desafios encontrados

Desafio	**Referências**
Desafio 1	(D'ANGELO *et al.*, 2020), (BOWERS *et al.*, 2019), (PéREZ; CORREAL, 2011), (ALI *et al.*, 2017), (ZHAO *et al.*, 2013)
Desafio 2	(D'ANGELO *et al.*, 2020), (GHEZZI; SHARIFLOO, 2013), (ROSA *et al.*, 2013), (MAHDAVI-HEZAVEHI, 2016),(AHMAD *et al.*, 2015), (PENG *et al.*, 2012), (HEZAVEHI *et al.*, 2017), (EDWARDS; BENCOMO, 2018), (ZHAO *et al.*, 2013)
Desafio 3	(FILIERI *et al.*, 2011), (D'ANGELO *et al.*, 2020), (CHENG *et al.*, 2009), (GHEZZI; SHARIFLOO, 2013), (AHMAD *et al.*, 2015), (BOWERS *et al.*, 2019), (PéREZ; CORREAL, 2011), (SOUSA *et al.*, 2019a), (ZHAO *et al.*, 2013), (PERUMA; KRUTZ, 2018)
Desafio 4	(GHEZZI; SHARIFLOO, 2013), (PéREZ; CORREAL, 2011), (LUCKEY; ENGELS, 2013), (ZHAO *et al.*, 2013), (ZHANG; CHENG, 2006), (ZHANG; CHENG, 2006)
Desafio 5	(EDWARDS; BENCOMO, 2018), (ZHAO *et al.*, 2013)
Desafio 6	(ZHAO *et al.*, 2013)

Fonte: Elaborado pela autora.

Os desafios identificados são descritos a seguir:

- **Desafio 1:** O domínio do problema de um SAS geralmente não pode ser totalmente compreendido na fase de desenvolvimento. Isso ocorre porque no momento do desenvolvimento não podemos antecipar todo o conjunto de condições ambientais possíveis. Como resultado, não é possível antecipar todos os comportamentos de adaptação possíveis de SAS.
- **Desafio 2:** Um SAS muitas vezes executa em um ambiente com mudanças imprevisíveis. Além disso, sistemas autoadaptativos estão sujeitos a questões de incerteza que podem surgir a qualquer momento.
- **Desafio 3:** Os requisitos de sistemas autoadaptativos podem precisar mudar e evoluir em tempo de execução em resposta ao ambiente em mudança. Isso é causado principalmente pela falta de conhecimento na fase de desenvolvimento e devido a incerteza associada ao ambiente desconhecido.
- **Desafio 4:** Os sistemas autoadaptativos precisam se adaptar às suas propriedades, eles tem a capacidade de modificar seu comportamento e estrutura quando se observa divergências entre os requisitos. Tal comportamento de adaptação inclui principalmente autocura, autoconfiguração e autootimização.
- **Desafio 5**: *Trade-offs* entre os requisitos de um sistema autoadaptativo são deslocados para tempo de execução. Isso ocorre pois quando o contexto mudar, caso exista a possibilidade de vários requisitos entrarem em conflito, o comportamento de fazer decisões entre esses requisitos será deslocado para tempo de execução.
- **Desafio 6:** Devido ao alto grau de incerteza contido em sistemas autoadaptativos, certas atividades de desenvolvimento e mudança são mudados do tempo de desenvolvimento para o tempo de execução. O que causa uma certa fronteira entre o tradicional tempo de desenvolvimento e tempo de execução de sistemas normais.

Com esses dados é possível inferir que boa parte das problemáticas tem ligação com o alto nível de incerteza que estes sistemas possuem devido à sua característica de realizar modificações em tempo de execução. Uma característica que causa impacto no funcionamento e interação entre os requisitos e que também é muito importante para o funcinamento desses sistemas em geral.

5.4.2 QP2 - Quais soluções tem sido propostas para requisitos em SAS?

Essa questão de pesquisa busca investigar quais métodos vem sendo utilizados para implementar RNFs em SAS. Além disso, essa questão também tem o objetivo de idetificar se há modos específicos de especificação de RNFs para SAS, visto que devido às características próprias que esses sistemas devem possuir, talvez seja necessário uma abordagem diferenciada. O Quadro 3 mostra todos os artigos que retornaram soluções para problemas envolvendo RNFs em SAS. Os estudos foram classificados de acordo com as categorias propostas por Petersen *et al.* (2008) e são definidas abaixo.

- **Pesquisas de avaliação** As técnicas são implementadas na prática e a avaliação da técnica é realizada, ou seja, é mostrado como a técnica é implementada na prática (implementação da solução) e quais são as consequências da implementação em termos de benefícios e desvantagens (avaliação da implementação). Isso também inclui identificar problemas na indústria.
- **Pesquisas de validação** Nas pesquisas de validação as técnicas investigadas são novas e ainda não foram implementadas na prática. As técnicas utilizadas são, por exemplo, experimentos.
- **Propostas de solução** Nas proprostas de solução uma solução para um problema é proposta, a solução pode ser uma novidade ou uma extensão significativa de uma técnica existente. Os benefícios potenciais e a aplicabilidade da solução são demonstrados por um pequeno exemplo ou uma boa linha de argumentação.
- **Trabalhos filosóficos** : Os trabalhos fislosóficos esboçam uma nova forma de olhar para as coisas existentes, estruturando o campo em forma de uma taxonomia ou estrutura conceitual.
- **Relatos de experiência** Os trabalhos de relatos de experiência explicam o que e como algo foi feito na prática. O artigo deve tratar da experiência pessoal do autor.
- **Trabalhos opinativos** : Os trabalhos opinativos são trabalhos que expressam a opinião pessoal de alguém sobre uma determinada técnica, se é uma técnica boa ou ruim, ou como as coisas devem ser feitas. Esses trabalhos não dependem de trabalhos relacionados ou metodologias de pesquisa.

Com base no Quadro 3 pode-se inferir que a maior quantidade de estudos está classificada como proposta de solução, o que aponta que a qualidade de SAS tem sido estudada, porém as técnicas propostas não tem sido avaliadas em sistemas reais ou industriais.

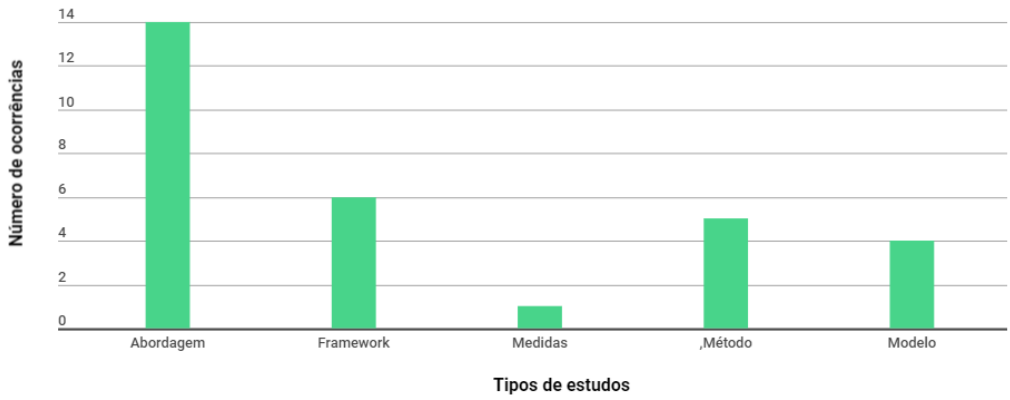
Quadro 3 – Conjunto Final das soluções encontradas

Categoria de pesquisa	**Artigos**
Pesquisas de avaliação	(ROSA *et al.*, 2013), (HORCAS *et al.*, 2015), (BOWERS *et al.*, 2019), (PENG *et al.*, 2012), (WU *et al.*, 2010)
Trabalhos opinativos	(PERUMA; KRUTZ, 2018)
Propostas de solução	(FILIERI *et al.*, 2011),(D'ANGELO *et al.*, 2020), (GHEZZI; SHARIFLOO, 2013), (AHMAD *et al.*, 2015), (PéREZ; CORREAL, 2011), (EDWARDS; BENCOMO, 2018), (GONZALEZ-SANCHEZ, 2013), (GOLDSBY *et al.*, 2008), (NETI; MULLER, 2007), (ALI *et al.*, 2017), (RAGO *et al.*, 2017), (ANGELOPOULOS *et al.*, 2018), (VILLEGAS *et al.*, 2011), (SUBRAMANIAN; CHUNG, 2001), (GHEZZI *et al.*, 2013), (PéREZ; CORREAL, 2011), (ZHANG; CHENG, 2006)
Pesquisas de validação	(CHENG *et al.*, 2009), (KOZIOLEK *et al.*, 2013), (LUCKEY; ENGELS, 2013) (ZHAO *et al.*, 2013)

Fonte: Elaborado pela autora.

A Figura 9 apresenta o número de estudos por tipo de contribuição, com esses dados é possível verificar que os resultados retornaram poucos estudos que propõem medidas. Em contra partida, grande parte dos estudos retornados apresentam soluções em forma de abordagens. Estudos que propõem *frameworks* foram o segundo mais retornados.

Figura 9 – Tipos de estudos selecionados



Fonte: Elaborado pela autora

5.4.3 QP3 - Quais os RNFs, características ou subcaracterísticas de qualidade em SAS?

Essa questão de pesquisa busca selecionar os RNFs, características ou subcaracterísticas de qualidade que tem sido citados nos trabalhos selecionados. Foi observado que a maioria dos artigos não faz uso dos termos apresentados pela norma ISO (ISO, 2011). Ao invés disso, os autores se referem a RNFs, características ou subcaracterísticas de qualidade utilizando termos genéricos, como fatores de qualidade, critérios de qualidade, objetivos de qualidade.

Devido a isso, após a extração de dados, foi necessário realizar a identificaçao das

características e subcaracterísticas de qualidade apresentados na norma ISO (ISO, 2011). Em seguida, as características e sub-características de qualidade foram analisadas e agrupadas de acordo com o modelo apresentado pela norma. Não foi levado em consideração os atributos de qualidade, apenas as características e subcaracterísticas de qualidade.

Todas as 8 características citadas na norma foram identificadas ao longo da análise deste mapeamento. Quanto as subcaracterísticas citadas na norma, 13 delas foram identificadas também. O Quadro 4 ilustra o resultado obtido.

Quadro 4 – Conjunto de características e suas respectivas subcaracteristicas identificadas nos estudos

Características	**Subcaracterísticas**
Adequação funcional	Nenhuma
Eficiência de desempenho	Capacidade e comportamento em relação ao tempo
Compatibilidade	Interoperabilidade
Usabilidade	Acessibilidade
Confiabilidade	Tolerância a falhas, recuperabilidade e disponibilidade
Segurança	COnfidencialidade e integridade
Manutenibilidade	Modificabilidade
Portabilidade	Adaptabilidade

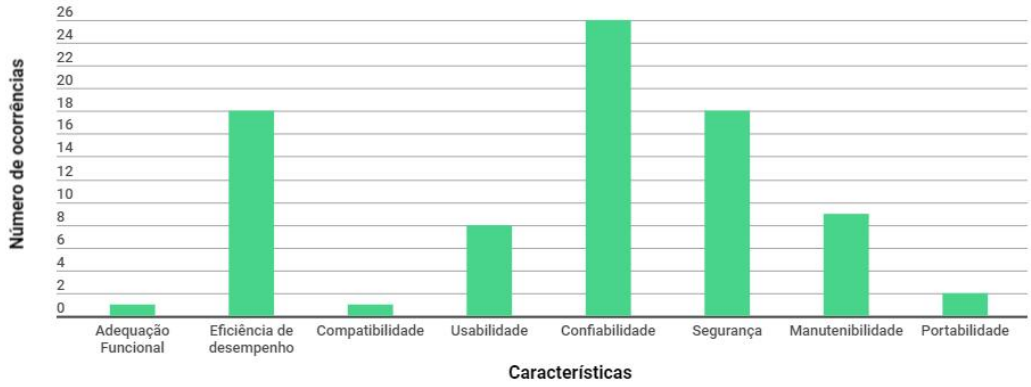
Fonte: Elaborado pela autora.

O Quadro 4 contém apenas os termos propriamente designados na norma ISO (2011). Apesar de ter sido necessário realizar a identificação das características para encaixá-las de acordo com a norma, as características e susbcaracteristicas mais recorrentes encontradas na literatura são aquelas providas pela norma ISO (2011).

Essa questão de pesquisa, também teve o propósito de identificar quais características ou subcaracterísticas foram mais avaliadas ou citadas, com a intenção de verificar a sua relevância na literatura. A Figura 10 mostra em quantos trabalhos cada característica foi citada. O Apêndice B contém uma tabela com as características encontradas e os artigos que retornaram cada uma delas.

Em relação as características de qualidade mais recorrentes pode-se destacar: confiabilidade, eficiência de desempenho e segurança, respectivamente. A subcaracterística de qualidade mais recorrente nos estudos selecionados foi a disponibilidade, o que reforça a importância da confiabilidade para SAS, uma vez que elas estão intrinsecamente relacionadas. A Figura 11 ilustra quantas subcaracterísticas foram encontradas. O Apêndice C contém uma tabela

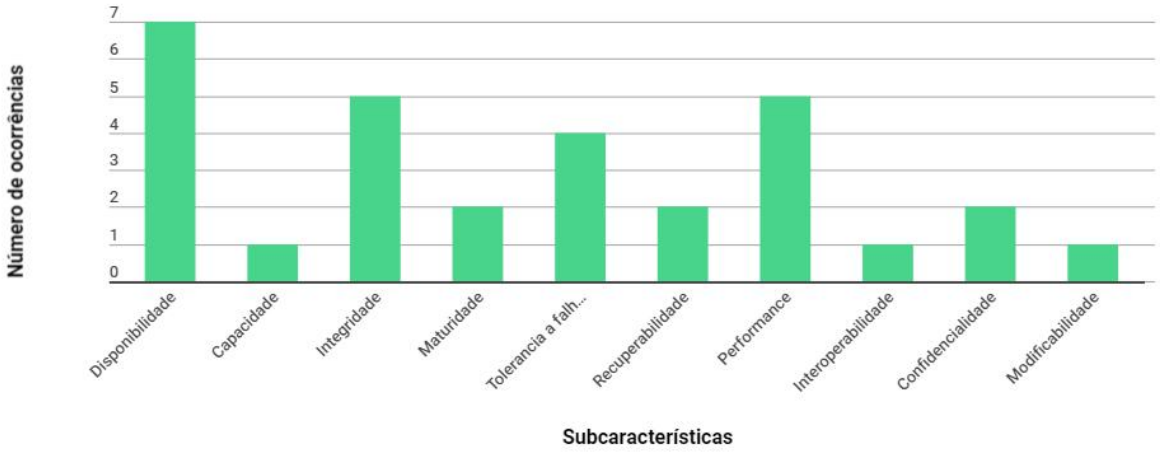
Figura 10 – Características mais recorrentes nos trabalhos selecionados



Fonte: Elaborado pela autora

com as subcaracterísticas encontradas e os artigos que retornaram cada uma delas.

Figura 11 – Subaracterísticas mais recorrentes nos trabalhos selecionados



Fonte: Elaborado pela autora

É importante lembrar que o termo eficiência de desempenho foi adotado neste trabalho, pois é o termo proposto na norma ISO 25010 (ISO, 2011). Nos estudos selecionados foi verificada a ocorrência do termo "desempenho/performance" apresentando a mesma definição que o termo apresentado pela ISO 25010. Portanto, neste trabalho, essas característica foi mapeada como "eficiência de desempenho".

Como citado anteriormente, o termo eficiência foi encontrado durante a leitura dos estudos selecionados, porém essa característica não está presente na ISO 25010 (ISO, 2011), utilizada como base para este trabalho, mas sim na norma ISO 25023 (ISO, 2016) e não deve ser confundida com eficiência de desempenho ou com suas subcaracterísticas. Visto que no modelo de qualidade em uso a eficiência está relacionada ao grau em que o sistema permite que

o usuário alcance seus objetivos de forma completa e precisa (ISO, 2016), enquanto a eficiência de desempenho diz respeito à propriedades computacionais (ISO, 2011).

Após estes resultados obtidos, foi decidido utilizar a característica confiabilidade para realizar a decomposição e gerar o SIG. Além desta característica ter sido a mais retratada nos trabalhos, ela também foi citada várias vezes como uma das características mais importamentes para esses sistemas, visto que ela influencia em muitos aspectos e características dos SASs.

5.5 Conclusão

Neste mapeamento sistemático foi obtido um conjunto de 31 artigos dos quais foram extraídos dados para investigação. Foi observado que a maioria dos autores não utilizou termos e modelos providos pela norma ISO (2011). Ao invés disso, os artigos apresentaram termos mais genéricos, dificultando um pouco a identificação imediata dos RNFs. Nos resultados, foram identificadas as 8 características de qualidade especificadas na norma ISO (2011) e 10 das subcaracterísticas listadas nela.

Também foi observado que as características de qualidade mais recorrentes nos estudos são as características apresentadas na norma ISO (2011), indicando que os pesquisadores tem recorrido aos modelos propostos pela norma. Uma das características de qualidade mais recorrentes nos estudos foi a confiabilidade, sendo citada várias vezes também como uma das características mais importamentes para esses sistemas, visto que ela influencia em muitos aspectos e características destes. Por meio da realização do estudo de mapeamento sistemático, foi observado que a confiabilidade tem uma influencia significante nas propriedades dos SAS, na performance do sistema, além de impactar a qualidade do serviço. Nesse sentido, para reforçar a avaliação da qualidade de SAS, o escopo deste trabalho se concentra na definição de um SIG de confiabilidade.

6 DECOMPOSIÇÃO DOS REQUISITOS DE CONFIABILIDADE PARA SISTEMAS AUTOADAPTATIVOS

Neste capítulo será apresentada a decomposição do requisito de confiabilidade para SAS, representado por um SIG. A Seção 6.1 discorre sobre o processo de construção do SIG, apresentando dados importates para a etapa de decomposição. A Seção 6.2 discorre sobre a execução da decomposicao do SIG. A Seção 6.3 discorre sobre como a avaliação do SIG foi planejada. E por último, a Seção 6.4 discorre sobre a execução da avaliação.

6.1 Processo de construção do SIG

Os SASs possuem características próprias que possuem relação direta com os RNFs. Sabendo disto, o mapeamento sistemático realizado foi utilizado como base para obter informações sobre como a confiabilidade se relaciona com as propriedades específicas de sistemas deste domínio, e quais são essas características ou subcaracterísticas. O RNF a ser decomposto será o de confiabilidade, pois ao fim do mapemanto sistemático observou-se que esta característica foi a mais citada nos trabalhos retornados. Além disso, esta característica tem uma influência significante nas propriedades dos SAS, na performance do sistema, além de impactar a qualidade do serviço.

Para realizar a decomposição utilizando dados dos artigos retornados no mapemaento, foram selecionados para análise todos os artigos que retornaram a característica de confiabilidade. O Quadro 5 apresenta os trabalhos analisados.

Quadro 5 – Características e subcaracterísticas autoadaptativas retornadas

Confiabilidade	(FILIERI *et al.*, 2011), (D'ANGELO *et al.*, 2020), (GHEZZI; SHARIFLOO, 2013), (BOWERS *et al.*, 2019), (AHMAD *et al.*, 2015), (KOZIOLEK *et al.*, 2013), (PéREZ; CORREAL, 2011), (EDWARDS; BENCOMO, 2018), (NETI; MULLER, 2007), (LUCKEY; ENGELS, 2013), (NURBOJATMIKO *et al.*, 2018), (PERUMA; KRUTZ, 2018), (ALI *et al.*, 2017), (RAGO *et al.*, 2017), (ANGELOPOULOS *et al.*, 2018), (SOUSA *et al.*, 2019a), (VILLEGAS *et al.*, 2011), (ZHAO *et al.*, 2013), (SUBRAMANIAN; CHUNG, 2001), (ZHANG; CHENG, 2006), (BOWERS *et al.*, 2019), (GOLDSBY *et al.*, 2008), (ROSA *et al.*, 2013)

Fonte: Elaborado pela autora.

Alguns trabalhos citam que baixa confiabilidade em SAS influencia negativamente no desempenho e no funcionamento correto do sistema e das propriedades de adaptação, mas nem todos citam exatamente quais propriedades dos SASs são afetadas por este RNF. Em contra partida, outros trabalhos citam exatamente quais características dos SASs são influenciadas pela confiabilidade. Para este trabalho autoadaptação foi considerada como uma nova característica de

qualidade específica dos SASs. E para essa característica, nós identificamos 3 subcaracterísticass ilustradas no quadro 6, com suas respetivas referências.

Quadro 6 – Características e subcaracterísticas autoadaptativas retornadas

Características retornadas	**Artigos**
Autoadaptação	(FILIERI *et al.*, 2011), (D'ANGELO *et al.*, 2020), (KOZIOLEK *et al.*, 2013), (SOUSA *et al.*, 2019a), (NETI; MULLER, 2007), (RAIBULET *et al.*, 2017), (ALI *et al.*, 2017), (GHEZZI *et al.*, 2013)
Subcaracterísticas retornadas	**Artigos**
Autocura	(SOUSA *et al.*, 2019a), (HEZAVEHI *et al.*, 2017), (GHEZZI *et al.*, 2013), (NETI; MULLER, 2007), (**??**), (ALI *et al.*, 2017)
Autoproteção	(GHEZZI *et al.*, 2013), (ALI *et al.*, 2017), (NETI; MULLER, 2007), (**??**)
Autoconfiguração	(ALI *et al.*, 2017)

Fonte: Elaborado pela autora.

Alguns artigos citam autocura, autoproteção e autoconfiguração como caracteristicas da autoadaptação. Para este trabalho, autoadaptação será identificada como uma característica específica que os SASs possuem e autocura, autoproteção e autoconfiguração serão consideradas subcaracterísticas de autoadaptação.

Alguns trabalhos retornaram informações de extrema importância, que impactaram diretamente na primeira decomposição do SIG. O trabalho de Souza *et al.* (2011) mostrou que as subcaracterísticas de autocura e autoproteção estão intrinsecamente relacionadas à confiabilidade. O trabalho de Peruma e Krutz (2018) mostra que a característica de segurança também é muito importante para SAS, visto que ela pode afetar diretamente outras características do sistema e principalmente a subcaracterística de autoproteção. Sabendo disso, também há necessiade de inserir outra característica como pertinente para esta decomposição, a característica de segurança, pois ela impacta positivamente na característica de confiabilidade.

Feita a análise dos dados sobre confiabilidade e as caracteríssticas e subcaracteristicas específicas dos SASs, a próxima etapa foi realizar a extração das subcaracterísticas, conceitos e outras informações pertinentes da ISO 25010 (ISO, 2011). Este trabalho utiliza a definição de confiabildiade contida na ISO, onde confiabilidade é o grau em que um sistema, produto ou componente executa funções especificadas sob condições especificadas por um período de tempo especificado (ISO, 2011). Ela também define maturidade, tolerância a falhas, recuperabilidade e disponibilidade como suas sub características. A seguir a Tabela 2 mostra o conceito utilizado para confiabilidade neste trabalho.

Tabela 2 – Definição da característica de confiabilidde

Característica	Confiabilidade
Definição	O grau em que um sistema, produto ou componente executa funções especificadas sobcondições especificadas por um período de tempo especificado.

Fonte: Elaborado pela autora.

A Tabela 3 mostra todas as subcaracterísticas de autocura abordadas neste trabalho, bem como suas respectivas definições. Autocura foi uma das 3 subcaracterísticas dos sistemas autoadaptativos extraídas durante a análise dos artigos. Disponibildiade, maturidade e tolerância a falhas são subcaracterísticas de confiabilidade, que naturalmente apareceriam na decomposição do SIG por estarem contidas neste RNF, mas além disso, elas aparecem decompostas na característica de autocura pois os dados extraídos dos artigos mostram que estas subcaracterísticas impactam diretamente nela.

Tabela 3 – Definição de autocura e suas subcaracterísticas

Subcaracterística	Autocura
Definição	A capacidade de descobrir, diagnosticar e reagir às interrupções. Também pode antecipar possíveis problemas e, portanto, tomar as medidas adequadas para evitaruma falha.
Subaracterística	Disponibilidade
Definição	Grau em que um sistema, produto ou componente está operacional e acessível quando necessário para uso.
Subaracterística	Maturidade
Definição	Grau em que um sistema, produto ou componente atende às necessidades de confiabilidade sob operação normal.
Subaracterística	Tolerância a falhas
Definição	Grau em que um sistema, produto ou componente opera como pretendido, apesar da presença de hardware ou falhas de software.

Fonte: Elaborado pela autora.

Tabela 4 – Definição de autoconfiguração e suas subcaracterísticas

Subcaracterística	Autoconfiguração
Definição	A capacidade de se reconfigurar automaticamente e dinamicamenteem resposta às mudanças instalando, atualizando, integrando e compondo criação oudecomposição de entidades de software.
Subaracterística	Recuperabilidade
Definição	Grau em que, em caso de interrupção ou falha, um produto ou sistema pode recuperar os dados diretamente afetados e restabelecer o estado desejado do sistema.

Fonte: Elaborado pela autora.

A Tabela 4 mostra todas as subcaracterísticas de autoconfiguração abordadas neste trabalho, bem como suas respectivas definições. Recuperabilidade é uma subcaracterística de

confiabilidade, e naturalmente apareceria na decomposição do SIG, mas além disso, os dados extraídos dos artigos mostram que esta subcaracterística pode causar impacto na autoconfiguração, portanto foi decidido adicionar esta subcaracterística nesta decomposição.

A autoproteção bem como as subcaracterísticas já citadas, também está contida neste trabalho pois foi uma das 3 subcaracterísticas dos sistemas autoadaptativos extraídas durante a análise dos artigos. A Tabela 5 apresenta o conceito desta subcaracterística e de suas decomposições. Foi decidido adicionar também a característica de segurança à decomposição, pois ela está diretamente ligada a autoproteção. Com a inclusão da característica de Segurança, duas de suas subcaracterísticas, integridade e confidencialidade, também foram inclusas.

Tabela 5 – Definição de autoproteção e suas subcaracterísticas

Subcaracterística	Autoproteção
Definição	A capacidade de detectar violações de segurança e se recuperar de seus efeitos. Tem dois aspectos, defender o sistema contra ataques maliciosos e antecipar problemas tomando ações.
Característica	Segurança
Definição	Grau em que um produto ou sistema protege informações e dados para que pessoas/produtos/sistemas têm o grau de acesso a dados apropriado para seus tipos e níveis de autorização.
Subaracterística	Integridade
Definição	Grau em que um sistema, produto ou componente impede o acesso não autorizado, ou modificação de programas de computador ou dados.
Subaracterística	Confidencialidade
Definição	Grau em que um sistema, produto ou componente impede o acesso não autorizado, ou modificação de programas de computador ou dados.

Fonte: Elaborado pela autora.

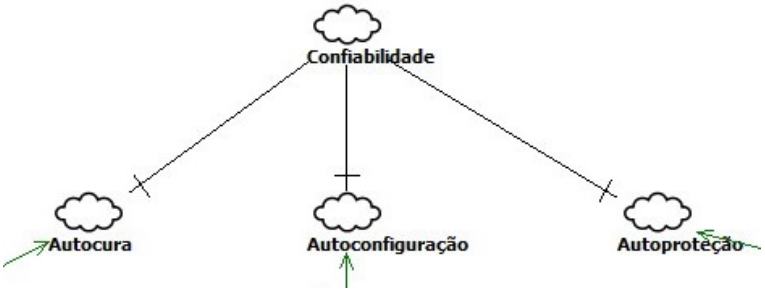
6.2 Decomposição do SIG

De acordo com o levantamento realizado na seção anterior, 3 características próprias de SAS são impactadas em maior nível pelo RNF de confiabilidade, a primeira decomposição do *softgoal* foi realizada. Confiabilidade foi decomposta em autocura, autoconfiguração e em autoproteção. A Figura 12 apresenta a decomposição realizada.

A partir disso, cada um desses *softgoals* foi decomposto individualmente. A decisão de quebrar confiabilidade nas 3 características dos SAS já na primeira decomposição do SIG, mantendo ambas na segunda linha de *softgoals* foi pensada principalmente pelo ponto de vista da organização. A primeira versão do SIG que foi construída, possuía essas características dispostas em linhas distintas. Porém, ao realizar as ligações entre os *softgoals*, foi visto que

ficaria desorganizado, e em alguns casos, confuso.

Figura 12 – Primeira decomposição do SIG



Fonte: Elaborado pela autora

Para realizar algumas decomposições do SIG foi necessário analisar os conceitos descritos nas tabela apresentadas na subseção anterior, visualizando quais subaracterísticas de confiabilidade se encaixariam em cada uma das 3 características dos SAS dispostas no SIG. Porém, essa análise foi apenas de cunho confirmativo em quase todos os *softgoals*, visto que muitos trabalhos retornaram informações sobre o relacionamento das características dos SAS com as subcaracterísticas de confiabilidade.

A partir disto, foi iniciada a segunda decomposição do SIG, que envolveu decompor os *softgoals* contidos na característica de autocura. O trabalho de Neti e Muller (2007) mostra que a característica disponibilidade é uma das características que mais está interligada e afeta positivamente a característica de autocura, portanto, ela foi decomposta em disponibilidade. A ISO 25010 (ISO, 2011) aponta tolerância a falhas e maturidade como características que impactam também na disponibilidade do sistema. Após isso, foi realizada uma anaĺise da definição de ambas as características citadas na ISO e partir disso disponibilidade foi decomposta em maturidade e maturidade foi decomposta em tolerância a falhas. A Figura 13 mostra a decomposição da característica de autocura.

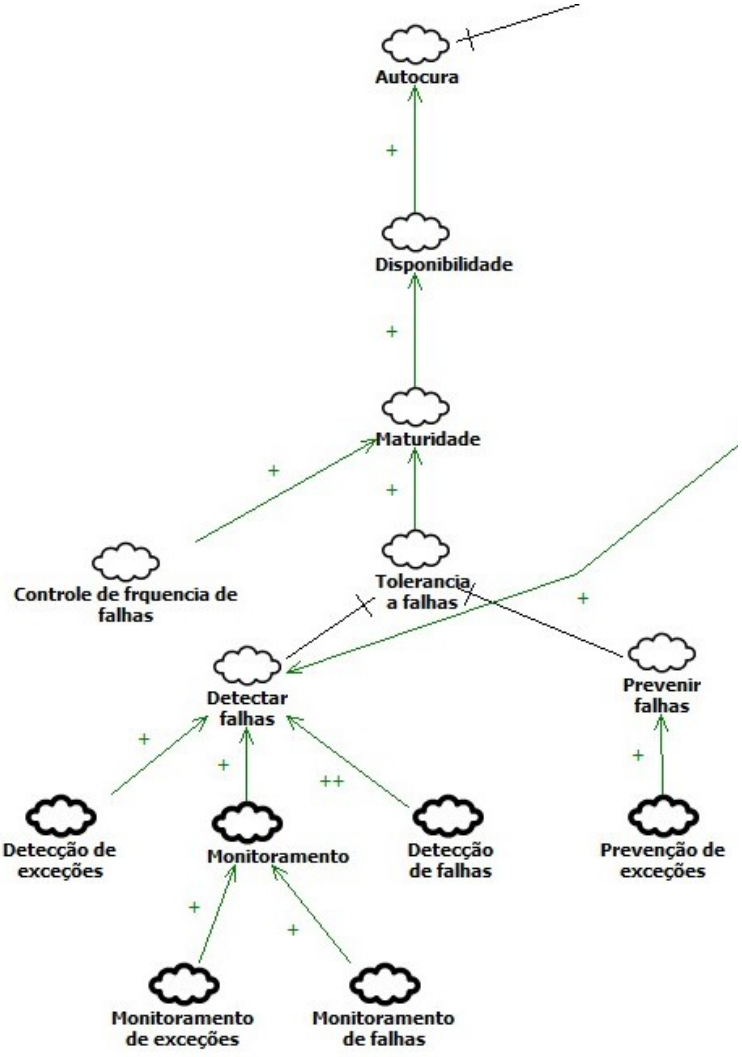
A terceira decomposição realizada foi a da característica de autoconfiguração. Nos artigos retornados no mapeamento não foram encontradas menções a características que estivessem relacionadas a autoconfiguração. Por conta disto, a decomposição deste *softgoal* em recuperabilidade, se deu ao analisar suas respectivas definições. Para evitar que esta decomposição seja subjetiva, na seção de avaliação do SIG serão apresentados os comentários do avaliador acerca da decomposição realizada. A Figura 14 mostra a decomposição desta característica.

Por fim, foi realizada a decomposição da característica de autoproteção. O trabalho de Peruma e Krutz (2018), cita que segurança está ligada à característica de autoproteção,

portanto, autoprotação foi decomposta em segurança. Peruma e Krutz (2018) também citam em seu trabalho que integridade e confidencialidade são duas características que impactam diretamenrte na segurança do sistema. Dessa forma, segurança foi decomposta em integridade e confidencialidade. Ao visualizar o *softgoal* de confidencialidade, existem mais duas características que impactam nela, autenticação e controle de acesso. O trabalho de Chung *et al.* (2000) contém um SIG de segurança onde autenticação e controle de acesso são decomposições de confidencialidade, por isso a escolha de realizar esta decomposição também para SASs, que necessitam de ambas as características. A Figura 15 mostra a decomposição completa desta característica.

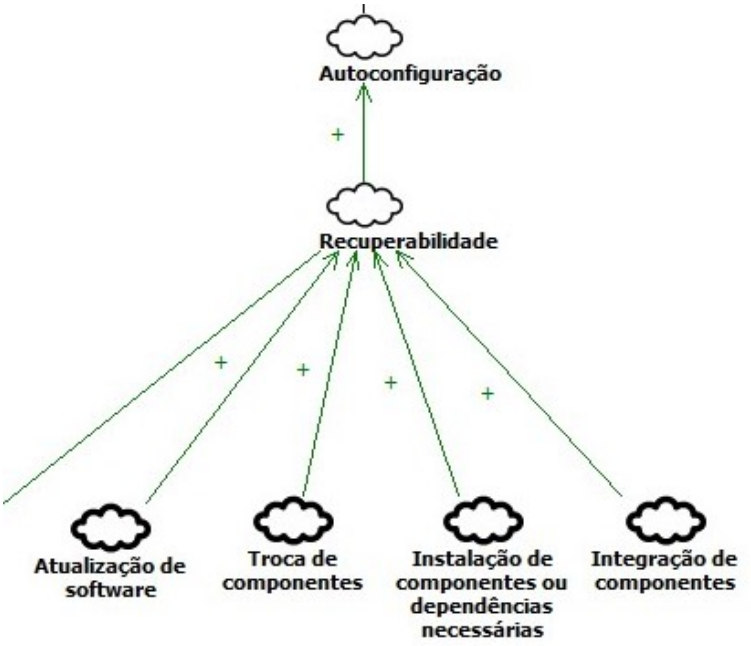
Com o fim da decomposição de cada uma das 3 características próprias de SAS, o SIG ficou completo. Após a finaização deste processo é possível visualizar um nível mais específico de como interações podem ocorrer entre as subcaraterísticas e as operacionalizações, dessa forma é possível apoiar o desenvolvimento destes sistemas de forma mais concreta. A Figura 16 mostra o resultado final obtido.

Figura 13 – Decomposição do *softgoal* autocura



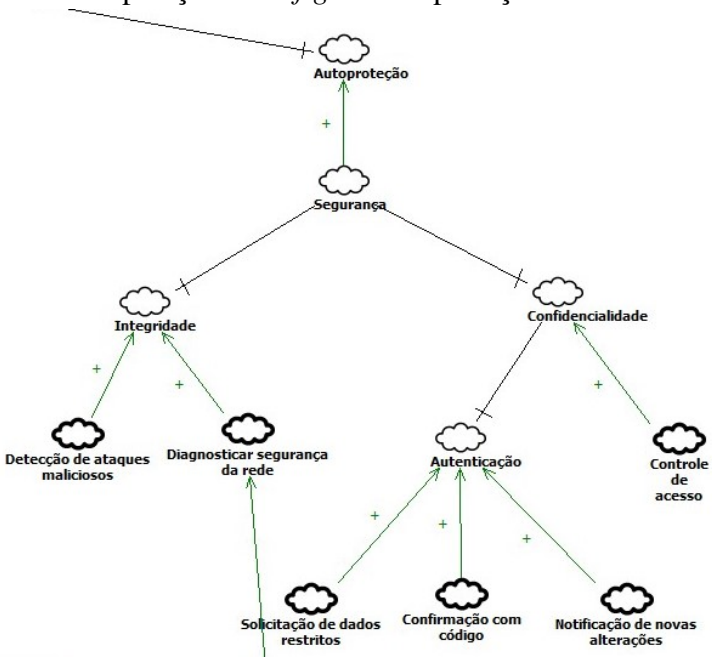
Fonte: Elaborado pela autora

Figura 14 – Decomposição do *softgoal* autoconfiguração



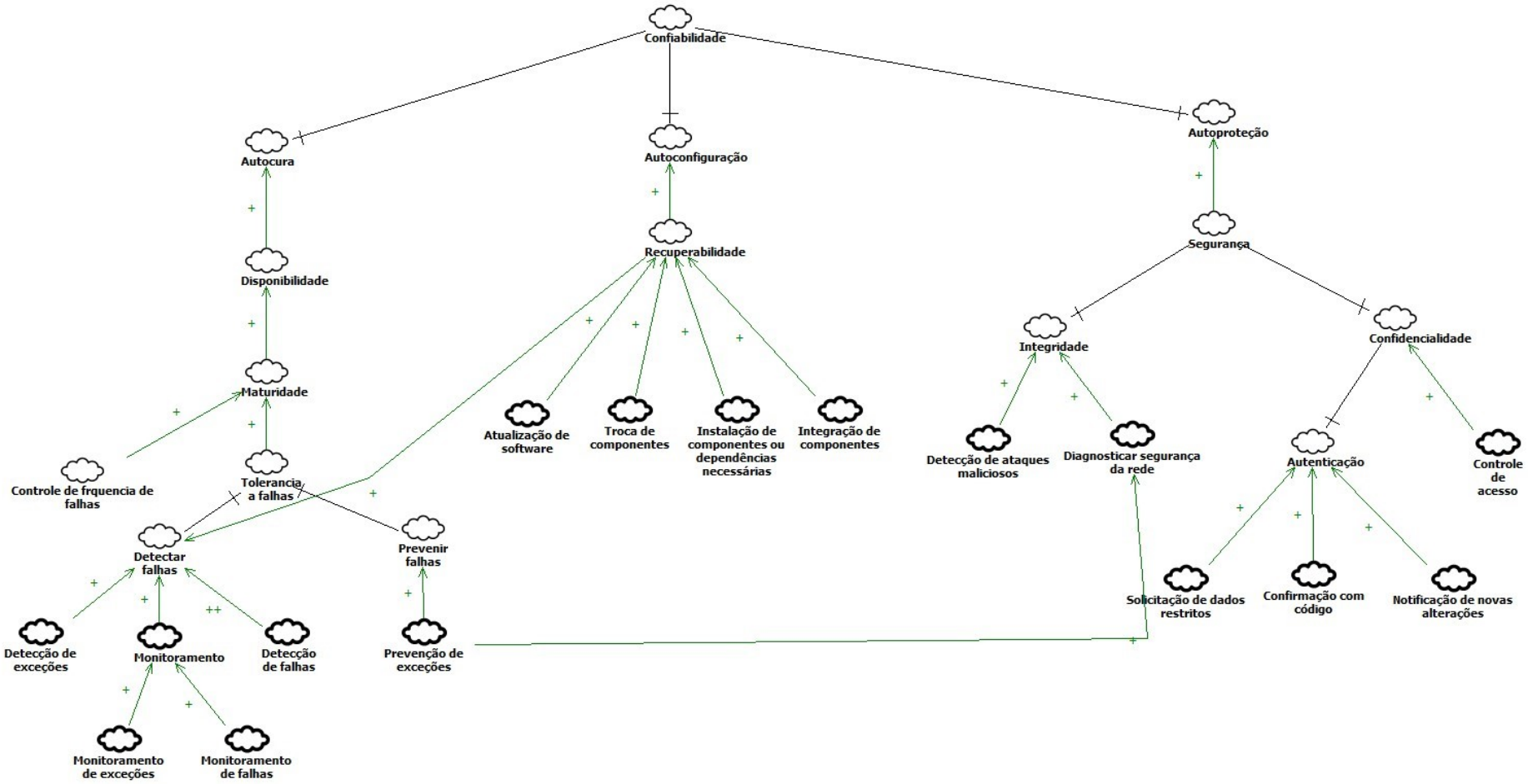
Fonte: Elaborado pela autora

Figura 15 – Decomposição do *softgoal* autoproteção



Fonte: Elaborado pela autora

Figura 16 – SIG de confiabilidade para SAS



Fonte: Elaborado pela autora

6.3 Planejamento da avaliação do SIG

Nesta etapa o processo de planejamento da avaliação do SIG foi iniciado. A primeira etapa foi realizar a escolha da metodologia de avaliação que seria utilizada neste trabalho, tendo em vista que a avaliação seria realizada com um especialista do domínio de SAS. Dessa forma, foi decidido utilizar um *checklist* para capturar os dados. *Checklists* são listas de verificação, uma ferramenta de controle baseada em uma lista com diversas condutas, nomes, tarefas ou atividades, que devem ser lembradas ou seguidas para que determinado resultado seja alcançado de forma sistemática.

A escolha de utilizar um *checklist* partiu da necessidade de obter e documentar de forma clara e objetiva os resultados obtidos neste processo. Para que os resultados alcançados estivessem de acordo com o esperado para a avaliação do SIG, antes da aplicação do *checklist* foi decidido realizar uma reunião com o especialista para apresentar o escopo do trabalho. O quadro 7 apresenta as 10 perguntas contidas no *checklist* apresentado ao avaliador.

Quadro 7 – Questões do *checklist*

P01	Os softgoals estão descompostos com clareza e consistência?
P02	Existem *softgoals* conflitantes?
P03	Existem *softgoals* implícitos?
P04	A distinção entre os *softgoals* e suas operacionalizações estão claras?
P05	Existe algum *softgoal* que possui níveis desnecessários de decomposição?
P06	O SIG possui todos os *softgoals* necessários?
P07	As características de SAS foram abordadas da maneira correta?
P08	As características de SAS abordadas no SIG deveriam possuir mais operacionalizações a serem abordadas?
P09	Existe alguma outra característica de SAS que deveria ter sido abordada no SIG?
P10	As operacionalizações do SIG devem possuir uma decomposição mais profunda?

Fonte: Elaborado pela autora.

Foram desenvolvidas 10 perguntas com o propósito de avaliar a coerência do SIG desenvolvido. Cada pergunta foca em uma possível problematica advinda da decomposição realizada. Existem 2 campos, sim e não, onde o avaliador identifica se a pergunta possui resposta positiva ou negativa. Além disso, todas as perguntas possuem um campo de sugestão, onde o avaliador poderá incluir as sugestões de melhoria para os problemas encontrados.

As perguntas P01, P02, P03, P04, P05, P06 e P10 buscam identificar problemas específicos na decomposição do SIG. A pergunta P01 tem como objetivo descobrir se o SIG,

de uma forma geral, está bem organizado e claro visualmente para o avaliador. A pergunta P02 tem como objetivo descobrir se há relação de conflito entre os *softgoals*, visto que não foram identificados conflitos na primeira decomposição. A pergunta P03 busca descobrir se há *softgoals* implícitos que não foram captados durante o processo de decomposição. A pergunta P04 busca descobrir se os *softgoals* e suas respectivas operacionalizações estão bem decompostas e se esta decomposição foi realizada de forma clara e coerente. A pergunta P05 busca descobrir se foram realizadas decomposições desnecessárias ou sem fundamentos. E a P06 tem como objetivo descobrir se o SIG possui os *softgoals* necessários, com base no *softgoal* inicial (confiabilidade) ou se houve falha na idetificação dos *softgoals* a serem decompostos.

As perguntas P07, P08 e P09 tem como objetivo identificar problemas ou deficiências no SIG do ponto de vista dos SASs. A pergunta P07 tem como objetivo descobrir se as características de SAS abordadas no SIG foram decompostas da maneira correta ou se existem subcaracterísticas importantes em falta. A pergunta P08 tem o objetivo de descobrir se as existem operacinalizações específicas do domínios dos SAS que não estão contidas no SIG. E por fim, a pergunta P09 tem como objetivo descobrir se existem características dos SAS que deveriam estar relacionadas ao *softgoal* de confiabilidade, mas não foram apresentadas na decomposição.

O Quadro 8 identifica as perguntas por área abordada.

Quadro 8 – Classificação das perguntas do *checklist*

Perguntas sobre validade do SIG do ponto de vista da autodaptação	P01, P02, P03, P04, P05, P06 e P10
Perguntas sobre a validade do ponto de vista da decomposição	P07, P08 e P09

Fonte: Elaborado pela autora.

Com a finalização das questões do *checklist*, o próximo passo foi realizar uma entrevista com o avaliador para apresentar de forma resumida o trabalho desenvolvido. A próxima Seção descreve a execução da reunião e os passos seguintes.

6.4 Execução da avaliação

Para a avaliação do SIG construído, uma reunião com um especialista no domínio dos SAS foi realizada. Nesta reunião, inicialmente, foi apresentado o SIG e conceitos importantes para sua compreensão. Houve uma rapida explicação sobre o conceito de conflitos entre requisitos não funcionais e sobre o escopo do projeto, identificando o motivo da realização deste trabalho e da criação do SIG. Além disso, houve uma breve explicação sobre como o NFR *Framework*

funciona e sobre conceito de SIG.

Após isso, o avaliador redigiu algumas perguntas sobre o processo de criação do SIG, a fim de descobrir mais sobre o processo de decomposição do RFN e como ele foi executado. Feito isso, o avaliador discorreu sobre o SIG de um ponto de vista geral, dando dicas de acordo com a primeira visão obtida, gerando resultados iniciais de melhorias para o processo. Ao final, o avaliador respondeu pela primeira vez o *checklist* com as 10 perguntas, ainda em reunião, para que fosse possível retirar as demais dúvidas que surgissem. Dessa forma, um primeiro *feedback* foi recebido, gerando dados iniciais para análise. Ainda assim, o SIG e o *checklist* foram enviados a ele posteriormente para uma avaliação mais detalhada, assim seria possível analisar problemas não identificados durante a reunião.

O avaliador retornou o *checklist* com algumas sugestões de melhorias e esses resultados foram utilizados para melhorar o SIG desenvolvido. A Seção 6.5 mostrará os resultados obtidos e as sugestões de melhorias obtidas com esta avaliação, bem como a remodelação do SIG de acordo com os dados retornados.

6.5 Resultados da avaliação do SIG

Esta seção apresenta os dados obtidos após a avaliação do SIG com o especialista no domínio de SAS. A reunião para apresentação do SIG durou cerca de 50 minutos. O checklist de 10 peguntas foi enviado ao especialista no mesmo dia, ao final da reunião. Das 10 perguntas contidas no questionário, 4 obtiveram resultado negativo, ou seja, apontavam aspectos de melhorias para o SIG e 6 perguntas foram identificadas como positivas, o que demonstra que mais da metade dos aspectos foi atingido com sucesso.

O Quadro 8 apresenta as perguntas divididas por focos de avaliação. A ideia foi avaliar a coerência do SIG do ponto de vista da sua decomposição e dos SASs. As sugestões de melhorias serão citadas a seguir:

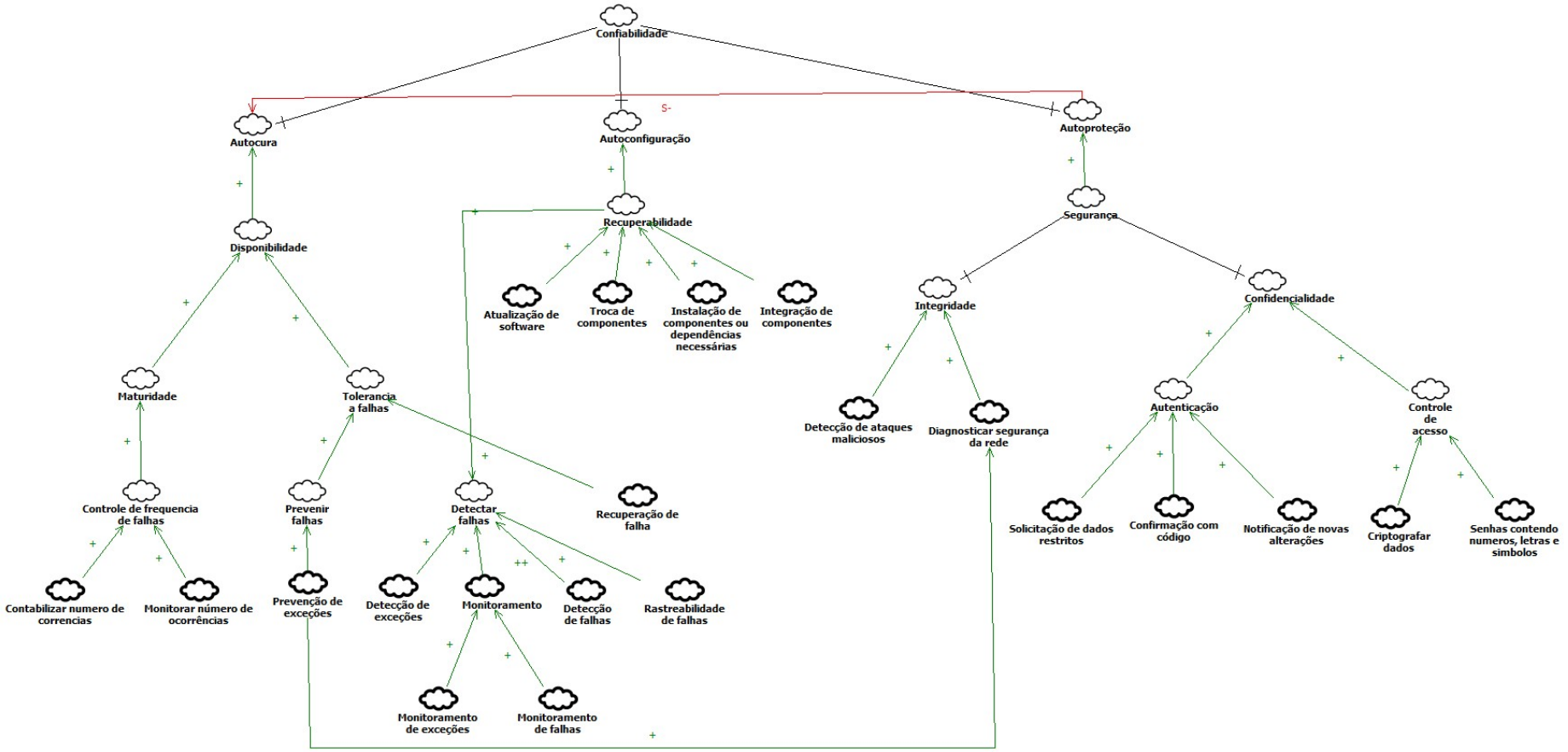
- A P02 busca identificar se existem *softgoals* conflitantes no SIG. Na primeira versão, não foram identificados conflitos entre os *softgoals* presentes. Porém, o avaliador retornou no *checklist* que haveriam conflitos entre as subcaracterísticas de autocura e autoproteção. Ele utilizou o seguinte exemplo para demonstrar: caso o sistema precise adotar medidas que interfiram na sua protecão para se recuperar de algum problema, a subcaracterística de autocura iria ferir a subcaracterística de autoproteção. Isso causaria um conflito.
- A P05 busca identificar se existem níveis de decomposição desnecessários no SIG. O

avaliador retornou no *checklist* que sim. Sua sugestão foi adicionar a subcaracterística de maturidade no mesmo nível das subcaracterísticas controle de frequência de falhas e tolerância a falhas.

- A P06 busca identificar se existem problemas na identificação dos *softgoals*, se existem *softgoals* que deveriam estar presentes e não estão. O avaliador retornou que, na visão dele, faltou o *softgoal* recuperação de falhas como subcaracterística de tolerância a falha. Ele informou que mesmo havendo uma ligação de ajuda entre recuperabilidade e tolerância a falhas, ainda seria necessário incluir recuperacão de falhas como uma operacionalização de tolerância a falhas. Isso decorre pois a subcaracterística de autoconfiguração não será ativada sempre que o sistema precisar se recuperar de alguma falha, apenas em alguns cenários. Além disso, ele citou que poderia haver o *softgoal* criptografia como solução de confidencialidade.
- A P08 busca identificar se as características de SAS abordadas no SIG deveriam possuir mais operacionalizações a serem abordadas. O avaliador respondeu que poderia haver uma operacionalização para o *softgoal* detectar falhas, chamada rastreabilidade.

As demais perguntas obtiveram respostas positivas, portanto, não houveram sugestões da parte do avaliador. O SIG sofreu alterações após as respostas obtidas e o resultado final é apresentado na Figura 17.

Figura 17 – SIG final



Fonte: Elaborado pela autora

7 AMEAÇAS À VALIDADE

Este trabalho possui possíveis ameaças à validade. A primeira delas está relacionada ao fato de não terem sidos considerados todos os estudos existentes sobre sistemas autoadaptativos, o que pode levar a resultados incompletos. Dessa forma, foi decidido investigar os trabalhos que abordavam sistemas autoadaptativos e RNFs, para que fosse possível realizar uma análise abrangente, com foco nestes dois conceitos. Além disso, o processo do mapeamento foi conduzido apenas por um pesquisador, apenas uma pessoa realizou a análise dos dados e extração dos conteúdos pertinentes a essa pesquisa, o que pode se considerar um risco, pois a extração foi realizada com base em análises realizadas por um número muito limitado.

Outra preocupação é sobre o SIG, está relacionada com a descrição de cada softgoal (requisitos de confiabilidade) sendo fundamentado em dados. Então, as descrições deste trabalho são baseadas apenas nos dados que foram extraídos do mapeamento e da ISO 25010 (ISO, 2011). Além disso, a avaliação do SIG foi realizada apenas com um especialista do domínio dos SAS, o que gerou dados limitados sobre a avaliação do gráfico e dos relacionamentos existentes entre os softgoals.

Por fim, a interpretação dos resultados do questionário foi realizada por apenas um pesquisador, o que pode levar a interpretações errôneas do que foi obtido. Para tentar mitigar essa ameaça, um campo de sujestão foi adicionado ao checklist, neste campo o especialista pôde explicar detalhadamente o motivo de sua resposta e caso fossem detectados problemas, ele poderia também exemplificar para ficar mais didático de entender.

8 CONCLUSÕES E TRABALHOS FUTUROS

Sistemas Autoadaptativos (SAS) são sistemas que conseguem se modificar e realizar operações de melhoria ou incremento em tempo de execução, identificando falhas de operação e dispararando estratégias de configuração para continuar em operação (WU *et al.*, 2019). Em decorrência disto, esses sistemas geralmente precisam atender a uma série de aspectos voltados para sua qualidade (e.g., disponibilidade, segurança, eficiência e desempenho) (RAIBULET *et al.*, 2017).

Em decorrência das características adaptativas que estes sistemas possuem, eles são complexos e possuem ligação direta com RNFs ou características de qualidade. Isso decorre, pelo fato que eles precisarem manter sua qualidade para seu funcionamento correto,impactando também no número de falhas que o sistema pode possuir (**??**).

Os RNFs estão diretamente ligados a questão da qualidade de software, mas possuem problemas que dificultam a sua implementação. Eles tendem a entrar em conflitos e se contradizer, o que é reconhecido como um dos principais problemas encontrados ao lidar com RNFs ZhangWang2019. Além dsisso, a interpretação acerca destes requisitos podem variar dependendo do contexto do sistema (CHAZETTE; SCHNEIDER, 2020).

Uma maneira de lidar com a problemática dos conflitos é utilizando o NFR *framework* e o conceito de SIG (*Softgoals Interdependence Graphics*). Desenvolvido pelo pesquisador Chung *et al.* (2000), seu objetivo é ajudar os desenvolvedores na implementação de soluções. Geralmente neste processo é utilizado o conceito de SIG, que descreve as dependências entre os *softgoals* e como eles são decompostos. *Softgoals* são unidades básicas que representam RNFs ou metas genéricas e flexíveis (CHUNG *et al.*, 2000).

Visando a preservação da qualidade dos SASs, neste trabalho foi realizado primeiramente um mapeamento sistemático com o objetivo de identificar o estado da literatura acerca dos RNFs e o processo de requisitos para SAS. Neste mapeamento, foram consideradas as características e subcaracterísticas apresentadas pela ISO (2011). Dentre os resultados do mapeamento, a confiabilidade foi a característica mais citada entre os 31 estudos selecionados. Quanto às subcaracterístias mais retornadas, disponibilidade foi a mais citada nos trabalhos. Além disso, foi observado que a confiabilidade influencia significante as propriedades e desemepnho dos SAS, além de impactar a qualidade do serviço.

Em decorrência dos dados obtidos, o foco deste trabalho foi o RNF de confiabilidade. Após a execução do mapeamento, foi iniciado o processo de criação do SIG, que utilizou trabalhos

do mapeamento sistemático para sua decomposição. Os artigos que retornaram informações acerca de confiabilidade foram analisados e a partir disto foi possível identificar características e subcaracterísticas próprias dos SAS que são influenciadas diretamente ou indiretamente por este RNF.

Ao fim na análise dos trabalhos, foi obtida uma nova característica de autoadaptação, e como subcaracterísticas foram obtidas autocura, autoproteção e autoconfiguração. A partir dessas informações, o SIG começou a ser construído. Ao todo, o processo de construção consistiu em 4 etapas: (i) na primeira foi realizada a decomposição do *softgoal* de confiabilidade em autocura, autoproteção e autoconfiguração, (ii) na segunda etapa, o *softgoal* de autocura foi decomposto até chegar nas suas operacionalizações, (iii) a terceira etapa consistiu no mesmo processo de decomposição, porém para a característica de autoconfiguração, e (iv) a quarta etapa consistiu no mesmo processo para a característica de autoproteção. Ao fim dessas etapas, uma primeira versão do SIG foi obtida.

Para realizar a avaliação do SIG, um especialista do domínio de SASs preencheu um *checklist* com 10 perguntas, com o objetivo de avaliar a qualidade do SIG proposto do ponto de vista da decomposição realizada e do domínio dos SASs. A avaliação do SIG mostrou que melhorias seriam necessárias na decomposição realizada na primeira versão. Além disso, uma relação de conflito foi identificada pelo avaliador entre as subcaracterísticas de autocura e autoconfiguração. A partir disto, a primeira versão do SIG foi adaptada, gerando a versão final apresentada neste trabalho. O SIG desenvolvido ao longo deste estudo pode ser utilizado como base de apoio aos desenvolvedores ou analistas de requisitos que estejam interessados em entender as relações entre a característica confiabilidade, e as características e subcaracterísticas específicas dos SASs que confiabilidade influencia.

Como trabalhos futuros, pode-se vislumbrar: (i) catalogar as informações obtidas neste trabalho, através de uma plataforma *online* ou um arquivo pdf; (ii) avaliar um conjunto de RNFs e como eles interagem entre si, investigando outros requisitos impactantes para SAS; e, (iii) mapear conflitos entre estes RNFs e construir SIGs ou catálogos que podem ajudar no processo de desenvolvimento dos SASs.

REFERÊNCIAS

ABDUL, H.; JAMIL, A.; IMRAN, U. Conflicts identification among non-functional requirements using matrix maps. **World Academy of Science Engineering and Technology**, Citeseer, v. 44, p. 1004–1009, 2010.

ADJOYAN, S.; SERIAI, A.-D. Reconfigurable service-based architecture based on variability description. In: **Proceedings of the Symposium on Applied Computing**. New York, NY, USA: Association for Computing Machinery, 2017. (SAC '17), p. 1154–1161. ISBN 9781450344869. Disponível em: https://doi.org/10.1145/3019612.3019767. Acesso em: 02 ago. 2021.

AHMAD, M.; BELLOIR, N.; BRUEL, J.-M. Modeling and verification of functional and non-functional requirements of ambient self-adaptive systems. **Journal of Systems and Software**, v. 107, p. 50–70, 2015. ISSN 0164-1212.

ALI, N.; MARTÍNEZ-MARTÍNEZ, A.; AYUSO-PéREZ, L.; ESPINOZA, A. Self-adaptive quality requirement elicitation process for legacy systems: A case study in healthcare. In: **Proceedings of the Symposium on Applied Computing**. New York, NY, USA: Association for Computing Machinery, 2017. (SAC '17), p. 1102–1107. ISBN 9781450344869.

ANGELOPOULOS, K.; PAPADOPOULOS, A. V.; SOUZA, V. E. S.; MYLOPOULOS, J. Engineering self-adaptive software systems: From requirements to model predictive control. **ACM Trans. Auton. Adapt. Syst.**, Association for Computing Machinery, New York, NY, USA, v. 13, n. 1, apr 2018. ISSN 1556-4665.

BOWERS, K.; FREDERICKS, E.; H.HARIRI, R.; CHENG, B. Providentia: Using search-based heuristics to optimize satisficement and competing concerns between functional and non-functional objectives in self-adaptive systems. **Journal of Systems and Software**, v. 162, p. 110497, 12 2019.

CARVALHO, R. M.; ANDRADE, R. M.; OLIVEIRA, K. M.; KOLSKI, C. Catalog of invisibility requirements for ubicomp and iot applications. In: IEEE. **2018 IEEE 26th International Requirements Engineering Conference (RE)**. [*S. l.*], 2018. p. 88–99.

CHAZETTE, L.; SCHNEIDER, K. Explainability as a non-functional requirement: challenges and recommendations. **Requirements Engineering**, v. 25, 12 2020.

CHEMUTURI, M. **Requirements engineering and management for software development projects**. [*S. l.*]: Springer Science & Business Media, 2012.

CHENG, B. H. C.; SAWYER, P.; BENCOMO, N.; WHITTLE, J. A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty. In: SCHÜRR, A.; SELIC, B. (Ed.). **Model Driven Engineering Languages and Systems**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 468–483. ISBN 978-3-642-04425-0.

CHUNG, L.; LEITE, J. C. S. do P. On non-functional requirements in software engineering. In: **Conceptual Modeling: Foundations and Applications**. [*S. l.*: *s. n.*], 2009.

CHUNG, L.; NIXON, B. A.; YU, E.; MYLOPOULOS, J. The nfr framework in action. In: **Non-Functional Requirements in software engineering**. [*S. l.*]: Springer, 2000. p. 15–45.

CHUNG, L.; NIXON, B. A.; YU, E.; MYLOPOULOS, J. **Non-functional requirements in software engineering**. [*S. l.*]: Springer Science & Business Media, 2012. v. 5.

CYSNEIROS, L. M.; LEITE, J. C. S. do P. Utilizando requisitos não funcionais para análise de modelos orientados a dados. In: **WER**. [*S. l.*: *s. n.*], 1998. p. 149–158.

D'ANGELO, M.; CAPORUSCIO, M.; GRASSI, V.; MIRANDOLA, R. Decentralized learning for self-adaptive qos-aware service assembly. **Future Generation Computer Systems**, v. 108, p. 210–227, 2020. ISSN 0167-739X.

EDWARDS, R.; BENCOMO, N. Desire: Further understanding nuances of degrees of satisfaction of non-functional requirements trade-off. In: **2018 IEEE/ACM 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)**. [*S. l.*: *s. n.*], 2018. p. 12–18.

EGYED, A.; GRUNBACHER, P. Identifying requirements conflicts and cooperation: How quality attributes and automated traceability can help. **IEEE software**, IEEE, v. 21, n. 6, p. 50–58, 2004.

FILIERI, A.; GHEZZI, C.; TAMBURRELLI, G. A formal approach to adaptive software: continuous assurance of non-functional requirements. **Formal Aspects of Computing**, v. 24, p. 163–186, 2011.

GHEZZI, C.; PINTO, L. S.; SPOLETINI, P.; TAMBURRELLI, G. Managing non-functional uncertainty via model-driven adaptivity. In: **Proceedings of the 2013 International Conference on Software Engineering**. [*S. l.*]: IEEE Press, 2013. (ICSE '13), p. 33–42. ISBN 9781467330763.

GHEZZI, C.; SHARIFLOO, A. M. Dealing with non-functional requirements for adaptive systems via dynamic software product-lines. In: **Software Engineering for Self-Adaptive Systems II**. [*S. l.*]: Springer, 2013. p. 191–213.

GOLDSBY, H. J.; SAWYER, P.; BENCOMO, N.; CHENG, B. H.; HUGHES, D. Goal-based modeling of dynamically adaptive system requirements. In: **15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ecbs 2008)**. [*S. l.*: *s. n.*], 2008. p. 36–45.

GONZALEZ-SANCHEZ, J. Toward a software product line for affective-driven self-adaptive systems. In: **Proceedings of the 2013 International Conference on Software Engineering**. [*S. l.*]: IEEE Press, 2013. (ICSE '13), p. 1381–1384. ISBN 9781467330763.

HAIGH, M. Software quality, non-functional software requirements and it-business alignment. **Software Quality Journal**, v. 18, p. 361–385, 2010.

HAMMANI, F. Z. Survey of non-functional requirements modeling and verification of software product lines. In: IEEE. **2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS)**. [*S. l.*], 2014. p. 1–6.

HEZAVEHI, S. M.; DURELLI, V.; WEYNS, D.; AVGERIOU, P. A systematic literature review onmethods that handle multiple quality attributes in architecture-based self-adaptive systems. **Information and Software Technology**, v. 90, 04 2017.

HORCAS, J.-M.; PINTO, M.; FUENTES, L. An automatic process for weaving functional quality attributes using a software product line approach. **Journal of Systems and Software**, v. 112, 12 2015.

IBM-AC. **IBM**. 2001. Disponível em: https://www.research.ibm.com/autonomic/overview/elements.html. Acesso em: 14 jul. 2021.

ISO. Iec 25010: 2011 systems and software engineering–systems and software quality requirements and evaluation (square)–system and software quality models. **International Organization for Standardization**, v. 34, p. 2910, 2011.

ISO. Iec 25023: 2016 systems and software engineering — systems and software quality requirements and evaluation (square) — measurement of system and software product quality. **International Organization for Standardization**, v. 1, p. 45, 2016.

ISO/IEC. **Software engineering–Product quality–Part 1**. [*S. l.*], 2001.

ISO/IEC. **ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models**. [*S. l.*], 2011.

ISO/IEC. **ISO/IEC 25000 - Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE**. [*S. l.*], 2014.

KEELE, S. *et al.* **Guidelines for performing systematic literature reviews in software engineering**. [*S. l.*], 2007.

KEPHART, J. O.; CHESS, D. M. The vision of autonomic computing. **Computer**, IEEE, v. 36, n. 1, p. 41–50, 2003.

KOZIOLEK, A.; ARDAGNA, D.; MIRANDOLA, R. Hybrid multi-attribute qos optimization in component based software systems. **Journal of Systems and Software**, v. 86, 10 2013.

LADDAGA, R.; ROBERTSON, P. Self adaptive software: A position paper. In: CITESEER. **SELF-STAR: International Workshop on Self-* Properties in Complex Information Systems**. [*S. l.*], 2004. v. 31, p. 19.

LAMSWEERDE, A. V.; DARIMONT, R.; LETIER, E. Managing conflicts in goal-driven requirements engineering. **IEEE transactions on Software engineering**, IEEE, v. 24, n. 11, p. 908–926, 1998.

LUCKEY, M.; ENGELS, G. High-quality specification of self-adaptive software systems. In: **2013 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)**. [*S. l.*: *s. n.*], 2013. p. 143–152.

MAHDAVI-HEZAVEHI, S. Handling multiple quality attributes trade-off in architecture-based self-adaptive systems. In: **Proccedings of the 10th European Conference on Software Architecture Workshops**. New York, NY, USA: Association for Computing Machinery, 2016. (ECSAW '16). ISBN 9781450347815.

MAIRIZA, D.; ZOWGHI, D. Constructing a catalogue of conflicts among non-functional requirements. In: SPRINGER. **International conference on evaluation of novel approaches to software engineering**. [*S. l.*], 2010. p. 31–44.

MARTINEZ, G. G.; CARPIO, Á. F. D.; GÓMEZ, L. N. A model for detecting conflicts and dependencies in non-functional requirements using scenarios and use cases. In: IEEE. **2019 XLV Latin American Computing Conference (CLEI)**. [*S. l.*], 2019. p. 1–8.

MORANDINI, M.; PENSERINI, L.; PERINI, A.; MARCHETTO, A. Engineering requirements for adaptive systems. **Requirements Engineering**, Springer, v. 22, n. 1, p. 77–103, 2017.

NETI, S.; MULLER, H. A. Quality criteria and an analysis framework for self-healing systems. In: **International Workshop on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '07)**. [*S. l.*: *s. n.*], 2007. p. 6–6.

NITTO, E. D.; GHEZZI, C.; METZGER, A.; PAPAZOGLOU, M.; POHL, K. A journey to highly dynamic, self-adaptive service-based applications. **Automated Software Engineering**, Springer, v. 15, n. 3, p. 313–341, 2008.

NURBOJATMIKO; BUDIARDJO, E. K.; WIBOWO, W. C. Slr on identification amp; classification of non-functional requirements attributes, and its representation in functional requirements. In: **Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence**. New York, NY, USA: Association for Computing Machinery, 2018. (CSAI '18), p. 151–157. ISBN 9781450366069.

NUSEIBEH, B.; EASTERBROOK, S. Requirements engineering: a roadmap. In: **Proceedings of the Conference on the Future of Software Engineering**. [*S. l.*: *s. n.*], 2000. p. 35–46.

PENG, X.; CHEN, B.; YU, Y.; ZHAO, W. Self-tuning of software systems through dynamic quality tradeoff and value-based feedback control loop. **Journal of Systems and Software**, v. 85, n. 12, p. 2707–2719, 2012. ISSN 0164-1212. Self-Adaptive Systems.

PERUMA, A.; KRUTZ, D. Security: A critical quality attribute in self-adaptive systems. In: **2018 IEEE/ACM 13th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)**. [*S. l.*: *s. n.*], 2018. p. 188–189.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic mapping studies in software engineering. In: **Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering**. Swindon, GBR: BCS Learning amp; Development Ltd., 2008. (EASE'08), p. 68–77.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software - 8ª Edição**. [*S. l.*]: McGraw Hill Brasil, 2016. ISBN 9788580555349.

PéREZ, B.; CORREAL, D. A model driven approach to the analysis of quality scenarios within self-adaptable soa systems. **Electronic Notes in Theoretical Computer Science**, v. 281, p. 113–126, 12 2011.

RAGO, A.; VIDAL, S.; DIAZ-PACE, J. A.; FRANK, S.; HOORN, A. van. Distributed quality-attribute optimization of software architectures. In: **Proceedings of the 11th Brazilian Symposium on Software Components, Architectures, and Reuse**. New York, NY, USA: Association for Computing Machinery, 2017. (SBCARS '17). ISBN 9781450353250.

RAIBULET, C.; FONTANA, F. A.; CAPILLA, R.; CARRILLO, C. An overview on quality evaluation of self-adaptive systems. **Managing Trade-Offs in Adaptable Software Architectures**, Elsevier, p. 325–352, 2017.

ROBERTSON, P.; LADDAGA, R.; SHROBE, H. Introduction: the first international workshop on self-adaptive software. In: SPRINGER. **International Workshop on Self-Adaptive Software**. [*S. l.*], 2000. p. 1–10.

ROBINSON, W. N.; PAWLOWSKI, S. D.; VOLKOV, V. Requirements interaction management. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 35, n. 2, p. 132–190, 2003.

ROSA, L.; RODRIGUES, L.; LOPES, A. Self-management of distributed systems using high-level goal policies. In: ____. **Software Engineering for Self-Adaptive Systems II: International Seminar, Dagstuhl Castle, Germany, October 24-29, 2010 Revised Selected and Invited Papers**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 162–190. ISBN 978-3-642-35813-5.

SALEHIE, M.; TAHVILDARI, L. Self-adaptive software: Landscape and research challenges. **ACM transactions on autonomous and adaptive systems (TAAS)**, ACM New York, NY, USA, v. 4, n. 2, p. 1–42, 2009.

SEN, S.; ALESIO, S. D.; MARIJAN, D.; SARKAR, A. Evaluating reconfiguration impact in self-adaptive systems–an approach based on combinatorial interaction testing. In: IEEE. **2015 41st Euromicro Conference on Software Engineering and Advanced Applications**. [*S. l.*], 2015. p. 250–254.

SOUSA, A.; UCHôA, A.; FERNANDES, E.; BEZERRA, C. I. M.; MONTEIRO, J. M.; ANDRADE, R. M. C. Rem4dspl: A requirements engineering method for dynamic software product lines. In: **Proceedings of the XVIII Brazilian Symposium on Software Quality**. New York, NY, USA: Association for Computing Machinery, 2019. (SBQS'19), p. 129–138. ISBN 9781450372824.

SOUSA, A. O. de; BEZERRA, C. I. M.; ANDRADE, R. M. C.; FILHO, J. M. S. M. Quality evaluation of self-adaptive systems: Challenges and opportunities. In: . New York, NY, USA: Association for Computing Machinery, 2019. (SBES 2019), p. 213–218. ISBN 9781450376518.

SOUZA, V. E. S.; LAPOUCHNIAN, A.; MYLOPOULOS, J. System identification for adaptive software systems: A requirements engineering perspective. In: SPRINGER. **International Conference on Conceptual Modeling**. [*S. l.*], 2011. p. 346–361.

SUBRAMANIAN, N.; CHUNG, L. Software architecture adaptability: An nfr approach. In: **Proceedings of the 4th International Workshop on Principles of Software Evolution**. New York, NY, USA: Association for Computing Machinery, 2001. (IWPSE '01), p. 52–61. ISBN 1581135084.

VILLEGAS, N. M.; MüLLER, H. A.; TAMURA, G.; DUCHIEN, L.; CASALLAS, R. A framework for evaluating quality-driven self-adaptive software systems. In: **Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems**. New York, NY, USA: Association for Computing Machinery, 2011. (SEAMS '11), p. 80–89. ISBN 9781450305754.

WAHONO, R. S. Analyzing requirements engineering problems. In: **IECI Japan Workshop**. [*S. l.*: *s. n.*], 2003. v. 2003.

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

WU, T.; LI, Q.; WANG, L. A validation method of self-adaptive strategy based on pomdp. In: **2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)**. [*S. l.*: *s. n.*], 2019. p. 373–375.

WU, Y.; WU, Y.; PENG, X.; ZHAO, W. Implementing self-adaptive software architecture by reflective component model and dynamic aop: A case study. In: **2010 10th International Conference on Quality Software**. [*S. l.*: *s. n.*], 2010. p. 288–293.

WYATT, J.; GULY, H. Identifying the research question and planning the project. **Emergency medicine journal: EMJ**, BMJ Publishing Group, v. 19, n. 4, p. 318, 2002.

ZHANG, J.; CHENG, B. H. C. Model-based development of dynamically adaptive software. In: **Proceedings of the 28th International Conference on Software Engineering**. New York, NY, USA: Association for Computing Machinery, 2006. (ICSE '06), p. 371–380. ISBN 1595933751.

ZHANG, X.; WANG, X. Tradeoff analysis for conflicting software non-functional requirements. **IEEE Access**, v. 7, p. 156463–156475, 2019.

ZHAO, T.; ZHAO, H.; ZHANG, W. A preliminary study on requirements modeling methods for self-adaptive software systems. In: **Proceedings of the 5th Asia-Pacific Symposium on Internetware**. New York, NY, USA: Association for Computing Machinery, 2013. (Internetware '13). ISBN 9781450323697.

APÊNDICE A – CONJUNTO FINAL DOS ARTIGOS SELECIONADOS NO MAPEAMENTO

Tabela 6 – Conjunto final dos estudos selecionados

ID	Título	Referência	Base
E1	A formal approach to adaptive software: continuous assurance of non-functional requirements	(FILIERI *et al.*, 2011)	SL
E2	Software quality, non-functional software requirements and IT-business alignment	(HAIGH, 2010)	SL
E3	Decentralized learning for self-adaptive QoS-aware service assembly	(D'ANGELO *et al.*, 2020)	SD
E4	A Goal-Based Modeling Approach to Develop Requirements of an Adaptive System with Environmental Uncertainty	(CHENG *et al.*, 2009)	SL
E5	Dealing with Non-Functional Requirements for Adaptive Systems via Dynamic Software Product-Lines	(GHEZZI; SHARIFLOO, 2013)	SL
E6	Self-management of Distributed Systems Using High-Level Goal Policies	(ROSA *et al.*, 2013)	SL
E7	Modeling and verification of Functional and Non-Functional Requirements of ambient Self-Adaptive Systems	(AHMAD *et al.*, 2015)	SD
E8	An automatic process for weaving functional quality attributes using a software product line approach	(HORCAS *et al.*, 2015)	SD
E9	Providentia: Using search-based heuristics to optimize satisficement and competing concerns between functional and non-functional objectives in self-adaptive systems	(BOWERS *et al.*, 2019)	SD
E10	Self-tuning of software systems through dynamic quality tradeoff and value-based feedback control loop	(PENG *et al.*, 2012)	SD
E11	Hybrid multi-attribute QoS optimization in component based software systems	(KOZIOLEK *et al.*, 2013)	SD
E12	A Model Driven Approach to the Analysis of Quality Scenarios within Self-Adaptable SOA Systems	(PéREZ; CORREAL, 2011)	SD
E13	DeSiRE: Further Understanding Nuances of Degrees of Satisfaction of Non-functional Requirements Trade-Off	(EDWARDS; BENCOMO, 2018)	IEEE
E14	Toward a software product line for affective-driven self-adaptive systems	(GONZALEZ-SANCHEZ, 2013)	IEEE
E15	Implementing Self-Adaptive Software Architecture by Reflective Component Model and Dynamic AOP: A Case Study	(WU *et al.*, 2010)	IEEE
E16	Goal-Based Modeling of Dynamically Adaptive System Requirements	(GOLDSBY *et al.*, 2008)	IEEE
E17	Quality Criteria and an Analysis Framework for Self-Healing Systems	(NETI; MULLER, 2007)	IEEE
E18	High-quality specification of self-adaptive software systems	(LUCKEY; ENGELS, 2013)	ACM
E19	Classification of Non-Functional Requirements Attributes, and Its Representation in Functional Requirements	(NURBOJATMIKO *et al.*, 2018)	ACM
E20	Security: A Critical Quality Attribute in Self-Adaptive Systems	(PERUMA; KRUTZ, 2018)	ACM
E21	Self-Adaptive Quality Requirement Elicitation Process for Legacy Systems: A Case Study in Healthcare	(ALI *et al.*, 2017)	ACM
E22	Distributed Quality-Attribute Optimization of Software Architectures	(RAGO *et al.*, 2017)	ACM
E23	Engineering Self-Adaptive Software Systems: From Requirements to Model Predictive Control	(ANGELOPOULOS *et al.*, 2018)	ACM
E24	REM4DSPL: A Requirements Engineering Method for Dynamic Software Product Lines	(SOUSA *et al.*, 2019a)	ACM
E25	Handling Multiple Quality Attributes Trade-off in Architecture-Based Self-Adaptive Systems	(MAHDAVI-HEZAVEHI, 2016)	ACM
E26	A Framework for Evaluating Quality-Driven Self-Adaptive Software Systems	(VILLEGAS *et al.*, 2011)	ACM
E27	A Preliminary Study on Requirements Modeling Methods for Self-Adaptive Software Systems	(ZHAO *et al.*, 2013)	ACM
E28	Software Architecture Adaptability: An NFR Approach	(SUBRAMANIAN; CHUNG, 2001)	ACM
E29	Managing Non-Functional Uncertainty via Model-Driven Adaptivity	(GHEZZI *et al.*, 2013)	ACM
E30	Software Architecture Adaptability Metrics for QoS-Based Self-Adaptation	(PéREZ; CORREAL, 2011)	ACM
E31	Model-based development of dynamically adaptive software	(ZHANG; CHENG, 2006)	ACM

Fonte: Elaborado pela autora.

APÊNDICE B – CARACTERÍSTICAS DE SAS IDENTIFICADAS NO MAPEAMENTO

Tabela 7 – Características de qualidade identificadas nos estudos selecionados

Característica	**Referências**
Adequação Funcional	(SOUSA *et al.*, 2019a),
Eficiência de Desempenho	(D'ANGELO *et al.*, 2020),(MAHDAVI-HEZAVEHI, 2016), (AHMAD *et al.*, 2015), (BOWERS *et al.*, 2019), (PENG *et al.*, 2012), (HEZAVEHI *et al.*, 2017), (PéREZ; CORREAL, 2011), (GONZALEZ-SANCHEZ, 2013), (NURBOJATMIKO *et al.*, 2018), (ALI *et al.*, 2017), (RAGO *et al.*, 2017), (SOUSA *et al.*, 2019a), (VILLEGAS *et al.*, 2011), (SUBRAMANIAN; CHUNG, 2001), (ANGELOPOULOS *et al.*, 2018), (ROSA *et al.*, 2013), (HAIGH, 2010), (GHEZZI; SHARIFLOO, 2013), (HEZAVEHI *et al.*, 2017)
Compatibilidade	(EDWARDS; BENCOMO, 2018),
Usabilidade	(FILIERI *et al.*, 2011), (AHMAD *et al.*, 2015), (HORCAS *et al.*, 2015), (MAHDAVI-HEZAVEHI, 2016), (NURBOJATMIKO *et al.*, 2018), (RAGO *et al.*, 2017), (SOUSA *et al.*, 2019a), (HAIGH, 2010)
Confiabilidade	(FILIERI *et al.*, 2011), (D'ANGELO *et al.*, 2020), (GHEZZI; SHARIFLOO, 2013), (BOWERS *et al.*, 2019), (MAHDAVI-HEZAVEHI, 2016), (AHMAD *et al.*, 2015), (KOZIOLEK *et al.*, 2013), (HEZAVEHI *et al.*, 2017), (PéREZ; CORREAL, 2011), (EDWARDS; BENCOMO, 2018), (NETI; MULLER, 2007), (LUCKEY; ENGELS, 2013), (NURBOJATMIKO *et al.*, 2018), (PERUMA; KRUTZ, 2018), (ALI *et al.*, 2017), (RAGO *et al.*, 2017), (ANGELOPOULOS *et al.*, 2018), (SOUSA *et al.*, 2019a), (VILLEGAS *et al.*, 2011), (ZHAO *et al.*, 2013), (SUBRAMANIAN; CHUNG, 2001), (ZHANG; CHENG, 2006), (BOWERS *et al.*, 2019), (GOLDSBY *et al.*, 2008), (ROSA *et al.*, 2013)
Segurança	(D'ANGELO *et al.*, 2020), (MAHDAVI-HEZAVEHI, 2016), (AHMAD *et al.*, 2015), (HORCAS *et al.*, 2015), (BOWERS *et al.*, 2019), (PENG *et al.*, 2012), (KOZIOLEK *et al.*, 2013), (HEZAVEHI *et al.*, 2017), (WU *et al.*, 2010), (LUCKEY; ENGELS, 2013), (NURBOJATMIKO *et al.*, 2018), (PERUMA; KRUTZ, 2018), (RAGO *et al.*, 2017), (SOUSA *et al.*, 2019a), (VILLEGAS *et al.*, 2011), (ZHANG; CHENG, 2006), (ANGELOPOULOS *et al.*, 2018), (ROSA *et al.*, 2013)
Manutenibilidade	(MAHDAVI-HEZAVEHI, 2016), (NETI; MULLER, 2007), (NURBOJATMIKO *et al.*, 2018), (SOUSA *et al.*, 2019a), (VILLEGAS *et al.*, 2011), (PéREZ; CORREAL, 2011), (GOLDSBY *et al.*, 2008), (ROSA *et al.*, 2013), (HAIGH, 2010)
Portabilidade	(MAHDAVI-HEZAVEHI, 2016), (SOUSA *et al.*, 2019a), (HAIGH, 2010)

Fonte: Elaborado pela autora.

APÊNDICE C – SUBCARACTERÍSTICAS DE SAS IDENTIFICADAS NO MAPEAMENTO

Tabela 8 – Subcaracterísticas de qualidade identificadas nos estudos selecionados

Subaracterística	**Referências**
Adaptabilidade	(PéREZ; CORREAL, 2011), (SUBRAMANIAN; CHUNG, 2001)
Performance	(GHEZZI; SHARIFLOO, 2013), (HORCAS *et al.*, 2015), (BOWERS *et al.*, 2019), (KOZIOLEK *et al.*, 2013), (GOLDSBY *et al.*, 2008)
Disponibilidade	(HORCAS *et al.*, 2015), (KOZIOLEK *et al.*, 2013), (NURBOJATMIKO *et al.*, 2018), (VILLEGAS *et al.*, 2011), (PéREZ; CORREAL, 2011), (ROSA *et al.*, 2013), (ALI *et al.*, 2017), (MAHDAVI-HEZAVEHI, 2016)
Integridade	(HORCAS *et al.*, 2015), (NURBOJATMIKO *et al.*, 2018), (VILLEGAS *et al.*, 2011), (ZHANG; CHENG, 2006), (HAIGH, 2010)
Confidencialidade	(HORCAS *et al.*, 2015), (VILLEGAS *et al.*, 2011)
Recuperabildiade	(HORCAS *et al.*, 2015), (ZHANG; CHENG, 2006)
Tolerância a falhas	(WU *et al.*, 2010), (NETI; MULLER, 2007), (ZHANG; CHENG, 2006), (ROSA *et al.*, 2013)
Modificabilidade	(NETI; MULLER, 2007)
Maturidade	(NETI; MULLER, 2007), (HAIGH, 2010)
Capacidade	(NURBOJATMIKO *et al.*, 2018)
Interoperabilidade	(HAIGH, 2010)

Fonte: Elaborado pela autora.