INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA FLUMINENSE

PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS APLICADOS À ENGENHARIA E GESTÃO

RODRIGO OLIVEIRA ZACARIAS

UMA PROPOSTA DE DOCUMENTAÇÃO E REÚSO DE REQUISITOS BASEADA EM TESAUROS SEMÂNTICOS

Campos dos Goytacazes/RJ

2020

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA FLUMINENSE

PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS APLICADOS À ENGENHARIA E GESTÃO

RODRIGO OLIVEIRA ZACARIAS

UMA PROPOSTA DE DOCUMENTAÇÃO E REÚSO DE REQUISITOS BASEADA EM TESAUROS SEMÂNTICOS

Aline Pires Vieira de Vasconcelos

(Orientadora)

Mark Douglas de Azevedo Jacyntho

(Coorientador)

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação do Instituto Federal de Educação, Ciência e Tecnologia Fluminense, no Curso de Mestrado Profissional em Sistemas Aplicados à Engenharia e Gestão (MPSAEG), como parte dos requisitos necessários à obtenção do título de Mestre em Sistemas Aplicados à Engenharia e Gestão.

Campos dos Goytacazes/RJ

2020

Biblioteca Anton Dakitsch
CIP - Catalogação na Publicação

Z13p

Zacarias, Rodrigo Oliveira
Uma Proposta de Documentação e Reúso de Requisitos baseada em Tesauros Semânticos / Rodrigo Oliveira Zacarias - 2020.
138 f.: il. color.

Orientadora: Aline Pires Vieira de Vasconcelos
Coorientador: Mark Douglas de Azevedo Jacyntho

Dissertação (mestrado) -- Instituto Federal de Educação, Ciência e Tecnologia Fluminense, Campus Campos Centro, Curso de Mestrado Profissional em Sistemas Aplicados à Engenharia e Gestão, Campos dos Goytacazes, RJ, 2020.

1. Requisitos. 2. Tesauros Semânticos. 3. Reúso. 4. Ontologia SKOS. 5. Web Service RESTful. I. Vasconcelos, Aline Pires Vieira de, orient. II. Jacyntho, Mark Douglas de Azevedo, coorient. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da Biblioteca Anton Dakitsch do IFF com os dados fornecidos pelo(a) autor(a).

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA FLUMINENSE

PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS APLICADOS À ENGENHARIA E GESTÃO

RODRIGO OLIVEIRA ZACARIAS

UMA PROPOSTA DE DOCUMENTAÇÃO E REÚSO DE REQUISITOS BASEADA EM TESAUROS SEMÂNTICOS

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação do Instituto Federal de Educação, Ciência e Tecnologia Fluminense, no Curso de Mestrado Profissional em Sistemas Aplicados à Engenharia e Gestão (MPSAEG), como parte dos requisitos necessários à obtenção do título de Mestre em Sistemas Aplicados à Engenharia e Gestão.

Aprovado em 14 de fevereiro de 2020.

Banca Examinadora:

Aline Pires Vieira de Vasconcelos
Doutora em Engenharia de Sistemas e Computação – IF Fluminense
(Orientadora)

Mark Douglas de Azevedo Jacyntho
Doutor em Informática – IF Fluminense
(Coorientador)

Luiz Gustavo Lourenço Moura
Doutor em Engenharia de Sistemas e Computação – IF Fluminense

Rodrigo Pereira dos Santos
Doutor em Engenharia de Sistemas e Computação – UNIRIO

AGRADECIMENTOS

A Deus, por ter me abençoado durante toda a jornada do mestrado, com saúde física e mental para encarar os desafios, principalmente na etapa final, e por me proteger durante as inúmeras viagens sozinho de carro para Campos durante o curso.

Aos meus pais, Pedro e Maria, e ao meu irmão, Ricardo, por sempre torcerem por mim e pelo apoio incondicional em todos os momentos de frustração e insegurança, e por serem compreensivos nos momentos de minha ausência para me dedicar à dissertação.

À minha orientadora Aline, por todo o conhecimento compartilhado. Sempre dedicada, paciente e atenciosa, me conduziu com maestria durante toda a dissertação.

Ao meu co-orientador Mark Douglas, pela disponibilidade e pela inspiração de grande parte da pesquisa, além todo o conhecimento compartilhado.

À minha amiga Luciana Costa e sua família, por terem me acolhido em Campos durante o primeiro ano do mestrado, pelo café e pelas conversas que aliviavam a tensão das atividades do curso. Sem dúvida, foram essenciais durante minha jornada.

Aos amigos do MPSAEG, por todos os momentos de descontração e trabalho em equipe durante as aulas e os artigos.

Ao corpo docente do MPSAEG, por todo o conhecimento passado durante as aulas que contribuíram para o meu crescimento acadêmico, profissional e pessoal.

E a todos que sempre torceram por mim, o meu muito obrigado.

RESUMO

Problemas relacionados à especificação de requisitos, tais como ambiguidade e incompletude, ainda são recorrentes nos processos de desenvolvimento de sistemas. O reúso de requisitos é um dos mecanismos que pode auxiliar na redução desses contratempos. Nesse sentido, o objetivo desta dissertação é propor a criação de tesauros semânticos para documentação de requisitos em domínios específicos, utilizando as tecnologias e padrões da Web Semântica, bem como publicá-los para reúso, de acordo com os princípios *Linked Data* (Dados Ligados), quer seja na Web, quer seja dentro da Intranet de uma organização. Para descrição formal desses tesauros, é proposto um *template* para a especificação de requisitos de forma estruturada aliado ao uso do vocabulário *Simple Knowledge Organization System* (SKOS) como ontologia principal. Esse modelo ontológico fornece um conjunto de axiomas e propriedades voltados para criação de tesauros, permitindo documentar de forma precisa e fidedignamente, em um grafo de conhecimento, a definição, hierarquia e outros inter-relacionamentos entre os requisitos elicitados de um sistema. Diretrizes metodológicas são propostas para o processo de construção desse tipo de tesauro, por meio da decomposição e construção de uma taxonomia de conceitos que representam os requisitos em questão. Tesauros semânticos contribuem para acesso, recuperação e reutilização precisa dessas informações tanto por humanos quanto por máquinas, agregando mais agilidade, qualidade e produtividade ao processo de desenvolvimento. Também é apresentado um protótipo de web service RESTful conectado a um banco de dados de grafos para funcionar como repositório para reúso e demonstrar a proposta na prática. É descrito um estudo sobre a viabilidade para implementação da proposta realizado com profissionais de tecnologia da informação, onde foi promovida uma discussão em grupo e posterior preenchimento individual de um questionário alinhado aos objetivos desta dissertação. O estudo obteve resultados favoráveis, em sua maioria, e sugestões de melhoria, sendo que os participantes consideraram a proposta como relevante para a Engenharia de Requisitos e com potencial de expansão, uma vez que as diretrizes apresentadas permitem a criação de novas formas de inferência e de verificação de rastreabilidade sobre os requisitos armazenados.

Palavras-chave: Requisitos. Tesauros Semânticos. Reúso. Ontologia SKOS. Web Service RESTful.

ABSTRACT

Problems related to the specification of requirements, such as ambiguity and incompleteness, are still recurrent in the system development processes. Requirements reuse is one of the mechanisms that can help to reduce these setbacks. As such, the objective of this master thesis is to propose the creation of semantic thesauri for requirements documentation in specific domains, using the technologies and standards of the Semantic Web, as well as to publish them for reuse, according to the Linked Data principles, whether on the Web or within an organization's Intranet. For a formal description of these thesauri, a template is proposed for requirements specification in a structured way combined with the use of the Simple Knowledge Organization System (SKOS) vocabulary as the main ontology. This ontological model provides a set of axioms and properties aimed at creating thesauri, allowing to accurately and reliably document, in a knowledge graph, the definition, hierarchy and other interrelationships between system requirements. Methodological guidelines are proposed for the construction process of this type of thesaurus, through the decomposition and construction of a taxonomy of concepts that represent the requirements in question. Semantic thesauri contribute to accurate access, recovery and reuse of this information by both humans and machines, adding more agility, quality and productivity to the development process. It is also presented a RESTful web service prototype connected to a graph database to function as a repository for reuse and to demonstrate the proposal in practice. A feasibility study for implementing the proposal is described, carried out with information technology professionals, where a group discussion was promoted and subsequent individual filling out of a questionnaire aligned with the objectives of this master thesis. The study obtained mostly favorable results and suggestions for improvement, and the participants considered the proposal to be relevant to Requirements Engineering and with potential for expansion, since the guidelines presented allow the creation of new forms of inference and traceability check on stored requirements.

Keywords: Requirements. Semantic Thesaurus. Reuse. SKOS Ontology. RESTful Web Service.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

API – *Application Programming Interface*

BD – Banco de Dados

ER – Engenharia de Requisitos

ES – Engenharia de Software

HTML – *Hypertext Markup Language*

HTTP – *Hypertext Transfer Protocol*

IDE – *Integrated Development Environment*

IEEE – *Institute of Electrical and Electronics Engineers*

JSON – *JavaScript Object Notation*

LOD – *Linked Open Data Cloud Diagram*

MOF – *Meta Object Facility*

NISO – *National Information Standards Organization*

RDF – *Resource Description Framework*

REST – *Representational State Transfer*

RF – Requisito Funcional

RNF – Requisito Não Funcional

SI – Sistema de Informação

SOC – Sistema de Organização do Conhecimento

SKOS – *Simple Knowledge Organization System*

SOAP – *Simple Object Access Protocol*

TI – Tecnologia da Informação

URI – *Uniform Resource Identifier*

W3C – *World Wide Web Consortium*

XML – *Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta o contexto e os objetivos da proposta, juntamente com a sua justificativa. Em seguida, apresenta os procedimentos metodológicos e, por fim, a estrutura deste documento.

1.1 Contextualização

É evidente a contínua evolução tencológica na sociedade contemporânea e a necessidade de adaptação a essas mudanças constantes é cada vez maior. Nos setores de Tecnologia da Informação (TI), os reflexos dessas transformações são mais perceptíveis, seja na infraestrutura do hardware ou na capacidade de processamento dos sistemas de software. Com isso, torna-se indispensável o desenvolvimento de técnicas e padrões capazes de apoiar esse processo evolutivo (FREITAS JUNIOR; JACYNTO, 2016).

Segundo Isotani *et al.* (2015), a identificação e a aplicação de padrões de produção de software que proporcionem um direcionamento que facilite o processo de criação de sistemas em diferentes domínios é um tópico relevante de pesquisa na área de desenvolvimento. Isso é cada vez mais perceptível quando se leva em consideração o crescimento de ambientes de desenvolvimento distribuídos geograficamente, onde há fragmentação de recursos, infraestrutura, pessoal e informação. Nesse âmbito, a coleta, o armazenamento e o gerenciamento adequado de dados e informação são tarefas desafiadoras.

O desenvolvimento de software é composto pelas etapas de elicitação e análise de requisitos, projeto, desenvolvimento, teste e implantação (PRESSMAN, 2011). Goés *et al.* (2013) destacam que a etapa de elicitação de requisitos é uma das mais importantes do ciclo de vida do software e deve ser realizada cuidadosamente, pois caso o contrário pode levar o projeto ao fracasso.

A identificação de problemas nas fases iniciais do projeto é crucial, pois mudanças contínuas nos requisitos podem resultar em alto percentual de ocorrência de falhas. Isso permite a adoção de uma solução de baixo custo e com menor impacto no escopo, tendo em vista o menor número de mudanças que são realizadas (MONTEQUIN *et al.*, 2014). A documentação de requisitos pode ser uma tarefa complexa, visto que é necessário conhecimento sobre o domínio de aplicação do sistema (KAIYA *et al.*, 2010). Por isso, Palomares *et al.* (2017) destacam que o reúso auxilia na minimização desse problema.

A partir disso, Diamantopoulos e Symeonidis (2018) destacam o reúso de software como um conceito importante que agrega diversos benefícios ao projeto. O reúso pode ocorrer a qualquer momento de um projeto de software e reduz o tempo e esforços empregados em cada

fase do mesmo e, consequentemente, aumenta a produtividade e qualidade no desenvolvimento do software.

Para Goldin e Berry (2015) o reúso de requitos, na fase inicial do projeto, pode afetar de maneira benéfica a reutilização dos demais componentes e artefatos do software produzido. Segundo Pacheco *et al.* (2017), quanto maior o nível de abstração em que ocorre o reúso, mais benefícios são agregados.

No entanto, garantir que requisitos com qualidade considerável sejam disponibilizados para reutilização é um dos desafios para a prática da Engenharia de Requisitos (ER) (PALOMARES *et al.*, 2017). Os requisitos documentados devem possuir completude, consistência, legibilidade, não conter ambiguidades, dentre outras características que assegurem um nível de qualidade que permita que o projeto atinja seus objetivos de forma precisa e consistente (BROWN *et al.*, 2016). Para que o reúso dos requisitos ocorra de forma eficiente, é necessária a implementação de mecanismos que permitam a disseminação e a recuperação dessas informações.

Os aspectos relacionados à classificação da informação a ser compartilhada, permitindo a recuperação de forma cada vez mais automatizada, vêm sendo baseados em atributos semânticos da linguagem. Nesse âmbito, o uso de Sistemas de Organização do Conhecimento (SOCs) contendo relacionamentos semânticos e conceituais, como vocabulários controlados, taxonomias, classificações e tesauros, auxilia na representação de um domínio do conhecimento e na interpretação semântica correta das terminologias (ALTOUNIAN; GOMES, 2016).

Isotani *et al.* (2015) afirmam que essa é uma questão que pode ser abordada com o uso extensivo de tecnologias da Web Semântica, como as ontologias, que permitem a aquisição, especificação e representação formal do conhecimento em Engenharia de Software (ES), essencial para o bom andamento dos projetos de sistemas.

O contexto da Web Semântica (também conhecida como Web 3.0), apresentada por Berners-Lee *et al.* (2001), se destaca por utilizar a web não apenas para compartilhar informação, mas também o significado atrelado à mesma. Ao associar aos documentos publicados na web um conjunto de declarações estruturadas (chamadas de metadados) e inteligíveis por máquinas, permitem que as mesmas compreendam o conteúdo desses documentos e possibilitam a realização de processamentos e tomadas de decisão com base nisso.

Estendendo o contexto da Web Semântica de forma mais pragmática, Berners-Lee *et al.* (2006) introduzem o conceito de *Linked Data* (Dados Interligados) com intuito de permitir que, além da publicação de metadados associados aos documentos, seja possível interligá-los por

meio de links ou relações semânticas, contruindo a chamada Web de Dados, uma evolução da web de documentos tradicional. Seria um ambiente totalmente compreendido ou interpretado de forma inteligível tanto por humanos quanto por máquinas, com essas contribuindo para o aprimoramento da recuperação de informação na web (HEATH; BIZER, 2011).

Nesse contexto, o domínio de requisitos de software pode se beneficiar com o movimento *Linked Data* com a criação de estruturas que permitam a organização desse conhecimento de forma sistematizada e hierarquizada. Por meio de tesauros com anotação semântica de recursos, é possível organizar os requisitos, de forma sumarizada, de acordo com domínio de atuação de um Sistema de Informação (SI), facilitando sua descoberta, integração e reúso; e possibilitando também incorporar a rastreabilidade de requisitos ou outras informações relacionadas a um projeto, um aspecto importante para o desenvolvimento de sistemas.

1.2 Objetivos

1.2.1 Objetivo Geral

Nesse contexto, esta dissertação tem o objetivo de desenvolver uma proposta de documentação de requisitos baseada em tesauros, com anotação semântica de recursos, para publicação na web e, assim, potencializar o seu reúso em outros projetos de um mesmo domínio.

1.2.2 Objetivos Específicos

Para atingir o objetivo geral, foram estabelecidos os seguintes objetivos específicos:

- Identificar padrões de descrição e especificação de requisitos e estabelecer relacionamentos entre os mesmos;
- Propor um modelo (*template)* para descrição e recuperação de requisitos para ser aplicado no contexto da web semântica;
- Investigar um mecanismo adequado para aferir e documentar aspectos relacionados à qualidade dos requisitos;
- Propor diretrizes para a criação de uma estrutura de organização taxonômica dos artefatos que permitam classificar, buscar e recuperar, de forma precisa, a definição de requisitos de software em conceito de tesauros, com auxílio de ontologias e de acordo com os princípios *Linked Data*;
- Desenvolver um web service RESTful que permitam que máquinas acessem e compreendam o conteúdo dos tesauros de maneira inteligente;
- Contribuir para a organização do conhecimento de um domínio de requisitos, bem como

oferecer uma taxonomia detalhada que possibilite checar a rastreabilidade entre informações e entre cada componente do sistema, atestando, pois, sua real necessidade;

- Criar mecanismos que permitam o acesso geograficamente distribuído, futuramente na Web ou em uma Intranet, aos tesauros de requisitos para promover a reutilização do seu conteúdo de forma simples e eficiente.

1.3 Justificativa

Não é novidade que um número considerável de casos de insucesso em projetos de software estão relacionados a requisitos mal compreendidos, seja pela falta de padronização, completude ou baixa qualidade, sendo um fator ainda recorrente nos ambientes de desenvolvimento (BROWN *et al.*, 2016).

Barcelos (2016) destaca que a compreensão e a especificação de requisitos estão entre as tarefas mais difíceis enfrentadas pelos profissionais da área. Como normalmente os requisitos são expressos em linguagem natural, é comum surgir os problemas relacionados à ambiguidade e completude. Por isso, Palomares *et al.* (2017) destacam que o reúso auxilia na minimização desse problema e contribui para a redução de esforço e tempo no desenvolvimento.

Essa abordagem surge a partir da complexidade existente na obtenção de requisitos em determinados domínios de conhecimento, tendo em vista a dificuldade dos *stakeholders* em expressar suas necessidades e dos profissionais, sem experiência naquele domínio, em entendê-las completamente (KAIYA *et al.*, 2010). Xavier (2001) definiu domínio como uma coleção de artefatos e componentes de software especializados em uma determinada tarefa com o intuito de fornecer aos usuários um conjunto de funções em relação a uma área de conhecimento em particular. Segundo Gualhano *et al.* (2014), o domínio dos requisitos é definido como um conjunto que pode ser particionado em conjuntos menores com as seguintes características: ser completo, independente, bem definido e único.

Além disso, há a dificuldade de encontrar na Web uma estrutura que facilite a especificação de requisitos para a publicação e reúso de forma que a semântica de seu conteúdo também possa ser compreendida pelas máquinas. Na web tradicional, essas informações são publicadas por meio de documentos HTML (*Hypertext Markup Language* ou Linguagem de Marcação de Hipertexto, em português) e cabe aos humanos todos os esforços para a consulta, integração e interpretação da informação dessa fase da ER, o que pode implicar em processos demorados e suscetíveis a erros. Com a anotação semântica, as máquinas podem prover um processamento semi-autoamizado e agilizar esses procedimentos.

Neste trabalho, opta-se pela criação de tesauros de requisitos, pois seu uso de forma

automática pode melhorar os resultados de buscas na web. Um tesauro é um SOC que classifica e estrutura conceitos em formas hierárquica, equivalente ou associativa. A delimitação das relações entre os termos indexados (requisitos, neste caso) aumenta a precisão dos mecanismos de recuperação e, quanto mais específicos são os domínios dos tesauros, melhores podem ser os resultados obtidos. Além disso, essa estruturação do conhecimento de forma taxonômica contribui para a redução de ambiguidades e melhoria da completude e da consistência das informações compartilhadas. Isso permite o seu processamento racional ou inferições por máquinas em um domínio de conhecimento (ALTOUNIAN; GOMES, 2016).

A disponibilização dos requisitos corretamente especificados seguindo um padrão na web pode facilitar a busca e o compartilhamento dessas informações pelas equipes de projeto e desenvolvimento de software. A ideia central é que, futuramente, a taxonomia semântica de requisitos seja utilizada como espinha dorsal do sistema, utilizada para categorizar todos os artefatos produzidos ao longo do ciclo de vida do sistema, desde sua especificação até sua implementação.

Ao ser armazenado em um banco de dados semântico, enriquecido e publicado na Web (ou na Intranet) de acordo com princípios *Linked Data*, o tesauro poderá fomentar o reúso por diversas aplicações. Além disso, por meio da criação de um web service RESTful semântico e sua disponibilização de forma aberta, esse conhecimento pode ser estendido. Com isso, a comunidade pode contribuir para o crescimento da abordagem proposta com inserção de novos dados ou criando mecanismos de manipulação dos dados já existentes, tais como estratégias de consulta ou mineração, a fim de atender os mais variados objetivos organizacionais, sendo uma das principais finalidades deste trabalho.

1.4 Metodologia

Os procedimentos metodológicos para a elaboração desta dissertação são:

1. **Pesquisas Bibliográficas:** consiste no estudo bibliográfico para construção da fundamentação teórica referente à abordagem proposta. Esta etapa é focada em processos e conceitos de Engenharia de Requisitos, Tesauro, Tecnologias da Web Semântica, Reúso de Artefatos de Software e Web Service RESTful. Neste momento, também são selecionados os trabalhos correlatos que reunam dois ou três temas e que permitam obter um panorama sobre o estado da arte dessas áreas.
2. **Análise:** consiste na investigação, a partir da fundamentação teórica e dos trabalhos correlatos, sobre padrões de descrição e especificação de requisitos que servirão como base para a criação do *template* semântico que será implementado no tesauro.

3. **Estruturação:** consiste na criação do *template* para a descrição e especificação de requisitos no contexto semântico. Neste momento também é feito o planejamento da estrutura e das relações dos termos que irão compor os tesauros de conceitos da ER.
4. **Desenvolvimento:** consiste na implementação do *template* criado e na construção dos tesauros com o auxílio de ontologias. Neste momento é desenvolvido um protótipo de web service RESTful para cadastro dos requisitos e realização de consultas ao repositório de dados.
5. **Avaliação:** consiste no planejamento e na realização de um estudo de viabilidade, onde profissionais ligados à área de desenvolvimento do software são apresentados à proposta e ao protótipo para avaliar os processos de criação e consulta dos tesauros. Posteriormente, é preenchido um questionário para documentar a avaliação.
6. **Disponibilização:** consiste no compartilhamento do código implementado no desenvolvimento protótipo e futura disponibilização do web service em um domínio da web para fomentar sua utilização de forma colaborativa.

1.5 Estrutura do Trabalho

A partir desta introdução, o presente trabalho está estruturado da seguinte forma:

No Capítulo 2, é apresentada a revisão bibliográfica dos temas relacionados ao objetivo deste trabalho, sendo eles: Engenharia de Requisitos, Tesauro, Tecnologias da Web Semântica, Reúso de Software e Web Service RESTful.

Em seguida, no Capítulo 3 são levantados os trabalhos relacionados existentes na literatura, no que se refere à aplicação de Tesauros na ER, a Tecnologias da Web Semântica aplicadas à ER e Repositórios para Reúso de Componentes de Software.

No Capítulo 4, é proposto o *template* para especificação de requisitos no contexto semântico, elaborado com base em outros modelos e padrões existentes na literatura. Em seguida, são descritos os processos para a elaboração e construção dos tesauros.

O Capítulo 5 apresenta o protótipo de web service RESTful desenvolvido para atestar, na prática, a estruturação dos requistos em tesauros semânticos. Assim, é descrita a aplicação no método apresentado no capítulo anterior, apresentada a arquitetura e o desenvolvimento do web service e seus serviços REST.

No Capítulo 6, é descrito o estudo de viabilidade realizado com profissionais de TI para avaliar a proposta desta dissertação.

Por fim, no Capítulo 7 são apresentadas as considerações finais, contribuições, limitações e os trabalhos futuros para este trabalho.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta, por meio de pesquisa bibliográfica, a fundamentação teórica adotada para a construção da abordagem proposta. Desse modo, a seção inicial apresenta os conceitos relacionados à ER, com enfoque na descrição e especificação de requisitos. Nas seções posteriores, são apresentados tópicos sobre Tesauro e Tecnologias da Web Semântica, bem como os conceitos de Ontologias e dos princípios *Linked Data*. Por fim, são apresentadas definições relativas a repositórios para Reúso de Artefatos de Software, assim como uma explanação sobre Web Service RESTful, conceito que é aplicado no protótipo desta pesquisa.

2.1 Engenharia de Requisitos

Um projeto de software alcança o sucesso quando o produto final cumpre o objetivo de atender às necessidades do ambiente para o qual foi desenvolvido. Na ES, a área que estabelece as diretrizes, condições e especificações para esse propósito é a Engenharia de Requisitos.

Olaronke *et al.* (2018) ressaltam que diferentes autores veem requisitos de software em diferentes contextos, não havendo uma definição precisa para o termo. Uma das definições presentes no glossário do *Institute of Electrical and Electronics Engineers* – IEEE (1990) apresenta requisito como uma condição ou capacidade que deve ser atendida ou possuída por um sistema ou componente do sistema para atender a um contrato, padrão, especificação ou outro documento formalmente imposto.

Sommerville (2011) define requisito como uma descrição do que o sistema deve fazer, bem como quais serviços deve oferecer e as restrições para o seu funcionamento. O principal objetivo é refletir a necessidade do cliente para que um sistema execute uma funcionalidade determinada.

Com base nessas definições, a ER é apresentada como uma área integrante da ES, sendo um processo ou conjunto de atividades, técnicas, métodos e práticas que auxiliam em atividades relativas ao desenvolvimento, documentação, gerenciamento e manutenção do conjunto de requisitos de um sistema (SOMMERVILLE, 2011).

De acordo com Pressman (2011), a ER é uma importante etapa da ES e fornece mecanismos apropriados para entender as necessidades do cliente, verificando a viabilidade do projeto e negociando a melhor solução para o que o cliente deseja.

Ainda pode ser definida como um processo sistemático que possui um grande número de responsabilidades sobre tarefas de elicitação, análise, especificação, validação e gerenciamento de requisitos durante todo o processo de desenvolvimento de software (NETO, 2008).

Quanto ao processo, Sommerville (2011) descreve a ER em quatro atividades principais que estão interligadas iterativamente por meio de ciclos, conforme a Figura 1. Em seguida, essas atividades fases são descritas de forma sucinta.

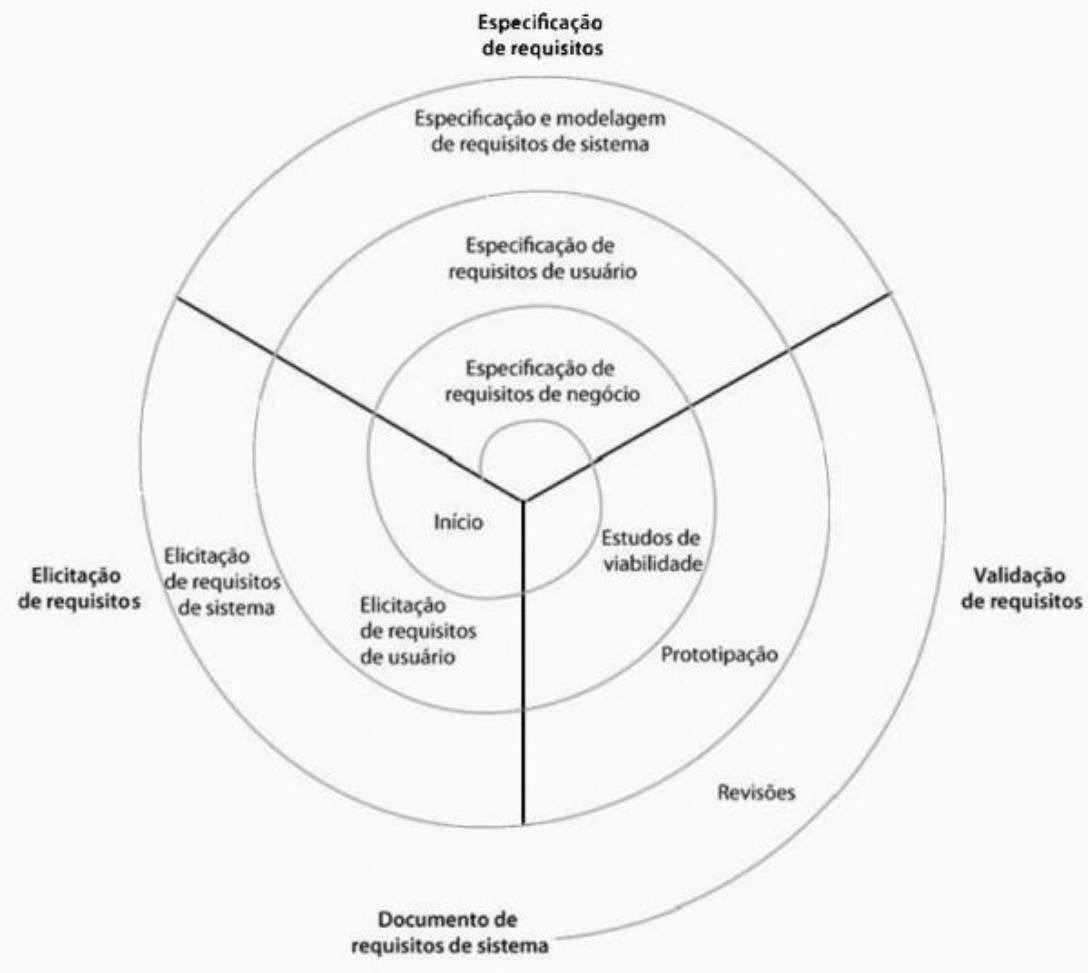


Figura 1. O processo de Engenharia de Requisitos.
Fonte: Sommerville (2011).

Estudo de viabilidade: É a primeira atividade do processo e se concentra em definir os objetivos, prazos e custos para o projeto, bem como avaliar a utilidade do produto final para a organização.

Elicitação e Análise de Requisitos: Essa atividade se inicia após os estudos de viabilidade. Aqui ocorre a descoberta dos requisitos partir de informações sobre o domínio do sistema e das necessidades do cliente. Existem diversas abordagens para esse fim, tais como: entrevistas estruturadas e não estruturadas, observação do comportamento do usuário, prototipação, *brainstorming*, análise de textos, entre outros (PRESSMAN, 2011). Em seguida, os mesmos são organizados, classificados e priorizados.

Especificação de Requisitos: Nesta etapa é feita a descrição, por meio de linguagem

natural ou de diagramas, dos requisitos funcionais e não funcionais elicitados na etapa anterior. Esse tópico é abordado com mais detalhes nas subseções de tipos de requisitos.

Validação de Requisitos: Essa atividade corresponde à realização de uma série de verificações dos requisitos considerando os critérios de atendimento às necessidades do cliente. Esta ligada à análise, uma vez que erros podem ser detectados e precisarão ser corrigidos. O produto final é o Documento de Requisitos, que descreve todas as condições do projeto e deve apresentar níveis de qualidade satisfatório.

Na visão de Pressman (2011), esse processo é dividido em sete etapas distintas, a saber: concepção, levantamento, elaboração, negociação, especificação, validação e gestão dos requisitos, de acordo com a necessidade do cliente. Apesar do número distinto de etapas, as atividades são as mesmas.

2.1.1 Tipos de Requisitos

Segundo Gambo *et al.* (2014), o requisito deve agregar valor aos seus usuários, bem como expressar as limitações nas escolhas que os desenvolvedores fazem ao implementar o software a partir de um documento de especificação.

Conforme o PMBOK (2013), os requisitos podem ser categorizados em: Requisitos de Negócio, Requisitos de *Stakeholder*, Requisitos da Solução (Funcionais e Não Funcionais), Requisitos de Transição, Requisitos de Projeto e Requisitos de Qualidade. Neste trabalho são abordadas, inicialmente, as taxonomias referente a requisitos funcionais e não funcionais, restando as demais categorias para serem inseridas em trabalhos futuros.

2.1.1.1 Requisitos Funcionais

Para Dabbagh *et al.* (2016), um requisito funcional (RF) descreve um comportamento funcional que um sistema ou artefato deve ser capaz de executar. Os requisitos funcionais descrevem e definem quais atividades o sistema deve executar para fornecer aos seus usuários uma funcionalidade requisitada.

Os RFs descrevem os serviços ou funcionalidades que serão implementados no sistema. Geralmente, os mesmos são descritos de acordo com a visão do cliente em relação as regras de seu negócio (OLARONKE *et al.*, 2018). Sommerville (2011) destaca que, além de descrever os serviços que um sistema deve oferecer, também declaram como um sistema deve reagir a entradas específicas e como deve se comportar em determinadas situações, podendo também explicitar o que um sistema não deve fazer.

2.1.1.2 Requisitos Não Funcionais

Segundo Olaronke *et al.* (2018), os requisitos não funcionais (RNFs) estão relacionados a especificações técnicas que qualificam os RFs, estipulando o quão bem o sistema deve executar as suas funções a fim de atender as necessidades do cliente. Também são conhecidos como atributos de qualidade que o software deve possuir e podem representar restrições aos serviços ou funções oferecidos pelo mesmo, abrangendo-o como um todo (SOMMERVILLE, 2011).

Dabbagh *et al.* (2016) ressaltam que os RNFs são qualidades que as funções ou o sistema devem ter, considerando que os requisitos não funcionais são basicamente ineficazes sem requisitos funcionais. Avaliando esses artefatos no momento da implementação, os RFs basicamente operam ou não, e os RNFs geralmente apresentam uma escala de valor variável de bom e ruim.

Na literatura, existem vários catálogos que contemplam os tipos de RNFs. No entanto, neste trabalho, para defini-los foi escolhida a norma ISO/IEC-25010 (2011), pois refere-se a um padrão internacional e contém tanto características quanto subcaterísticas dos requisitos não funcionais, ademais, este catálogo possui uma classificação bem definida de uma grande gama de requisitos e o mesmo abrange sua utilização para especificação de RFs e RNFs do cliente e do usuário.

De acordo com a ISO/IEC-25010, os requisitos não funcionais podem ser definidos, brevemente, da seguinte forma:

Confiabilidade: Grau para o qual um sistema, produto ou componente executa funções especificadas sob condições definidas por um período de tempo especificado. É composta das seguintes subcaracterísticas: maturidade, disponibilidade, tolerância a erro e recuperabilidade.

Segurança: Grau em que um produto ou sistema protege as informações e os dados para que as pessoas, outros produtos ou sistemas tenham o grau de acesso aos dados adequado aos seus tipos e níveis de autorização. É composta das seguintes subcaracterísticas: confidencialidade, integridade, não repúdio, prestação de contas e autenticidade.

Eficiência: Essa característica representa o desempenho em relação à quantidade de recursos utilizados nas condições declaradas. É composta das seguintes subcaracterísticas: desempenho/comportamento do tempo, utilização de recursos e capacidade.

Manutenibilidade: Capacidade de o produto de software ser modificado. As modificações podem incluir correções, melhorias ou adaptações do software devido a mudanças no ambiente, nos requisitos ou especificações funcionais. É composta das seguintes subcaracterísticas: modularidade, reusabilidade, analisabilidade, modificabilidade e

testabilidade.

Portabilidade: Grau de eficácia e eficiência com o qual um sistema, produto ou componente pode ser transferido de um hardware, software ou outro ambiente operacional ou de uso para outro. É composta das seguintes subcaracterísticas: adaptabilidade, instalabilidade e substituibilidade.

Usabilidade: Grau em que um produto ou sistema pode ser usado por usuários específicos para atingir metas especificadas com eficácia, eficiência e satisfação em um contexto específico de uso. É composta das seguintes subcaracterísticas: reconhecimento de adequabilidade, aprendizagem, operabilidade, proteção contra erros do usuário, estética da interface do usuário e acessibilidade.

Compatibilidade: Grau para o qual um produto, sistema ou componente pode trocar informações com outros produtos, sistemas ou componentes e/ou executar suas funções necessárias, enquanto compartilha o mesmo ambiente de hardware ou software. É composta das seguintes subcaracterísticas: coexistência e interoperabilidade.

2.1.2 Especificação de Requisitos

A descrição e a especificação de requisitos não são tarefas trivais, seja para os RFs como para os RNFs. Em relação aos RFs, quando expressos como requisitos de usuário, podem ser descritos de forma mais abstrata para serem entedidos pelos usuários, sendo descritos, em sua maior parte, em linguagem natural e suplementados com diagramas no documento de requisitos. Quando expressos como requisitos de sistema, podem conter detalhes das funções do sistema, como entradas, saídas, exceções, etc (SOMMERVILLE, 2011).

Nos caso dos RNFs, Silva *et al.* (2017) ressalta que é uma atividade complexa, porque exige conhecimentos específicos em diversas áreas, como confiabilidade, desempenho, eficiência, modificabilidade, portabilidade e usabilidade. Na literatura, existem vários tipos de mecanismos utilizados para auxiliar no processo de elicitação de RNFs, pode-se citar:

-Catálogos de tipos de RNFs: lista organizada de RNFs de acordo com normas, recomendações ou padrões.

-Questões/*Checklist:* instrumento de revisão técnica formal que apresenta um conjunto de perguntas baseadas em critérios pré-determinados que tem a função de detectar erros e inconsistências em algum aterfato de software.

-Modelos (*Templates)*: estrutura pré-definida contendo atributos baseados em normas ou recomendações.

Alexander e Stevens (2002, *apud* WIEGERS; BEATTY, 2013) descrevem uma

estrutura básica para especificação de requisitos em linguagem natural: O [tipo de usuário ou nome do ator] deve ser capaz de [realizar algo] [ao objeto] [sob alguma condição]. A seguir, é apresentado um exemplo sob a perspectiva do sistema:

"O sistema deve ser capaz de atualizar o número de materiais no estoque"

Analogamente, o mesmo requisito pode ser descrito da seguinte forma sob a perspectiva do usuário:

"O almoxarife deve ser capaz de atualizar no sistema o número de materiais no estoque"

É permitida a criação de sentenças alternativas a partir dos sinônimos das palavras-chave da estrutura: "O sistema deve permitir (aceitar ou autorizar) que [tipo de usuário ou nome do ator] [fazer alguma coisa]." Wiegers e Beatty (2013) ressaltam que o importante é que um documento de requisitos apresente a especificação em um único padrão de descrição e o mais explicíto possível para reduzir interpretações errôneas.

Quanto maior o nível de detalhamento das informações relacionadas a um requisito, maior a garantia de seu entendimento, seja por parte do usuário quanto da equipe de desenvolvimento. Nesse sentido, a especificação estruturada é uma ferramenta capaz de fornecer o nível de detalhamento desejado.

Segundo Sommerville (2011), a especificação estruturada parte do princípio de uso de padrões, *templates* ou formulários que guiam o escritor durante o detalhamento das informações sobre os objetos manipulados, funções desempenhadas e eventos processados pelo sistema. O caso de uso é um dos tipos de especificações estruturadas mais utilizados.

Segundo Pressman (2011), um caso de uso é uma descrição detalhada de um cenário de interação entre um ou mais atores com uma funcionalidade de um sistema que foi determinada por meio de um requisito. Os atores podem ser pessoas ou outros sistemas que de alguma forma interagem com o sistema principal. O Quadro 1 apresenta um exemplo de *template* para elaboração de um caso de uso de um requisito de acesso à camera de vigilância via Internet.

Os casos de uso são eficazes também para elicitar novos requisitos não identificados anteriormente e que estão relacionados aos *stakeholders* que vão interagir com o sistema de forma direta. Eles se concentram em elicitar e descrever os RFs, sendo geralmente não adequados para os RNFs, devido ao foco nas interações do sistema. A definição do formato e dos atributos do *template* varia de acordo com a necessidade do projeto podendo ser adicionadas quantas informações forem julgadas necessárias, não havendo um único modelo a ser seguido. As especificações estruturadas, como o caso de uso, reduzem a variabilidade na especificação e organizam as informações de um requisito de forma eficaz (PRESSMAN, 2011; SOMMERVILLE, 2011).

Quadro 1. Exemplo de Caso de Uso de Requisito.

Casa Segura	
Caso de Uso	Acessar a vigilância por câmeras via Internet
Ator	Proprietário do imóvel
Objetivo	Visualizar imagens de câmeras espalhadas pela casa de qualquer ponto remoto via Internet.
Precondições	O sistema deve estar totalmente configurado; devem ser obtidos ID de usuário e senhas apropriadas.
Disparador	O proprietário do imóvel decide fazer uma inspeção na casa enquanto se encontra fora.
Cenário	1. O proprietário do imóvel faz o login no site Produtos da CasaSegura. 2. O proprietário introduz seu ID de usuário. 3. O proprietário introduz duas senhas (com pelo menos oito caracteres). 4. O sistema mostra os botões de todas as principais funções. 5. O proprietário seleciona a “vigilância” por meio dos botões principais. 6. O proprietário seleciona “escolher uma câmera”. 7. O sistema mostra a planta da casa. 8. O proprietário seleciona um ícone de câmera da planta da casa. 9. O proprietário seleciona o botão “visualização”. 10. O sistema mostra uma janela de visualização com o ID de câmera. 11. O sistema mostra imagens de vídeo na janela de visualização a uma velocidade de um quadro por segundo.
Exceções	1. O ID ou senhas são incorretos ou não foram reconhecidos — veja o de uso Validar ID e senhas. 2. A função de vigilância não está configurada para este sistema — o sistema mostra a mensagem de erro apropriada; veja o caso de uso Configurar função de vigilância. 3. O proprietário seleciona “Visualizar as imagens em miniatura para todas as câmeras” — veja o caso de uso Visualizar as imagens em miniatura para todas as câmeras. 4. A planta não está disponível ou não foi configurada — exibir a mensagem de erro apropriada e ver o caso de uso Configurar planta da casa. 5. É encontrada uma condição de alarme — veja o caso de uso Condição de alarme encontrada.
Pós-condições	As imagens da câmera escolhida são exibidas para o usuário via Internet, após a autenticação.

Fonte: Adaptado de Pressman (2011).

Além disso, os casos de uso podem ser expressos de forma gráfica por meio de diagramas utilizando a UML. A Figura 2 apresenta o diagrama que inclui o caso de uso descrito no Quadro 1, sendo o ator representado pelo desenho do boneco.

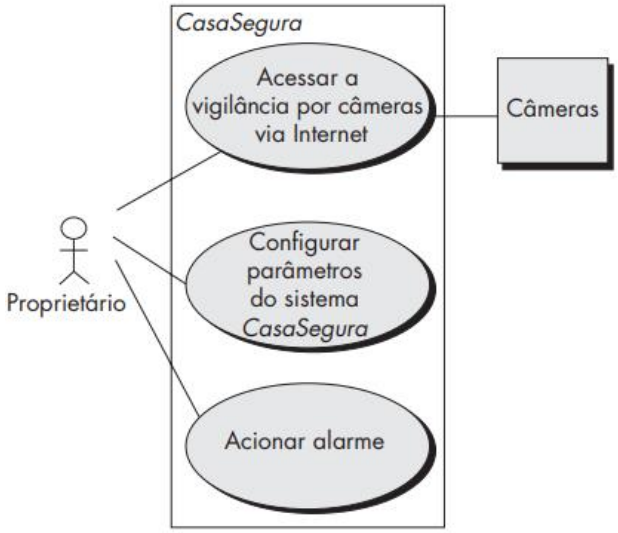


Figura 2. Diagrama de Caso de Uso para o sistema Casa Segura.
Fonte: Pressman (2011).

No presente trabalho, para a estruturação do Tesauro de Requisitos, o modelo (*template)* para a especificação estruturada dos RFs e RNFs foi o mecanismo escolhido para auxiliar neste processo. A partir disso, foi realizada uma pesquisa em bases de trabalhos científicos para identificar *templates* presentes na literatura e suas características, como apresentado na subseção a seguir.

2.1.3 Modelos Estruturados de Especificação Requisitos

Durante a pesquisa nas bases científicas, foram localizados os trabalhos acadêmicos de Barcelos (2016) e Silva (2016), que realizaram uma vasta pesquisa sobre especificação de RFs e RNFs, relacionando um conjunto de padrões e *templates* mais relevantes e utilizados para ambos os casos. A seguir, são apresentados os modelos estruturados elencados nas pesquisas dos autores.

2.1.3.1 Especificação de Requisitos no Domínio de Sistemas de Informação com o Uso de Padrões

Barcelos (2016) apresenta um conjunto de padrões elaborado para especificar RFs e regras de negócio no domínio de SI, assim como o apoio computacional baseado nesse padrões para elaboração de um documento de requisitos. O escopo do projeto não abrange os RNFs.

A metodologia para a elaboração dos padrões consistiu das seguintes etapas: (1) Obtenção de diferentes documentos de requisitos; (2) Organização dos Requisitos; (3) Análise de Requisitos; (4) Especificação do Padrão; (5) Estabelecimento de Relações entre os Padrões. A Figura 3 expressa a estrutura elaborada para a apresentação dos padrões e a Figura 4 apresenta

um exemplo de aplicação.

Elemento	Descrição
Nome	Especifica o nome do padrão, que deve ser único e refletir a aplicabilidade do padrão.
Domínio	Corresponde ao domínio de aplicação do padrão.
Propósito	Descreve o objetivo da aplicação do padrão.
Problema	Descreve a situação em que o padrão pode ser aplicado.
Consequência	Descreve as consequências de se utilizar o padrão.
Tipo	Especifica o tipo de requisito: funcional, não funcional ou regra de negócio.
Solução	Apresenta um *template* para a especificação da parte fixa e variável do requisito que o padrão deve representar. A parte fixa apresenta um texto padrão para a especificação do requisito. A notação <...> é usada para descrever a parte variável que é denominada de parâmetro e deve ser substituída pelos dados pertinentes ao requisito a ser elaborado.
Padrões Relacionados	Especifica os padrões relacionados que são complementares ao padrão em questão. Esse elemento contribui para o engenheiro de software na indicação de outros possíveis padrões que podem ser usados quando ocorrer o uso do padrão em questão.

Figura 3. Estrutura do Padrão de Especificação de Requisitos.
Fonte: Barcelos (2016).

Nome	Processar Transação
Domínio	Sistemas de Informação
Propósito	Processar e armazenar informação de uma transação.
Problema	Estabelecer quais dados da transação devem ser armazenados.
Consequência	A especificação de um requisito que descreve a necessidade de um usuário de processar uma transação e armazenar sua informação.
Tipo	Funcional
Solução	**Template**: Permitir o processamento de <transação> de <entidade>, com o armazenamento dos seguintes atributos: <atributos>.
	Parâmetro: <transação> nome da transação que contém informações do domínio (1). <entidade> nome da entidade que descreve o tipo de transação (1). <atributos> nome dos atributos da entidade que devem ser armazenados (1..*). **Sugestões de Parâmetros:** <transação> venda <entidade> produto <atributos> número, data, cliente, produto, preço unitário e preço total.
	Exemplo da Descrição do Requisito: - Permitir o processamento de venda de produto, com o armazenamento dos seguintes atributos: número, data, cliente, produto, preço unitário e preço total.
Padrões Relacionados	Informação Requerida, Condição de Execução, Limite de Execução, Calculo de Valor e Recuperar Informação.

Figura 4. Exemplo da estrutura preenchida com um padrão para o requisito “Processar Transação”.
Fonte: Barcelos (2016).

Para diminuir possíveis dificuldades com a utilização da estrutura proposta, foi elaborada uma ferramenta computacional para auxiliar tanto na documentação do padrão como

na sua recuperação para a atividade de reúso. A Figura 5 apresenta a interface da ferramenta. É possível observar que os padrões são armazenados organizados em uma estrutura de árvore.

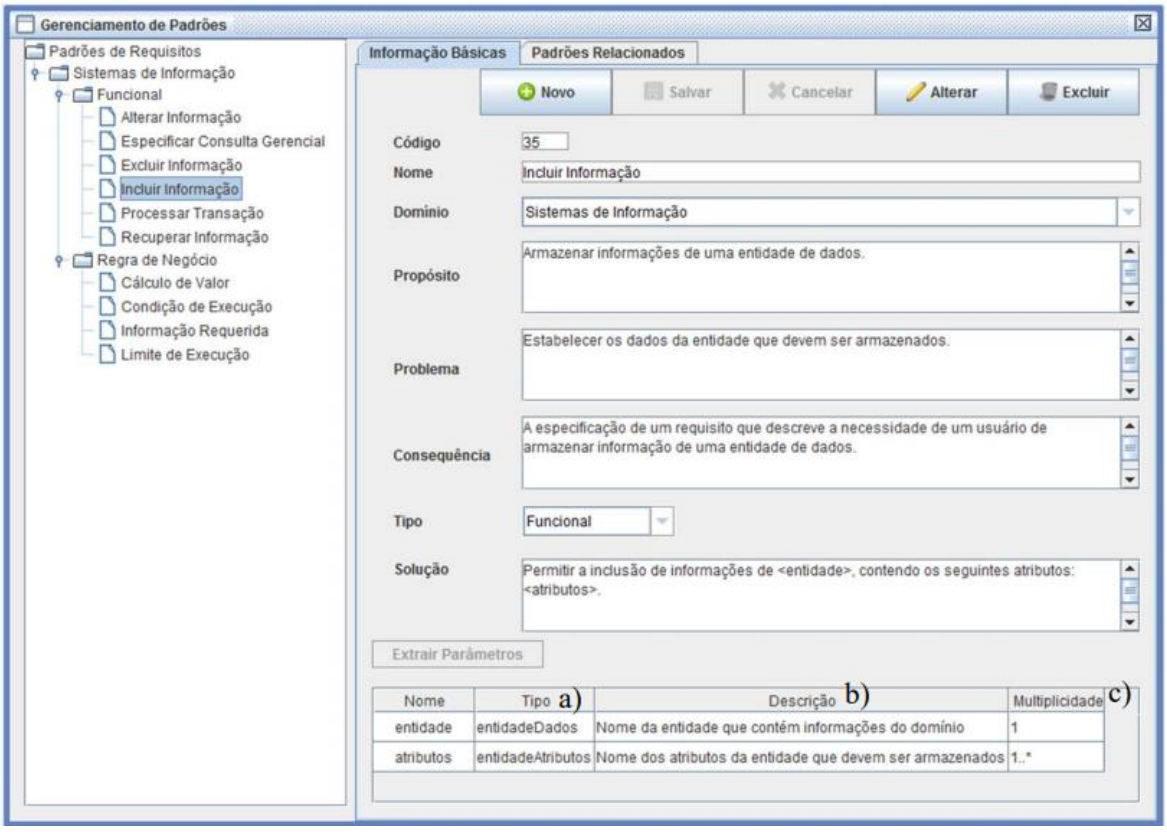


Figura 5. Interface de Especificação e Gestão de Padrões.
Fonte: Fonte: Barcelos (2016).

2.1.3.2 SENoR - Uso de Templates

Kopczynska e Nawrocki (2014) utilizam um método de Elicitação Estruturada de RNFs (SENoR). O método é composto por uma sequência de sessões curtas de *brainstorming* orientadas pelos RNFs definidos na ISO/IEC-25010. Neste trabalho é apresentado um estudo de caso que utiliza *templates* para a elicitação de RNFs. A Figura 6 apresenta um exemplo dos *templates* dos requisitos, que são formulados usando linguagem natural.

O modelo é baseado na reutilização de requisitos pré-definidos com parâmetros que serão preenchidos pelos analistas. Por exemplo em (a) tem-se uma parte opcional, seguida de uma solicitação e de uma resposta em um determinado período de tempo (parâmetro), ou seja, é apresentado o modelo a ser seguido pelo analista. Em (b) observa-se o preenchimento deste modelo pré-definidos de requisitos para um determinado projeto, como por exemplo no item 2: “Tempo limite <frequência> não deve ser mais do que <quantidade de tempo>”.

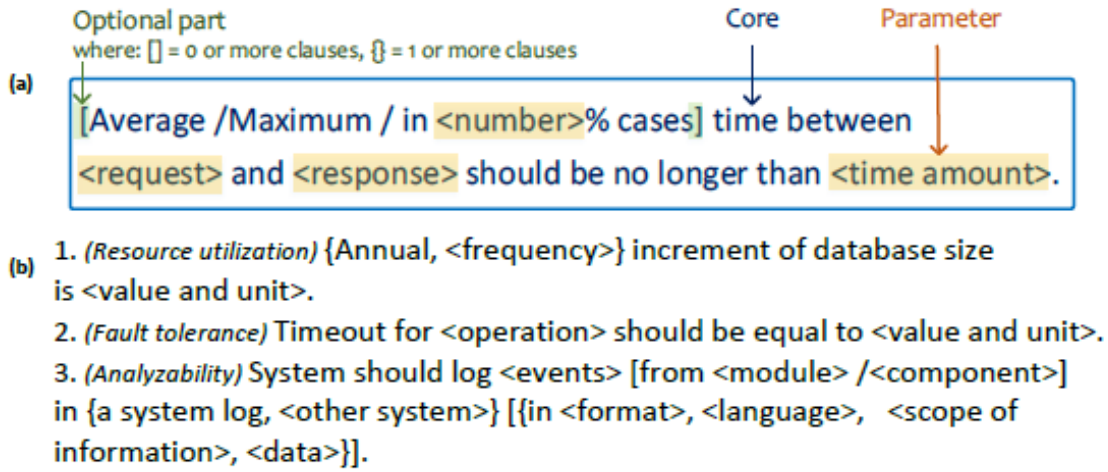


Figura 6. Template de Requisitos Não Funcional.
Fonte: Kopczynska e Nawrocki (2014).

Vale ressaltar que novos *templates* podem ser criados e reutilizados em outros projetos de software, porém é necessária a participação de especialistas durante a elicitação em um determinado projeto.

2.1.3.3 ElicitO

O trabalho de Al Balushi *et al.* (2013) apresenta um *framework* que auxilia na elicitação, validação e priorização de requisitos. Utiliza como parâmetro ontologias baseadas em requisitos de qualidade e contém uma base de requisitos previamente elicitados. Uma forma de estruturar a ontologia é através de entrevista com especialistas do domínio, além de outras abordagens.

Na Figura 7 é apresentada a tela que descreve os RNFs, no qual são exibidos os dados referentes a elicitação de requisitos. Na ferramenta, o usuário escolhe inicialmente a base de dados da busca, e posteriormente características e subcaracterísticas de qualidade e a métrica desejada. Além disso, existem outras abas para inserção de outros dados referentes aos RNFs, como, por exemplo, a descrição do requisito.

A proposta deste trabalho é que a base de dados seja alimentada inicialmente com requisitos apresentados na norma ISO/IEC-9126 (2011) (permite que a qualidade do produto de software seja especificada e avaliada em diferentes perspectivas pelos envolvidos com aquisição, requisitos, desenvolvimento, uso, avaliação, apoio, manutenção, garantia de qualidade e auditoria de software). De acordo com os autores, a escolha pela norma supracitada está relacionada à necessidade de um vocabulário comum para os analistas. Assim, a cada processo de elicitação, o BD será atualizado com os novos requisitos identificados.

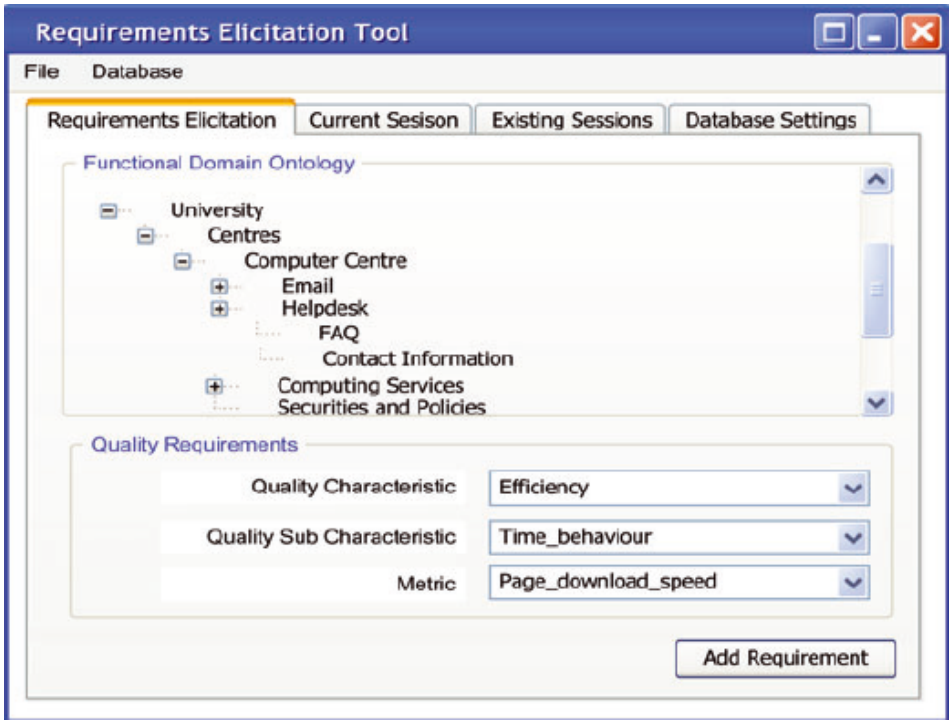


Figura 7. Modelo para definição de RNF (ElicitO).
Fonte: Al Balushi *et al.* (2013)

2.1.3.4 Abordagem de Juristo

Juristo *et al.* (2007) apresentam padrões de usabilidade com o objetivo de apoiar a elicitação de RNFs. No trabalho foi proposta uma abordagem baseada no desenvolvimento de diretrizes para auxiliar no processo de elicitação e especificação do RNF de usabilidade. A Figura 8 mostra um exemplo do modelo utilizado pela ferramenta da proposta.

IDENTIFICATION
Name: System Status Feedback
Family: Feedback
Alias: Status Display [51] Modelling Feedback Area [15]
PROBLEM
Which information needs to be elicited and specified in order to provide users with system status information.
CONTEXT
When changes that are important to the user occur or When failures that are important to the user occur, for example: - During task execution - Because there are not enough system resources - Because external resources are not working properly. Examples of status feedback can be found on status bars in windows applications; train, bus or airline schedule systems; VCR displays; etc.

Figura 8. Modelo para definição de RNFs de Juristo.
Fonte: Juristo *et al*. (2007).

As informações são coletadas e fornece aos analistas de requisitos um repositório de

conhecimento que, por sua vez, formam um conjunto de padrões reutilizáveis, que auxiliam na elicitação e especificação de características funcionais de usabilidade. Os autores salientam que este repositório está relacionado apenas com usabilidade. Na Figura 8, o exemplo é composto inicialmente por uma identificação do requisito, posteriormente é apresentado o problema que ocasionou a necessidade da criação desse requisito, ou seja, a partir desse atributo é possível definir quais RNFs estarão presentes no sistema. O atributo contexto apresenta a solicitação a ser realizada de acordo com o problema proposto.

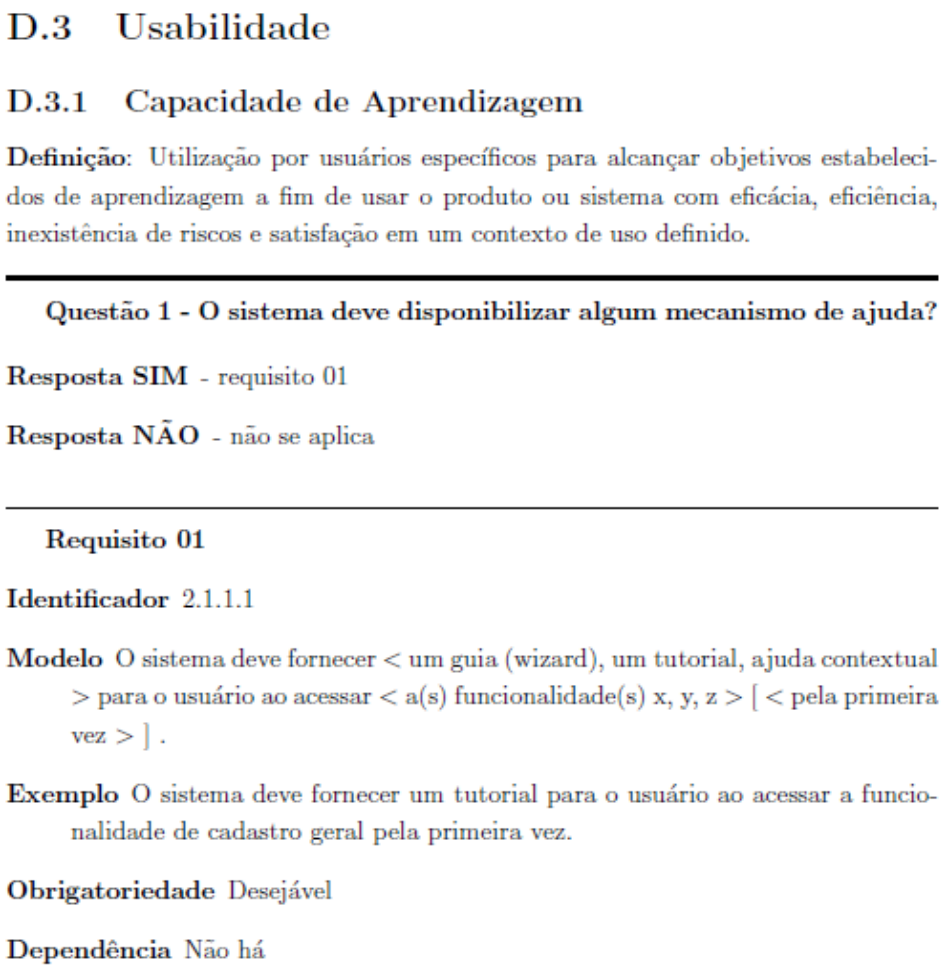
2.1.3.5 ADEG-NFR

Silva (2016) propõe um guia (ADEG-NFR) para auxiliar a definição e elicitação dos RNFs. Esse guia consiste em um conjunto de perguntas, modelos e exemplos de requisitos com o objetivo de facilitar o processo de elicitação dos mesmos.

Como *template* para os RNFs, os autores propõem um modelo a ser definido que permite ao engenheiro de requisitos obter do cliente informações específicas que são particulares do negócio do cliente. Para tanto, o modelo definido envolve, quando aplicável, um conjunto de parâmetros que podem ser modificados a cada projeto elicitado com o uso do guia, apresentado na Figura 9. Um exemplo prático do *template* proposto por Silva (2016) é apresentado na Figura 10.

Atributo	Descrição
Identificador	Identificador único do requisito no guia.
Tipo de Requisito	Classe de requisitos a qual o requisito pertence, selecionada dentre os tipos definidos.
Conceitos	Definição do significado do tipo do requisito, descrita preferencialmente com base em normas, padrões, artigos técnicos, livros ou outras referências reconhecidas.
Questão	Questionamento feito ao cliente a respeito do requisito correspondente. As respostas devem ser simples e diretas para evitar subjetividade na interpretação.
Modelo	Padrão de escrita do requisito, incluindo os parâmetros. Há dois tipos de parâmetros: obrigatórios, delimitados pelos símbolos < >, e opcionais, delimitados pelos símbolos [ ].
Exemplo	Exemplo do requisito com base no modelo estabelecido, incluindo valores reais para os parâmetros.
Obrigatoriedade	Indicador que determina se o requisito é exigido ou apenas desejável.
Dependência	Indica quais requisitos devem ser definidos para que sua completude seja alcançada.

Figura 9. Atributos dos modelos de requisitos.
Fonte: Silva (2016).



D.3 Usabilidade

D.3.1 Capacidade de Aprendizagem

Definição: Utilização por usuários específicos para alcançar objetivos estabelecidos de aprendizagem a fim de usar o produto ou sistema com eficácia, eficiência, inexistência de riscos e satisfação em um contexto de uso definido.

Questão 1 - O sistema deve disponibilizar algum mecanismo de ajuda?

Resposta SIM - requisito 01

Resposta NÃO - não se aplica

Requisito 01

Identificador 2.1.1.1

Modelo O sistema deve fornecer < um guia (wizard), um tutorial, ajuda contextual > para o usuário ao acessar < a(s) funcionalidade(s) x, y, z > [< pela primeira vez >] .

Exemplo O sistema deve fornecer um tutorial para o usuário ao acessar a funcionalidade de cadastro geral pela primeira vez.

Obrigatoriedade Desejável

Dependência Não há

Figura 10. Exemplo de definição de requisitos utilizando ADEG-NFR.
Fonte: Silva (2016).

2.1.4 Qualidade de Requisitos

O bom entendimento do funcionamento do negócio da organização durante a fase de Elicitação de Requisitos e a consistência do Documento de Requisitos impactam diretamente na qualidade do produto final. Logo, os requisitos documentados devem apresentar um conjunto de atributos que expressem a qualidade de sua especificação, assegurando que um projeto de software atinja seus objetivos de forma precisa e consistente (PRESSMAN, 2011). O Quadro 2 apresenta alguns desses atributos de qualidade que devem ser incorporadas à especificação de requisitos.

Quadro 2. Atributos de Qualidade na especificação de requisitos.

Atributo	Descrição
Completo	Cada requisito deve conter todas as informações necessárias para o leitor entender.
Correto	Cada requisito deve descrever com precisão uma capacidade que atenda às necessidades de algumas partes interessadas (*stakeholders*) e deve descrever claramente a funcionalidade a ser construída.

Fonte: Wiegers e Beatty (2013).

Quadro 2. Atributos de Qualidade na especificação de requisitos

Atributo	Descrição
Factível	Deve ser possível implementar cada requisito dentro dos recursos e limitações conhecidos do sistema e de seu ambiente operacional, bem como dentro das restrições de tempo, orçamento, e pessoal.
Necessário	Cada requisito deve descrever uma capacidade que forneça valor de negócio às partes interessadas.
Priorizado	Os requisitos devem ser priorizados de acordo com o que é mais importante para alcançar o desejado valor.
Sem ambiguidade	O requisito deve possuir uma única interpretação para o leitor.
Verificável	O requisito deve permirtir a realização de testes para verificar sua implementação.
Consistente	Requisitos consistentes não se confundem com outros requisitos do mesmo tipo ou com requisitos de negócio, usuário ou sistema.
Modificável	O requisito deve permitir alterações, mas sem grande impacto
Rastreável	Um requisito rastreável pode ser associado à sua origem e bem como com os demais requisitos, elementos de design, código que o implementa e testes que verificam sua implementação.

Fonte: Wiegers e Beatty (2013).

Bigio *et al.* (2017) relatam que a aferição da qualidade durante as fases da Engenharia de Requisitos é relevante, pois os erros podem ser detectados e corrigidos o quanto antes no ciclo de vida software, diminuindo os custos dessas correções. Como os requisitos incidem sobre todo o processo, a contínua mensuração afeta positivamente a qualidade do produto final.

2.1.5 Rastreabilidade de Requisitos

A rastreabilidade é um mecanismo para prover relacionamentos entre requisitos e artefatos produzidos no projeto, desde documentos até elementos implementados por meio de código, que possibilitem a compreensão de suas relações de interdependência. Fornece suporte para inúmeras tarefas de ES, como validação de requisitos, análise de impacto, análise de cobertura, verificação de conformidade, e análise de derivação (SAYÃO; LEITE, 2005; REMPEL; MADER, 2015).

A rastreabilidade de requisitos é definida como um pré-requisito para a obtenção de um produto final de qualidade. Além disso, no padrão IEEE 830 (1998) são recomendadas práticas de especificação de requisitos de software, bem como a norma ISO 29148 (2018) afirma que, na ER, os requisitos devem ser rastreáveis.

Segundo Sayão e Leite (2005), a rastreabilidade permite descrever e acompanhar a vida de um requisito, em todas as direções, seja a pré-rastreabilidade que a apresenta no contexto de surgimento do requisito ou pós-rastreabelidade que vincula os requisitos à modelagem ou desenho do sistema e à sua implementação, como apresentado na Figura 11.

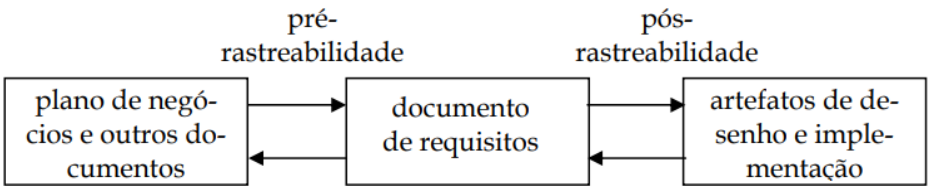


Figura 11. Rastreabilidade de Requisitos.
Fonte: Sayão e Leite (2005).

Uma das técnicas mais comuns de rastreabilidade é a referência cruzada de documentos e artefatos com os requisitos. Neste contexto, se destaca a matriz de rastreabilidade, onde em uma planilha são listados todos os requisitos e relacionados a todos os artefatos do projeto. A Figura 12 apresenta um modelo simplificado de matriz de rastreabilidade.

Projeto <nome_projeto> - Matriz de Rastreabilidade				
Requisito	Documento fonte	Arquitetura	Componente	Caso de teste

Figura 12. Modelo simplificado de matriz de rastreabilidade entre requisitos e demais artefatos de projeto.
Fonte: Sayão e Leite (2005).

Na primeira coluna são descritos os requisitos em linguagem natural e ordenados sequencialmente. Nas demais colunas, são preenchidos os artefatos gerados a partir desses requisitos ou que possuem algum tipo de relação. Essa prática auxilia no controle e análise dos impactos de mudanças em requisitos.

Além disso, Spanoudakis e Zisman (2005) destacam que a rastreabilidade pode ser uma ferramenta importante para análise em procedimentos de reúso de artefatos de sistemas de software, bem como para fins de comparação sobre necessidades em sistemas novos em relação aos já existentes.

2.2 Tesauro

A palavra tesauro se origina do grego *thesaurós* e significa tesouro ou repositório. Esse termo surgiu com a publicação do dicionário de Peter Mark Roget, em 1852, com seu conteúdo organizado de acordo com o significado dos termos, e não por ordem alfabética. Na década de 1960 e ampliando os estudos de Roget, Brian Campbell Vickery, cientista da informação, classificou o termo tesauro como um conjunto de palavras inter-relacionadas de acordo com o

seu significado, de forma que a partir de um único descritor, demais termos sinônimos podem ser localizados (MENDES *et al*., 2015; ALTOUNIAN; GOMES, 2016).

Sales e Café (2009) definem tesauro como um vocabulário controlado e organizado formado por termos que estão relacionados por meio de semântica e ordenados de forma pré-estabelecida. É utilizado principalmente como instrumento para a indexação e recuperação de documentos e informações através de seu conteúdo.

Para Mendes *et al.* (2015), um tesauro pode ser definido como uma lista estruturada de termos associados entre si de forma sintática e semântica, baseada na análise de conceitos, por meio de aspectos hierárquicos de generalização e especialização do significado dos termos. Basicamente, um tesauro é composto por uma taxonomia (hierarquia) de conceitos (termos), onde cada conceito é detalhadamente descrito. Esta hierarquia de conceitos é utilizada para classificar (indexar) artefatos de interesse, como livros, produtos, páginas Web etc

Um tesauro pode ser construído conforme sua especificidade, abordando grande número de conceitos gerais de uma área temática, como conceitos bem específicos, podendo ser multidisciplinares ou de uma única disciplina. Ainda podem ser monolíngues (escritos em um único idioma) ou multilíngues (possuindo versões em vários idiomas).

Sob o ponto de vista estrutural, um tesauro é um vocabulário organizado e dinâmico de conceitos, e termos descritores que os representam, formado por quatro elementos: (1) terminologia, composta por termos descritores preferidos e não preferidos; (2) estrutura gramatical, que determina a forma de apresentação e composição dos descritores; (3) rede paradigmática, que indica relações essenciais e estáveis entre os conceitos; e (4) rede sintagmática, que determina relações contingentes entre os descritores, válidas somente em determinado contexto de uso (MACULAN, 2015).

Maculan (2015) ainda relata que um tesauro tradicional apresenta terminologia própria, a saber:

- Descritor preferido (USE): um termo escolhido para representar o conceito de forma preferencial no tesauro, sendo utilizado para indexação e recuperação desse conteúdo.
- Descritor não preferido (UP – usado para): um termo que também pode descrever o conceito do tesauro, mas que não é indicado para a representação desse conceito em determinado contexto, para não criar uma grande variabilidade de sinônimos.
- Nota de escopo (NE): apresenta uma definição ou informação sobre determinado descritor.

- Termo genérico (TG): indica a existência de uma relação hierárquica entre conceitos, representando um conceito mais abrangente.
- Termo específico (TE): indica relação de subordinação entre um ou mais conceitos a um conceito mais genérico.
- Termo relacionado (TR): indica que há relações não hierárquicas entre conceitos ou não equivalência entre descritores, mas sim um tipo de relação de associação que orienta o usuário sobre a possibilidade de encadear esses conceitos em um eventual contexto.

Os termos de um tesauro podem ser interconectados por meio de relações de hierarquia (superordenação e subordinação), de equivalência (sinonímia) ou associação (MENDES *et al*., 2015; ALTOUNIAN; GOMES, 2016):

-Relações de Hierarquia: indicam a existência de uma relação de superordenação ou de subordinação. Isso acontece quando a relação entre dois termos indica que o significado de um sobrepõe sistematicamente o do outro. Ainda podem ser:

- Relações gênero/espécie (TG/TE): o termo superordenado representa o gênero e o termo subordinado especifica o gênero. Ex.: roupa (gênero) e camisa (espécie).
- Relações todo/parte: o termo superordenado por ser decomposto e vários outros termos que o compõem. Ex.: sistema solar (todo) e planetas, estrelas e asteroides (partes).

-Relações de Equivalência: são termos considerados sinônimos ou quase sinônimos para efeito de vocabulário. Em outras palavras, são diferentes termos que podem representar um único conceito. Ex.: nomes populares e científicos; nomes comuns e científicos e marcas comerciais; nomes instituídos cientificamente e os devidos à linguagem popular; palavras com ortografia diferente; termos de origem linguística diferente, termos originados de grupos étnicos diferentes, mas que usam uma língua comum; termos atuais e de menor uso; abreviaturas e nomes complexos.

-Relações de Associação: são termos/conceitos afins que devem estar relacionados, mas que não representam relações de hierarquia ou de equivalência. O Quadro 3 apresenta um conjunto de exemplos de relações associativas conforme a norma da *National Information Standards Organization* (NISO) Z39.19 (2005).

Quadro 3. Exemplos de relações associativas.

Relações Associativas	Exemplos
Causa / Efeito	acidente / lesão
Processo / Agente	medição de velocidade / velocímetro
Processo / Contra-agente	fogo / retardadores de chama
Ação / Produto	escrever / publicação
Ação / Propriedade	comunicação / habilidades de comunicação
Ação / Alvo	ensino / aluno
Conceito ou Objeto / Propriedade	liga de aço / resistência à corrosão
Conceito ou Objeto / Origem	água / poço
Conceito ou Objeto / Unidade ou Mecanismo de medida	cronômetro / minuto
Matéria-prima / Produto	uvas / vinho
Disciplina ou Campo / Objeto ou Praticante	Neonatologia / infantil

Fonte: NISO Z39.19 (2005).

Para exemplificar, um pequeno tesauro sobre esportes, utilizando as terminologias apresentadas:

- “Tênis”
    - TR “Quadra”
    - TG “Esporte”
- “Esporte”
    - TG “Atividade”
    - TE “Tênis”
    - TE “Futebol”
    - TE “Basquetebol”

No primeiro exemplo, apresentada uma relação hierárquica gênero/espécie entre “Tênis” e “Esporte”, utilizando a terminologia TG. O termo “Quadra” estabelece uma relação associativa por meio da terminologia TR. No segundo exemplo, é apresentada uma relação hierárquica todo/parte com o termo “Esporte” e os termos “Tênis”, “Futebol” e “Basquetebol”, por meio da terminologia TE que especifica as partes. Como “Atividade” é um termo genérico (TG) para “Esporte”, podemos dizer que “Tênis”, “Futebol” e “Basquetebol” também são atividades, devido a relação hierárquica.

A construção de tesauros é um procedimento sistematizado e bem definido. Segundo Maculan (2015), as metodologias para a construção de tesauros apresentam um esquema composto por três etapas: (I) inicial: planejamento geral do tesauro, incluindo delimitação do domínio a ser modelado e definição dos objetivos, tema, público-alvo e principais fontes de terminologia; (II) desenvolvimento: estrutura conceitual, critérios de modelagem, organização de descritores e estabelecimento de relações semânticas; e (III) edição: montagem da estrutura

de relações, escolha de forma de apresentação e simbologia das relações, e escolha de software para gerenciamento das etapas de construção do tesauro.

2.3 Tecnologias da Web Semântica

A concepção da Web Semântica surge a partir da necessidade de interação entre agentes de software com outros sistemas sem interposição dos seres humanos. Para que isso seja possível é necessário um conjunto de tecnologias e conceitos (SEGUNDO; CONEGLIAN, 2015).

Berners-Lee *et al.* (2001) definiram a Web Semântica como um projeto em que dados e informações dispostas para usuários em páginas web possam ser processadas automaticamente por máquinas. Dessa forma, esses dados e informações podem ser acessados e interpretados tanto por humanos quanto por máquinas simultaneamente.

O *World Wide Web Consortium* (W3C), conjunto de empresas, instituições acadêmicas e pesquisadores motivados por Tim Berners-Lee, se dedica ao desenvolvimento e padronização de novas tecnologias baseadas no ambiente web, se empenhando em integrá-las ao projeto Web Semântica (W3C, 2011).

Segundo Heath e Bizer (2011), para possibilitar essa distribuição de dados e interoperabilidade na web, é necessário um padrão de estruturação, especificação e interligação entre as instâncias, por meio da infraestrutura já disponibilizada pela web comum. A representação da semântica sobre o conteúdo compartilhada é dada por meio de metadadados, que descrevem os dados de forma que as máquinas possam compreendê-los de maneira inteligente.

A Figura 13 apresenta da artequitetura da Web Semântica apresentada por Tim Berners-Lee com apoio do 3WC. Essa arquitetura está em constante processo de evolução sendo essa a versão mais recente. As camadas de base facilitam a comunicação computador-computador com transmissão da informação de forma estruturada e padronizada. As camadas intermediárias auxiliam na estruturação e na interpreção semântica dos dados, por meio de mecanismos que atribuem significados sobre os mesmos (ISOTANI *et al.*, 2015).

Por fim, as camadas superiores realizam a interface humano-computador, onde por meio de aplicações e intefaces esses dados e informações podem ser consultados pelos usuários. O funcionamento dessas camadas são interdependentes, com a base oferecendo as tecnologias necessárias para o funcionamento das camadas superiores (ISOTANI *et al.*, 2015).

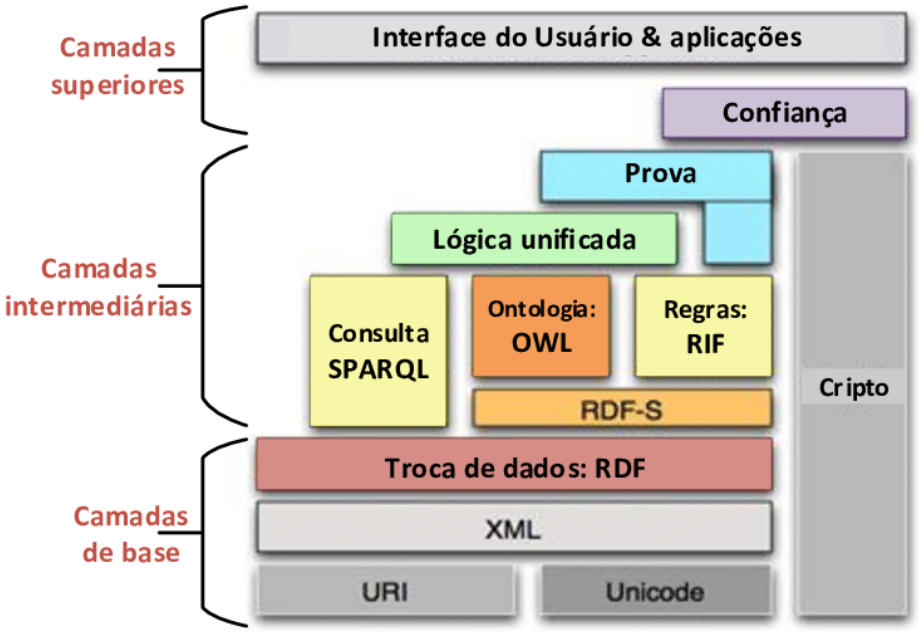


Figura 13. Arquitetura da Web Semântica.
Fonte: Isotani *et al.* (2015).

A seguir as tecnologias que compõem cada camada são apresentadas brevemente.

- **Unicode/URI:** é um padrão que permite a interoperabilidade em relação à codificação de caracteres na camada de base. Para que seja possível identificar e acessar um recurso disponibilizado na web, é utilizado o *Uniform Resource Identifier* (URI), que estabelece uma identidade única para aquele elemento (ISOTANI *et al.*, 2015; FREITAS JUNIOR; JACYNTO, 2016).

- **XML (*Extensible Markup Language)***: é uma liguagem de marcação que atua na representação sintática de recursos, independentemente da plataforma. Recomendada pelo W3C, atua na camada de base permitindo a integração dos dados, por meio da descrição das características inerentes aos mesmos (metadados). As XML Schemas são utilizadas para definir os vocabulários XML, detlhalhando a estrutura e o conteúdo dos documentos XML (FAWCETT *et al.*, 2012).

- **Padrão RDF:** para possibilitar essa distribuição de dados na web, é necessário um padrão de estruturação, especificação e interligação entre as instâncias, por meio da infraestrutura já disponibilizada pela web comum. Esse padrão é o *Resource Description Framework* (RDF), que se trata de um grafo que permite realizar declarações e estabelecer relacionamentos entre os dados disponíveis na web (HEATH; BIZER, 2011). A subseção 2.3.1 apresenta mais detalhes desta tecnologia.

- **Consulta SPARQL:** *SPARQL Protocol and RDF Query Language* é uma linguagem de

consulta e o protocolo de acesso para padrões RDF desenvolvida pelo W3C. A SPARQL permite manipulação de recursos representados em RDF para retornar informações a partir de uma consulta efetuada. A busca das triplas RDF pode ser feita a partir da filtragem de recursos, propriedades e valores (PRUD'HOMMEAUX; SEABORNE, 2008). A sintaxe da SPARQL é também baseada no conceito de triplas, assim como o modelo RDF. Com propósito similar à linguagem SQL, a estruturação da seleção e projeção dos dados consultados é baseado no conceito *Select-From-Where*.

- Ontologia (OWL): No contexto da informática e da ciência da informação, uma ontologia é definida como um modelo constituído de termos organizados de forma taxonômica, com definições e axiomas utilizados para representação formal de uma área de conhecimento definida. Esse conjunto de conceitos é dotado de classes, propriedades (atributos e relacionamentos), incluindo os significados e restrições. (DERMEVAL *et al.*, 2016).

- Regras RIF (*Rule Interchange Format*): é uma formato padronizado para a criação de regras de compartilhamento e interoperabilidade de aplicações baseadas na Web Semântica (ISOTANI *et al.*, 2015).

- Camada de Lógica Unificada: tem o intuito de aumentar, por meio da unificação de lógicas, a expressividade dos documentos descritos por meio de ontologias para que seja possível a capacidade de raciocínio em relação as informações obtidas das camada inferiores (ISOTANI *et al.*, 2015).

- Camada de Prova: permite a execução, verificação e validação dos recursos e mecanismos da camada de lógica, sob o ponto de vista semântico (ISOTANI *et al.*, 2015).

- Camada de Confiança: é responsável pela verificação da autenticidade e da confiabilidade dos recursos, dados e serviços (ISOTANI *et al.*, 2015).

- Camada de Criptografia: executa mecanismo que verificam a autoria, segurança e privacidade das informações, de forma integrada às camadas de base e intermediária (ISOTANI *et al.*, 2015).

- Camada de Interface e Aplicação: é camada responsável por realizar a interação de usuários e agentes de software com a aplicação da Web Semântica, fazendo uso do conhecimento formalizado por meio arquitetura (ISOTANI *et al.*, 2015).

2.3.1 Padrão RDF

O RDF fornece uma estrutura comum para expressar uma informação com o intuito de ser trocada entre aplicações sem perda de significado. Um recurso descrito em RDF é representado por meio de um grafo, também chamado de triplas, onde os vértices são

representados pelos conceitos sujeito, predicado e objeto. O sujeito é o recurso principal cuja a setença está referenciando. O predicado consiste em uma propriedado do sujeito, podendo ser uma característica ou relação. O objeto é o valor atrelado à propriedade em relação ao sujeito, podendo ser um valor literal ou outro recurso (MANOLA; MILLER, 2004). A Figura 14 apresenta essa estrutura de forma gráfica.

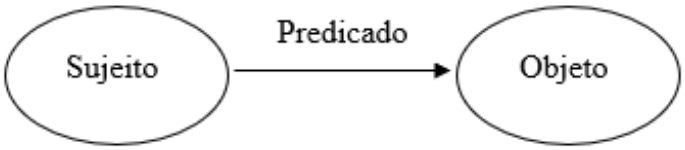


Figura 14. Representação genérica da tripla RDF.
Fonte: Elaboração Própria.

Segundo Manola e Miller (2004), essa estrutura permite que o RDF represente instruções simples sobre recursos como um gráfico de nós e arcos representando os recursos e suas propriedades e valores. A Figura 15 apresenta um exemplo realista de um recurso representado em RDF. Nesse caso, há uma pessoa identificada por *http://www.w3.org/People/EM/contact#me*, cujo nome é *Eric Mille*r, cujo endereço de e-mail é *em@w3.org*, e cujo título é *Doutor (Dr)*.

Figura 15. Exemplo realista de um recurso descrito em RDF.
Fonte: Manola e Miller (2004).

2.3.2 Ontologia

As ontologias representam um artefato de engenharia composto de um vocabulário específico, uma realidade e um conjunto de premissas sobre o significado pretendido para as palavras deste vocabulário. É considerada uma especificação explicíta de uma conceituação em

determinada área do conhecimento (HEFLIN, 2004). Segundo Agrisani (2006), para a construção de uma ontologia são necessários os seguintes componente:

- Classes: conceitos organizados de forma estruturada e informal, indicando hierarquias, generalizações e especializações em um domínio de conhecimento.
- Relações: interações entre conceitos de um determinado domínio, tais como propriedades semelhantes, dependências e associações.
- Axiomas: conjunto de premissas consideradas verdadeiras.
- Instâncias: as representações reais (dados) dos conceitos.

Ontologias são especificadas com linguagens padrão de criação de ontologias (metaontologias) que permitem algum tipo de abstração de estruturas de dados e de estratégias de implementação, tais como RDF Schema (RDFS) e *Web Ontology Language* (OWL) (HITZLER *et al.*, 2012; BRICKLEY *et al.*, 2014). A seguir é apresentada a ontologia SKOS, principal vocabulário utilizado para a construção dos tesauros semânticos propostos.

2.3.2.1 SKOS

SKOS – *Simple Knowledge Organization System* (Sistemas de Organização do Conhecimento Simples)[1] é um modelo padronizado para representação de informações e de conteúdo de Sistemas de Organização do Conhecimento, baseado nas estruturas de ontologias da Web Semântica. Baseada na linguagem OWL, o modelo SKOS é expresso de maneira estruturada com base no padrão RDF (SEGUNDO; CONEGLIAN, 2015).

Segundo Ramalho (2015), a Ontologia SKOS permite a representação de taxonomias, tesauros, esquemas de classificação, glossários e outros vocábulos controlados, para favorecer a reutilização e a interoperabilidade entre os vocábulos, identificados pode meio de URIs. Os vocábulos controlados podem ser categorizados de forma simples e/ou hierárquica em triplas RDF a partir dos seguintes elementos:

- Conceitos: podem ser definidos como uma invenção da mente humana para "encapsular" fragmentos da realidade. São os elementos-chave da estrutura e identificados por URI.
- Propriedades: têm a função de descrever os atributos dos Conceitos, a partir do objetivo do esquema conceitual. As propriedades podem ser de Etiquetagem ou de Documentação.
- Relações: têm a função de imputar a semântica aos esquemas a partir da categorização

[1] https://www.w3.org/2009/08/skos-reference/skos.html

e hierarquia dos conceitos. As relações podem ser Hierárquicas, Associativas ou de Equivalência.

O Quadro 4 apresenta alguns dos principais tipos de relações e propriedades da Ontologia SKOS. É possível observar as semelhanças com as terminologias e as relações dos tesauros apresentadas na subseção 2.2.

Quadro 4. Principais Tipos de Relações e Propriedades do SKOS.

Tipo	Propriedade	Definição
Relações Hierárquicas	*skos:broader*	Relaciona o conceito com um conceito mais geral.
	skos:narrower	Relaciona o conceito com um conceito mais específico.
Relações Associativas	*skos:related*	Define qualquer relação associativa não-hierárquica entre dois conceitos.
Relações de Equivalências entre conceitos de tesauros distintos	*skos:closeMatch*	Define relação de equivalência entre dois conceitos que podem ser considerados como similares em contexto previamente determinado.
	skos:exactMatch	Subpropriedade de *skos:closeMatch* que define relação de equivalência entre dois conceitos que possuem alto grau de correspondência e podem ser utilizados indistintamente em uma ampla gama de aplicações.
	skos:broadMatch	Subpropriedade de *skos:broader* que indica o relacionamento com um conceito mais geral.
	skos:narrowMatch	Subpropriedade de *skos:narrower* que que indica o relacionamento com um conceito mais específico.
	skos:relatedMatch	Subpropriedade de *skos:related* que indica um relacionamento associativo não-hierárquico com outro conceito.
Propriedades de Etiquetagem	*skos:prefLabel*	Subpropriedade de *rdfs:label* que define o termo (nome) preferencial de um conceito, em um determinado idioma.
	skos:altLabel	Subpropriedade de *rdfs:label* que define termos (nomes) alternativos de um conceito, como sinônimos, acrônimos, abreviações, variações de ortografia, e formas de plural/singular.
Propriedades de Documentação	*skos:note*	Nota geral sobre o conceito.
	skos:definition	Subpropriedade de *skos:note* que apresenta uma explicação formal completa do significado de um conceito.
	skos:example	Subpropriedade de *skos:note* que fornece exemplo de uso de um conceito.
	skos:scopeNote	Subpropriedade de *skos:note* que indica o escopo (contexto) onde o conceito pode ser usado.

Fonte: Miles e Bechhofer (2009); Ramalho (2015).

2.3.3 Linked Data

A prática de estabelecer relacionamentos entre os recursos disponíveis na web como o objetivo de promover o reuso dos mesmos é conhecido como *Liked Data* ou Dados Ligados. Marcondes (2012) ressalta que a proposta de interligar dados abertos por meio links semânticos

oferece grande potencial. Ao contrário dos links convencionais, que funcionam como uma etiqueta textual para humanos, os links semânticos carregam um significado que legível por computadores, enriquecendo o processamento de busca, interpretação e reutilização.

Berners-Lee *et al.* (2006) descrevem quatro tópicos para a publicação de dados ligados na web, conhecido como princípios *Liked Data*:

- Utilização de URIs para nomear recursos;
- Utilização de URIs HTTP (*Hypertext Transfer Protocol*) para que pessoas e máquinas possam acessar esses recursos na web;
- Assim que um URI for acessado, deve-se retornar algum tipo de informação útil a partir dos padrões RDF;
- Inclusão de links para acesso a outros URIs, para que pessoas e agentes de software possam navegar para outros recursos e obter outras informações.

Quanto mais as aplicações disponibilizam dados e informações a partir dos princípios *Liked Data*, maior interoperabilidade é adquirida para web, agilizando a identificação e reutilização dos recursos disponíveis. Ao seguir os padrões, esses dados podem ser inseridos no *Linked Open Data Cloud Diagram* ou LOD, um canal na web onde diagramas representam fontes de dados *Linked Data* (MCCRAE, 2019).

Dentre os conjuntos de dados presentes no LOD, destaca-se a DBpedia[2], uma versão *Linked Data* da conhecida Wikipedia[3]. Essa fonte de dados ocupa uma localização centralizada e um tamanho considerável no diagrama LOD, em função do grande reuso de seus dados por outras aplicações semânticas (*mashup* semântico). Todas esses dados podem ser acessados por com uso o uso da linguagem de consulta SPARQL.

2.4 Reúso de Software

Uma abordagem que tem como objetivo reutilizar partes do processo de construção de um software que já foi utilizado, assim como o código e outras etapas do processo, para o desenvolvimento de outros sistemas de software, de acordo com Ferreira e Naves (2011) é denominada Reúso de Software.

De acordo com Frakes e Kang (2005), o conceito de reúso de software significa reutilizar tanto conhecimento, quanto artefatos de software, e não se trata de um conceito novo, tendo sido criado por especialistas durante o desenvolvimento de software, ao mesmo tempo em que desenvolviam nova aplicação em contextos diferentes.

[2] https://wiki.dbpedia.org/
[3] https://www.wikipedia.org/

Um artefato de software é um resultado ou produto de conjunto de atividades envolvidas no processo de desenvolvimento de software, tais como documento de requisitos, diagrama de casos de uso, diagrama de classes, código-fonte, componentes, entre outros (PRESSMAN, 2011).

Segundo Pacheco *et al.* (2017), a reutilização de software tem como principal finalidade aumentar a produtividade e a qualidade, além de fornecer benefícios econômicos. Sommerville (2011) também destaca como benefícios a confiança aumentada, o risco de processo reduzido, conformidade com padrões e desenvolvimento acelerado.

Nesse cenário, o reúso de requisitos de software, no contexto de Reúso de Software destacou-se entre as outras etapas, pois fornece um suporte sólido para desenvolver um software de qualidade, partindo da etapa inicial do seu ciclo de vida (PACHECO *et al.*, 2017). Além disso, é um tópico importante de estudo em análise de domínios de sistemas.

O principal objetivo da análise de domínio é identificar, analisar e especificar requisitos comuns em um específico campo de aplicação para que sejam utilizados em demais projetos deste mesmo campo. Esses campos apresentam alta variabilidade de especificidades, podem ser da área de educação, médica, aviação, bancária, comercial, jogos, multimídia, marketing etc (PRESSMAN, 2011).

Um dos maiores desafios no reúso de artefatos de software está no gerenciamento eficiente dos artefatos com o intuito de prover localização e buscas rápidas. A recuperação dos componentes pode ocorrer através de repositórios e a partir de metadados, sem a necessidade de realizar buscas pelo seu conteúdo (SHIVA; SHALA, 2007).

Repositórios podem ser caracterizados como sistemas de bancos de dados (BD) que armazenam os artefatos e os seus metadados. Geralmente, um repositório é composto por uma arquitetura em camadas para facilitar o seu gerenciamento. Um exemplo é a arquitetura MOF (*Meta Object Facility*), dividida em três camadas. As camadas superiores são referentes à aplicação e às interfaces que gerenciam os objetos e seus relacionamentos, formas de recuperação e outras funcionalidades definidas. Na camada seguinte há o sistema do repositório, onde se encontram os metadados dos artefatos. Por fim, há a camada de persistência, que contém o BD responsável pelo armazenamento dos artefatos (PETROV; BUCHMANN, 2008).

Para facilitar o gerenciamento dos artefatos armazenados, eles podem ser categorizados e/ou classificados. Para isso, podem ser utilizados diferentes tipos de vocabulários, controlados ou não. Os métodos de classificação podem estar embasados em mais de um tipo de vocabulário e podem ser aplicadas técnicas de enumeração, facetas ou palavras-chave (VANDERLEI *et al.*, 2006). O Quadro 5 apresenta esses métodos com uma breve descrição.

Quadro 5. Tipos de métodos de classificação de artefatos de software.

Método	Descrição
Enumeração	Os artefatos são organizados classes hierárquicas, a partir de uma enumeração. Os métodos de indexação podem ser de forma manual ou automática. A enumeração define uma estrutura hierárquica em forma de árvore.
Facetas	Faceta é uma propriedade pré-definida de um artefato, que representa um par atributo-valor, onde os valores pré-definidos são associados aos atributos pelos usuários. O agrupamento de artefatos com uma mesma faceta gera as classificações.
Palavras-chave	As palavras-chave podem ser extraídas diretamente dos artefatos ou definidas por especialistas. As palavras-chave devem ser organizadas, conforme peso e frequência, para definir seu grau de importância e sinônimos devem ser definidos para facilitar a recuperação do conteúdo.

Fonte: Vanderlei *et al.* (2006).

2.5 Web Service RESTful

Os web services são um conjunto de funções genéricas de software que apresentam uma arquitetura que possibilita a comunicação entre diferentes aplicações, mesmo que em linguagens diferentes. É interoperável e seu acesso é feito através de protocolos e formatos de dados independentes de plataforma como o HTTP, XML e SOAP (*Simple Object Access Protocol*). Sendo assim, esses serviços possuem uma funcionalidade que pode ser reutilizada sem a preocupação de como é implementada (TAMAE; LIMA, 2005).

Para que o web service seja considerado RESTful, sua arquitetura deve ser desenvolvida conforme os princípios REST (**RE**presentational **S**tate **T**ransfer). Esse estilo ou padrão arquitetônico para sistemas hipermídia distribuídos foi apresentado na tese de Fielding (2000). O padrão REST apresenta 6 (seis) princípios ou restrições apresentados a seguir:

- Cliente-Servidor (*Client-Server*): Seguindo o modelo cliente-servidor, onde há a separação entre as funções da interface com o usuário e com o armazenamento de dados, há melhoria na portabilidade, uma vez que a interface pode interagir com diferente plataformas, além de permitir a escalabilidade da camada de armazenamento, contato que a conexão com a camada de interface não seja alterada.

- Sem estado (*Stateless*): Cada solicitação do cliente para o servidor deve conter todas as informações necessárias para o entendimento da mesma, não havendo nenhuma consulta ao contexto do servidor. Assim, o estado da sessão é mantido inteiramente no lado do cliente.

- Armazenável em cache (*Cacheable*): As restrições em relação ao cache devem exigir que os dados de uma resposta a uma solicitação possam ser classificados em armazenáveis ou não em cache para que possam ser reutilizados pelo cache do cliente em uma futura solicitação.

- Interface unifome (*Uniform interface*): Para obter uma interface uniforme são necessárias restrições na arquitetura que reflitam no comportamento dos componentes, com simplificação e melhoria na visibilidade das interações. O padrão REST é definido por quatros restrições na interface: identificação de recursos, manipulação de recursos através de representações, mensagens autodescritivas e hipermídia como motor do estado de aplicativo.

- Sistema em camadas (*Layered system*): A construção de uma arquitetura de sistema dividida em camadas permite estabelecer uma separação hierárquica entre os componentes, de modo que cada componente não "enxergue" além da camada com a qual está interagindo. As camadas ainda permitem que a implementação de políticas de segurança, encapsulamento e restrições que contribuem para a escalabilidade do sistema.

- Código sob demanda (*Code on demand*): como restrição opcional, o REST ainda permite que a funcionalidade do cliente seja estendida baixando e executando o código na forma de *applets* ou *scripts*, o que simplifica o cliente, com a redução do número de recursos para sua pré-implementação.

Muitos web services trabalham com requisições do protocolo HTTP, como é o caso do protótipo desenvolvido neste trabalho. O protocolo HTTP possui diversos métodos, cada um com uma semântica distinta, que indicam qual o tipo de manipulação será realizado em determinado recurso. Os mais utilizados são:

- GET: Obter os dados de um recurso;
- POST: Criar um novo recurso;
- PUT: Atualizar os dados de um determinado recurso;
- DELETE: Excluir um determinado recurso.

2.6 Considerações Finais

Ao longo deste capítulo, foram apresentados alguns conceitos considerados relevantes para o desenvolvimento da proposta deste trabalho. O conteúdo apresentado sobre ER foi sintetizado e restrito às necessidades do projeto. Em Tesauros, abordou-se os conceitos de caráter introdutório para dar suporte ao entendimento da aplicação de tesauros em SOCs.

Em relação às Tecnologias da Web Semântica, foram apresentados conceitos introdutórios sobre origem, motivações e funcionamento da Web Semântica. A busca nesta literatura se ateve à apresentação de tecnologias da Web Semântica que estão diretamente ligadas ao objetivo deste trabalho, ou seja, fornecem suporte à construção, compartilhamento e recuperação dos tesauros de requisitos.

Quanto ao tema de Reúso de Software, foram apresentados os principais benefícios do

reúso de artefatos de software em um projeto e como isso pode auxiliar na análise de um domínio. Por fim, foi apresentado um panorama conceitual sobre web services, a tecnologia escolhida para estruturar a base de conhecimento que armazena os tesauros semânticos.

Com este trabalho, espera-se que o uso de tecnologias da Web Semântica possa potencializar a atividade de reúso de requisitos com os tesauros semânticos, que permitem uma organização precisa e fidedigna por meio da formalização de conceitos ligados a essa área de conhecimento.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta uma seleção de trabalhos científicos que realizam interseções entre áreas temáticas que este trabalho aborda: Engenharia de Reguisitos, Tesauro, Tecnologias da Web Semântica e Reúso de Software. Em seguida, é realizada uma análise sobre os documentos que possam prover embasamento para o desenvolvimento da abordagem proposta neste trabalho.

3.1 Trabalhos selecionados

O Apêndice A apresenta os procedimentos de seleção dos trabalhos relacionados, com a metodologia de busca e os critérios de seleção, além de uma descrição detalhada de cada trabalho. O conjunto de trabalhos selecionados foi subdivido em três áreas temáticas. A seguir, é apresentada uma síntese do conteúdo de cada trabalho.

3.1.1 Área Temática: Tesauros na Engenharia de Requisitos

Nesta seção são apresentados alguns trabalhos utilizaram tesauros como ferramentas para a melhoria de algum processo ou atividade da ER.

Kaiya e Saeki (2005) propõem um método para análise de requisitos de software baseado em uma ontologia de domínio. O diferencial é que esse sistema ontológico consiste de um tesauro e regras de inferência. Uma parte do tesauro é responsável por conter os conceitos do domínio e relacionamentos específicos para o processamento adequado da ontologia. Por meio de um estudo de caso, os autores puderam comprovar que o tesauro contribuiu positivamente para a construção da ontologia de domínio, tornando mais precisos os processamentos semânticos.

Ninaus et al. (2014) apresentam a IntelliReq, uma ferramenta baseada em técnicas de inteligência artifical para apoiar o reuso de requisitos. Para isso, a IntelliReq auxilia na aferição da qualidade dos requistos por meio de processamento de linguagem natural, com auxílio de um tesauro de mesmo domínio do conjunto de requisitos. Com isso, é possível detectar eventuais ambiguidades ou incompletudes no significado dos requisitos. Para testar a eficácia da ferramenta , foi realizado um estudo de caso com dois grupos de estudantes, onde ambos deveriam escolher dentre um conjunto de requisitos, quais poderiam ser reutilizados, só que somente um trabalharia com a ferramenta. Como resultado, observou-se que o grupo que utilizou a ferramenta conseguiu identificar cerca de 20% a mais de possíveis requisitos reutilizáveis.

3.1.2 Área Temática: Tecnologias da Web Semântica na Engenharia de Requisitos

Nesta seção são apresentadas algumas aplicações de tecnologias da Web Semântica na ER, exemplificando como elas podem apoiar as atividades de elicitação e de especificação de requisitos por meio da organização de conhecimento e de mecanismos para recuperação da informação para reúso.

Farfeleder *et al.* (2011) apresentam a implementação de um protótipo de sistema orientado por ontologias para auxiliar profissionais a obter requisitos usando uma representação semi-formal. O protótipo tem como base o conceito de boilerplate, um template para requisito em que o usuário substitui certas palavras-chave pelos elementos que identificam o sistema que está a construir. Ex.: *The <stakeholder> shall be able to <capability>*. A ferramenta conseguiu fornecer sugestões úteis na maioria dos casos, cerca de 85%.

Motta *et al.* (2017) propõem um modelo para redução do impacto de problemas existentes na comunicação entre stakeholders durante a elicitação de requisitos, gerando um impacto nos atributos de qualidade relacionados. Este modelo de ERC (Elicitação de Requisitos em Ciclos) aplica o método MAIA e cria uma hierarquia de conceitos que auxiliam na análise da consistência e completude da informação da arquitetura de informação dos requisitos, permintindo a rastreabilidade. A mesma estrutura pode ser aplicada para requisitos funcionais ou não funcionais. A proposta de compartilhamento de informações estruturadas em triplas ontológicas se mostrou viável e útil.

Nardi e Falbo (2006) apresentam o desenvolvimento de uma Ontologia de Requisitos de Software para a formalizar o conhecimento referente a um domínio de requisitos. A ontologia foi elaborada com a utilização do método SABiO. Para definição e taxonomia de requisitos, são definidos os axiomas que estabelecem as relações. Um dicionário foi elaborado para as definições dos conceitos que compõem a taxonomia. Alguns atributos que constituem o dicionário: requisito, documento de especificação de requisitos de software, tipo de requisito, módulo, escopo e contexto.

3.1.3 Área Temática: Repositórios para Reúso de Componentes de Software

Nesta seção são apresentados alguns trabalhos que descrevem repositórios ou bibliotecas de reúso de componentes de software.

CHANG *et al.*, (2011) apresentam uma biblioteca de componentes reutilizáveis baseada em XML para software incorporado. Os componentes considerados para reutilização no trabalho foram: controle de gestão, diagrama UML, diagrama de Estrutura, código e linguagem de programação, além de restrições do sistema operacional. O mecanismo de busca utilizado

na RC editor é realizado através de duas formas, a primeira é realizada uma consulta em um banco de dados nativo do XML, assim através de palavras-chaves, o usuário faz a busca. Além disso, os próprios usuários podem adicionar componentes a Biblioteca, para que o mesmo seja acessado posteriormente por outro usuário. Não são mencionado mecanismos de aferição de qualidade.

De acordo com Santos *et al.* (2013), a Biblioteca Brechó é um sistema de informação Web com uma base de aplicações, componentes, serviços, produtores/consumidores, com mecanismos de armazenamento, documentação, publicação, busca e recuperação. Considera como componente, os artefatos produzidos no desenvolvimento (processo, modelos, manuais, código, binário, testes etc.). O mecanismo de busca utilizado é baseado em alguns requisitos elencados, com o objetivo de oferecer opções de busca mais flexíveis e automatizadas, e assim, facilitar a pesquisa de componentes na biblioteca, aumentando a precisão dos resultados na busca. O mecanismo de avaliação dos componentes da Biblioteca é composto por dois módulos (uma para a avaliação dos usuários e disponibilazação dos percenctual de avaliação). Além disso, há mecanismos relacionados a categorização dos componentes.

Esta ferramenta é uma ferramenta proposto por Wang *et al.* (2016) e tem como objetivo a reutilização do código fonte da linguagem JAVA, ademais a Biblioteca conta com a funcionalidade do alinhamento de interface e síntese do código reutilizável. Como mecanismo de busca, os autores utilizaram um hibridismo de métodos, no qual integraram o mecanismo do código Pliny além da utilização de um BD com métodos Java coletados de repositórios de código aberto. Não foi mencionado nenhum mecanismo de aferição de qualidade dos códigos reutilizáveis.

De acordo com Abid *et al.* (2017), CodeEase é uma ferramenta que tem como principal objetivo a reutilização de código fonte, e foi desenvolvida como um *plug in* para o Eclipse. O mecanismo de busca está associado ao Eclipse, assim quando um usuário escreve ou edita linhas de código em um método, o CodeEase fornece recomendações de conclusão para o mesmo (*Reconmedation Engine)*, baseado na reutilização de código. Essa busca é realizada através da consulta em repositórios de código aberto, e um repositório local, o qual foi pré-populado com uma grande gama de códigos. Não foi mencionado nenhum mecanismo de aferição de qualidade dos códigos reutilizáveis.

3.2 Considerações Finais

Partindo do objetivo geral deste trabalho, que é uma proposta de documentação de requisitos baseada em tesauros semânticos para promover seu reúso em outros projetos em um

domínio específico, o intuito é que as pesquisas relatadas nos trabalhos elencados possam contribuir com métodos e abordagens que possam ser aplicados nesta pesquisa. A análise foi realizada de acordo com a área temática.

-A1: Tesauros na Engenharia de Requisitos

Em relação ao uso de tesauros na ER, os trabalhos de Kaiya e Saeki (2005) e Ninaus *et al.* (2014) demonstram aspectos bastante positivos ao estruturar e organizar requisitos dessa forma. A estrutura do tesauro apresentado por Kaiya e Saeki (2005) na Figura 41 foi um elemento primordial para que a ontologia de domínio fosse construída de forma correta para a realização das inferências.

Ninaus *et al.* (2014) também destacam que o uso de tesauro auxilia tanto na documentação de termos de significado dentro um domínio quanto na implementação de mecanismo para a aferição de qualidade em requisitos escritos em linguagem natural de forma automatizada. Apesar de nenhum desses trabalhos ter sido implementado na web semântica, os mesmos trazem diretrizes que podem auxiliar na construção dos tesauros deste trabalho, principalmente por exemplicar a questão do mapeamento das informações.

Altounian e Gomes (2016) destacam que o uso de tesauros pode melhorar os resultados de buscas na web. Como, neste caso, as informações serão descritas na web semântica por meio de metadados, a combinação da metodologia de criação de tesauros com os princípios *Linked Data* pode potencializar a precisão e a inteligência dos mecanismos de recuperação dessa informação, além de permitir a sua extensão quando necessário.

-A2: Tecnologias da Web Semântica na Engenharia de Requisitos

A proposta apresentada por Farfeleder *et al.* (2011) demonstra como o mapemaneto correto da estrutura descritiva de um requisito potencializa o poder de inferência da ontologia. Apesar do mapeamento estruturado que será prosposto neste trabalho não focar na sentença de descrição do requisito, o conceito de *boilerplate* de Farfeleder *et al.* (2011) corrobora com a necessidade da existência de um padrão descritivo para facilitar a atividade de reúso, uma vez que a descrição é baseada no modelo de Alexander e Stevens (2002, *apud* WIEGERS; BEATTY, 2013), que também é uma das bases para este trabalho.

A estrutura apresentada por Motta *et al.* (2017) facilita o mapeamento das informações de RFs e RNFs para triplas ontológicas. É criada uma taxonomia simples e clara, onde é possível verificar aspectos de “subtipo”, um dos pontos principais para a aplicação do conceito de termos genérico-específico na construção de tesauros. Em conjunto com as diretrizes para a definição de uma taxonomia de requisitos apresentadas por Nardi e Falbo (2006), com destaque para o

dicionário proposto que elenca informações importante para auxiliar na interpretação de um requisito, os dois trabalhos contribuem com pontos que serão considerados ao realizar o mapeamento das informações dos requisitos para criar os tesauros neste trabalho.

-A3: Repositórios para Reúso de Componentes de Software

Um dos objetivos da proposta deste trabalho é criar um repositório para que os tesauros sejam armazenados e consultados. Tendo isso vista, foram analisadas as características de alguns repositórios e bibliotecas de reúso a fim de investigar a estrutura e os mecanismos de buscas e como elas se relacionam com os aspectos da qualidade dos componentes armazenados. O Quadro 6 apresenta um paralelo entre as funcionalidade dos repositórios.

Quadro 6. Paralelo entre as funcionalidades dos repositórios de reúso.

Ferramenta	Mecanismo de Busca	Aferição de Qualidade	Componentes
RCR Editor	- Palavra Chave; **Local da Busca:** - BD nativo XML.	Não foi abordado no artigo.	- Diagrama UML; - Diagrama de Estrutura; - Código Fonte; - Linguagem de programação.
Brechó	- Palavra Chave; - Por categoria; - Busca Sintática. **Local da Busca:** BD Local.	- Feedback do usuário; - Graficamente.	- Processos - Modelos; - Manuais; - Código Fonte; - Binário.
Hunter	- Palavra Chave; **Local da Busca:** - BD: GitHub e Bitbucket .	Não foi abordado no artigo.	- Código Fonte.
CodeEase	- Complemento de Palavras; **Local da Busca:** BD: Local e de Código Aberto;	Não foi abordado no artigo.	- Código Fonte.

Fonte: Elaboração Própria.

É possível constatar que a principal forma de armazenamento são os bancos de dados comuns. Muitos deles abragem diversos componentes de sofware para reutilização. A Brechó é a que possui diferentes mecanismos de buscas e um método para a aferição de qualidade dos componentes. No entanto, nenhum deles é desenvolvido para funcionar em ambiente semântico ou se utiliza de tesauros para a organização de seu conteúdo.

Em comparação às características apresentadas, é possível destacar que uma das

principais diferenças do repósitorio de tesauros deste trabalho para os demais é que o conteúdo é focado em requisitos e desenvolvido dentro de um ambiente semântico. É utilizado um banco orientado a grafos e o mecanismo inicial de buscas é por meio da URI de um recurso. É dito inicial pois, como o repositório é construído para funcionar como um web service, o seu conteúdo poderá ser acessado por diferentes aplicações a partir de um *endpoint*, permitindo que as mesmas possam manipular os dados de diversas maneiras. Além disso, há a intenção de disponibilizar o código fonte para que o projeto possa ser melhorado e expandido pela comunidade. Um dos principais desafios está na garantia da aferição da qualidade dos requisitos, onde a biblioteca Brechó poderá ser uma base importante essa pesquisa.

Assim, espera-se contribuir para a possibilidade de implementação de mecanismos de consultas inteligentes que proporcionem sugestões de requisitos candidatos ao reúso de forma eficaz e eficiente para a elaboração de documento de requisitos de um novo projeto.

.

4 MÉTODO PARA CRIAÇÃO DE TESAUROS SEMÂNTICOS DE REQUISITOS

Este capítulo descreve a proposta desse trabalho, iniciando pela definição do modelo (*template*) de especificação de requisitos que será utilizado. Em seguida, são apresentados os procedimentos metodológicos para a construção dos tesauros.

4.1 Apresentação do modelo proposto para especificação Requisitos no contexto semântico

Na literatura, existem vários tipos de mecanismos utilizados para auxiliar no processo de elicitação de requisitos, tais como: catálogos de tipos de requisitos, questões/*checklist* e *templates*. Para documentar os requisitos que constituirão os tesauros, foi escolhido o método de especificação estruturada (*template*), onde há uma relação de atributos que devem ser catalogados para que haja a completude das informações relacionadas a um requisito.

A proposta de um modelo para especificação de requisitos no contexto semântico tem como finalidade reunir um grande número de atributos relevantes para descrever um requisito e, após o mapeamento com auxílio de ontologias (procedimento que será descrito ao apresentar o método), proporcionar mais subsídios que facilitem a recuperação (tais como a pesquisa por domínios ou por tipo de requisito) e a inferência da máquina sobre essas informações.

Considerando o modo de inserção manual por parte dos usuários do repositório e a quantidade de informações atreladas aos requisitos que compõem os tesauros, foi decidida a criação de um único *template* capaz de ser a interface para captar as informações tanto de RFs como de RNFs. Os atributos que compõem esse *template* são um híbrido de conceitos encontrados na literatura, nos trabalhos relacionados e nos modelos descritos na seção 2.1.3. A Figura 16 apresenta, em síntese, as bases do modelo proposto e, em seguida, são apresentadas as justificativas para suas escolhas, bem como quais atributos advém dos mesmos.

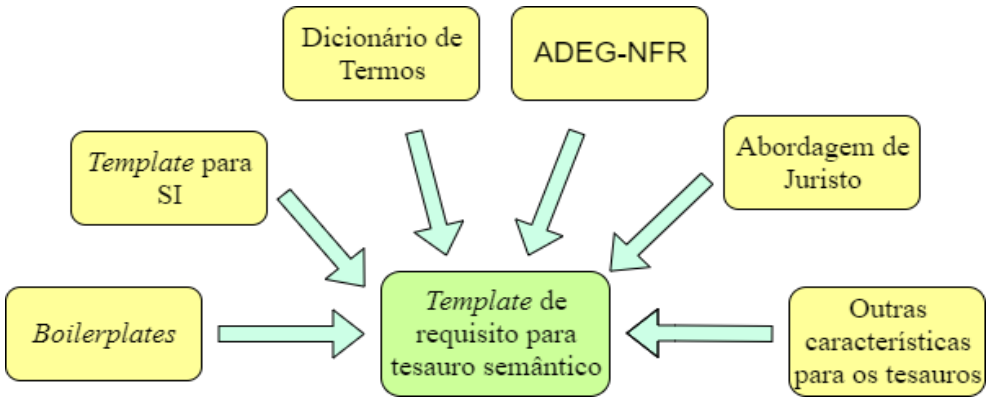


Figura 16. Síntese dos modelos/padrões utilizados na definiçao do *template* de requisito proposto.
Fonte: Elaboração Própria.

- Uso de *templates* ou *boilerplates* para a escrita dos requisitos: Alexander e Stevens (2002, *apud* WIEGERS; BEATTY, 2013) e Farfeleder *et al.* (2011) trazem abordagens que se complementam para padronizar a escrita de um requisito, criando possibilidades de reúso mais eficientes. Ambos possuem um número de citações considerável, sendo o primeiro em 305 artigos e o segundo em 62 artigos, cujos periódicos se encontram em bases científicas renomadas como *Scopus*® e IEEE.
Contribuição para o *template*: Padrão de especificação escrita para o atributo Modelo.

- Padrões para especificação de requisitos de SI: O *template* apresentado por Barcelos (2016) é produto de sua dissertação de mestrado, onde foi feita uma ampla pesquisa sobre os padrões mais utilizados em SI, sendo verificados artigos de bases científicas, como *Scopus*® e IEEE.
Contribuição para o *template*: Atributos "Nome", "Domínio", "Problema", "Tipo de Requisito", "Modelo" e "Requisitos relacionados".

- Dicionário de Termos: Nardi e Falbo (2006) apresentam um dicionário com os principais termos que se relacionam a especificação de requisitos para estruturar sua ontologia. Diante dos resultados obtidos no trabalho em relação à inferência da ontologia, isso reforça a importância da presença de alguns atributos já identificados em outros modelos.
Contribuição para o *template*: Atributos "Tipo de Requisito" e "Contexto".

- ADEG-NFR: O *template* proposto por Silva *et al.* (2016) é parte de uma tese de doutorado recente, onde foi feita uma ampla pesquisa sobre os RNFs, sendo encontrados artigos provenientes de bases confiáveis, como por exemplo*: Scopus*® e IEEE, além de outros eventos de renome na área de ES.
Contribuição para o *template*: Atributos "Identificador", "Tipo de Requisito", "Modelo" e "Exemplo".

- Abordagem de Juristo: O *template* proposto por Juristo *et al.* (2007) se mostrou um modelo intuitivo e de fácil entendimento. Além disso, o artigo mencionado possui uma grande relevância na área de RNFs, sendo encontrado em bases sólidas, como *Scopus*® e IEEE, além de ter sido citado em outros 48 artigos de periódicos, o que corrobora a importância do mesmo para a literatura.
Contribuição para o *template*: Atributos "Problema" e "Contexto".

- Outras características para os tesauros: Com intuito de agregar mais informações aos requisitos, esta proposta sugere o acréscimo dos atributos "Nome Preferencial" e "Nome

Alternativo", para agregar novas possibilidades de identificação do requisito. A finalidade é que se o requisito for identificado pelo nome alternativo, que seja utilizado o nome preferencial durante o reúso. Outro atributo é "Tipo de Sistema", para que os requisitos também possam ser vinculados a um determinado tipo de sistema. Por fim, o atributo "Idioma" tem a finalidade de classificar os requisitos conforme a língua em que são escritos. O conceito "genérico-específico", muito característico dos tesauros, é representado por meios atributos "Requisito Generalista" e "Requisitos Específicos" para criar a relação hierárquica entre os requisitos.

O Quadro 7 apresenta os atributos consolidados para o *template* proposto de especificação de requisito no contexto semântico, com a descrição do conteúdo de cada atributo e sua origem de acordo com a abordagem apresentada ao longo desta seção em relação aos modelos de especificação estruturada de requisitos.

Quadro 7. Modelo (*template)* proposto para especificação de requisito no contexto semântico.

Campo	Descrição	Origem
Identificador	Descrição da URI.	ADEG-NFR
Nome	Nome do requisito, que possibilite entender a sua função apenas pelo nome.	*Template* para SI.
Nome Preferencial	Nome preferencial para indentificar o requisito.	Sugerido por esta proposta.
Nome Alternativo	Um outro que pode ser utilizado para ser referir ao mesmo requisito.	Sugerido por esta proposta.
Idioma	Define o idioma em que o requisito está escrito.	Sugerido por esta proposta.
Problema	Quais informações precisam ser elicitadas e especificadas para que o aplicativo forneça informações de status aos usuários, ou seja, situação que deu origem a necessidade do requisito.	*Template* para SI e Abordagem de Juristo.
Contexto	Quando ocorrem mudanças importantes para o usuário e quando isso é importante para o usuário. Contexto utilizado para a utilização do requisito.	Dicionário de Termos e Abordagem de Juristo.
Modelo	Padrão (*boilerplate*) de escrita do requisito, incluindo os parâmetros. Há dois tipos de parâmetros: obrigatórios, delimitados pelos símbolos < >, e opcionais, delimitados pelos símbolos [ ]. Ex.: O [tipo de usuário ou nome do ator] deve ser capaz de [realizar algo] [ao objeto] [sob alguma condição].	*Boilerplates, Template* para SI e ADEG-NFR.
Exemplo	Demonstração do uso do modelo.	ADEG-NFR.

Fonte: Elaboração Própria.

Quadro 7. Modelo (*template)* proposto para especificação de requisito no contexto semântico.

Campo	Descrição	Origem
Domínio(s)	Relação de domínios aos quais o requisito está vinculado.	*Template* para SI.
Tipo(s) de Sistema	Indica qual os tipos de sistema (Informações Gerenciais, Processamento de Transações, Apoio a Decisão, etc) que o requisito está relacionado.	Sugerido por esta proposta.
Tipo de Requisito	Indica se o requisito é classificado como funcional ou como uma das subcategorias de não funcional (conforme ISO/IEC-25010).	*Template* para SI, Dicionário de Termos e ADEG-NFR.
Requisito Generalista	Indica o requisito que está acima deste no tesauro.	*Template* para SI, adaptado por esta proposta.
Requisitos Específicios	Indica a relação de requisitos que estão abaixo deste no tesauro.	*Template* para SI, adaptado por esta proposta.

Fonte: Elaboração Própria.

A especificação de requisitos de forma estruturada facilita geração de triplas RDF, criando uma estrutura similiar à apresentada no trabalho de Motta *et al.* (2017), que será armazenada em um BD orientado a grafos.

4.2 Apresentação do método

Neste trabalho, propõe-se a criação de tesauros semânticos de requisitos com o intuito de incorporar os princípios da Web Semântica e *Linked Data* para otimização dos procedimentos de organização do conhecimento e de disponibilização e recuperação de informações. Para isso, é necessário criar um ambiente conforme o modelo de integração de dados de Berners-Lee apresentado na Figura 17.

Nesse modelo, um conjunto de dados não estruturados, semi-estruturados e estruturados são trazidos para o formato RDF pode meio de uma camada de conversão e são armazenados em uma base de mesmo formato. Esses dados convertidos podem ser acessados por meio de consultas SPARQL e Ontologias que alimentam uma série de aplicações de natureza semântica. Em conjunto com esse ambiente, será executada a seguinte metodologia elaborada com base no plano de construção de tesauros de Maculan (2015), conforme o Quadro 8.

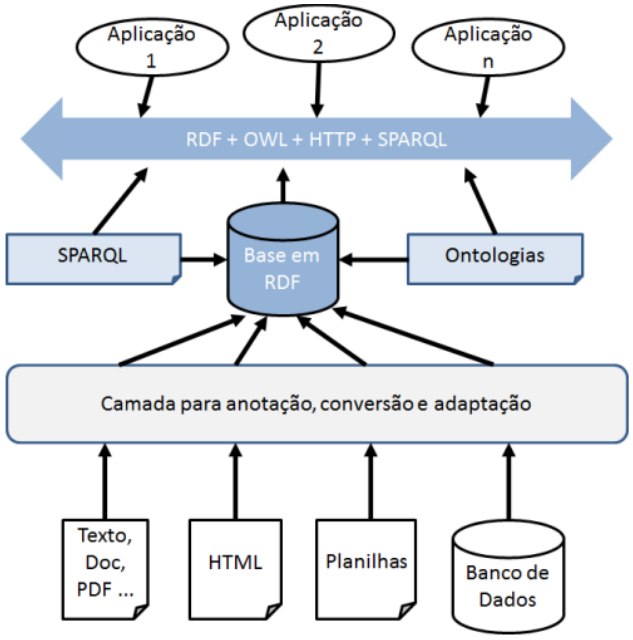


Figura 17. Modelo adaptado de integração de dados de Berners-Lee.
Fonte: Cruz (2015).

Quadro 8. Principais atividades de cada etapa metodológica.

Etapas	Atividades
(1) Planejamento Geral do Tesauro	1- Elaboração de tanoxomias de conceitos de ER a partir do *template*. 2- Definição formato de dados de entrada para cada campo do *template* (arquivo, texto, recuperação de conceitos já armazenados, etc).
(2) Estruturação do Tesauro	1- Definição das propriedades SKOS que serão utilizadas para relacionar cada atributo (campo do *template*) ao conceito principal do requisito.
(3) Edição e extensão do tesauro	1- Realização de consultas SPARQL para avaliar a conformidade das relações entre os conceitos armazenados, verificando a rastreabilidade dos dados. 2- Disponibilização do repositório em RDF, para promover reúso e extensão.

Fonte: Elaboração Própria.

É importante ressaltar que esta metodologia pode ser aplicada a qualquer modelo de especificação de requisitos, conforme a necessidade do projeto de SI.

Etapa 1: Planejamento Geral do Tesauro

Grande parte dos procedimentos ocorre na camada de conversão do modelo de integração de dados de Berners-Lee. Inicialmente, é necessária a escolha de um modelo de especificação de requisitos, como o proposto no Quadro 7, que será convertido em um formulário. Em seguida, deve ser definido o formato de entrada dos campos do formulário, que podem ser arquivo, texto ou recuperação de conceitos já armazenados. Se as opções de um

campo puderem ser hierarquizadas ou taxonomizadas, um novo tesauro ou subtesauro pode ser criado para facilitar a organização e recuperação dos recursos.

A ação de recuperar um recurso já armazenado por um tesauro aumenta o número de conexões entre os dados e, consequentemente, os mecanismos de recuperação. Um exemplo é a conexão com fontes de dados externas, como a DBpedia, seguindos os princípios *Linked Data* para estabelecer uma web de dados interligados por meio de URIs HTTP, além de permitir a checagem da rastreabilidade entre essas informações.

Etapa 2: Estruturação do Tesauro

A estrutura dos tesauros semânticos é construída com base nas relações e propriedades fornecidas por ontologias, sobretudo, a ontologia SKOS. Cada requisito se torna um conceito SKOS. A seguir, devem ser estabelecidas as relações hierárquicas e associativas entre esses conceitos. O Quadro 9 apresenta os principais tipos de relações e propriedades do vocabulário SKOS, que podem ser empregadas nesta etapa de desenvolvimento.

Quadro 9. Principais Tipos de Relações e Propriedades do SKOS.

Tipo	Propriedade	Definição
Relações Hierárquicas	*skos:broader*	Relaciona o conceito com um conceito mais geral.
	skos:narrower	Relaciona o conceito com um conceito mais específico.
Relações Associativas	*skos:related*	Define qualquer relação associativa não-hierárquica entre dois conceitos.
Relações de Equivalências entre conceitos de tesauros distintos	*skos:closeMatch*	Define relação de equivalência entre dois conceitos que podem ser considerados como similares em contexto previamente determinado.
	skos:exactMatch	Subpropriedade de *skos:closeMatch* que define relação de equivalência entre dois conceitos que possuem alto grau de correspondência e podem ser utilizados indistintamente em uma ampla gama de aplicações.
	skos:broadMatch	Subpropriedade de *skos:broader* que indica o relacionamento com um conceito mais geral.
	skos:narrowMatch	Subpropriedade de *skos:narrower* que que indica o relacionamento com um conceito mais específico.
	skos:relatedMatch	Subpropriedade de *skos:related* que indica um relacionamento associativo não-hierárquico com outro conceito.
Propriedades de Etiquetagem	*skos:prefLabel*	Subpropriedade de *rdfs:label* que define o termo (nome) preferencial de um conceito, em um determinado idioma.
	skos:altLabel	Subpropriedade de *rdfs:label* que define termos (nomes) alternativos de um conceito, como sinônimos, acrônimos, abreviações, variações de ortografia, e formas de plural/singular.

Fonte: Miles e Bechhofer (2009); Ramalho (2015).

Quadro 9. Principais Tipos de Relações e Propriedades do SKOS.

Tipo	Propriedade	Definição
Propriedades de Documentação	*skos:note*	Nota geral sobre o conceito.
	skos:definition	Subpropriedade de *skos:note* que apresenta uma explicação formal completa do significado de um conceito.
	skos:example	Subpropriedade de *skos:note* que fornece exemplo de uso de um conceito.
	skos:scopeNote	Subpropriedade de *skos:note* que indica o escopo (contexto) onde o conceito pode ser usado.

Fonte: Miles e Bechhofer (2009); Ramalho (2015).

O uso dessas relações e propriedades SKOS deve ser planejado de acordo com semântica que se deseja estabelecer entre dois conceitos, a partir do pensamento em triplas RDF (recurso-propriedade-valor).

Nesse planejamento devem ser considerados também os princípios *Linked Data*, gerando informações úteis por meio das relações, assim como com a criação de URIs únicos para cada conceito, sendo esses diretamente ligados a URIs de outros conceitos, permitindo a navegabilidade na estrutura do tesauro.

Etapa 3: Edição e extensão do tesauro

As triplas RDF geradas são armazenadas em uma base de conhecimento semântica de requisitos, um banco de dados RDF (*triple store*). Por padrão, esse banco oferece um SPARQL *endpoint*, ou seja, um web service que permite executar consultas SPARQL *ad-hoc* sobre o tesauro, obtendo precisamente a informação que se queira, como, por exemplo, na questão de rastreabilidade de requisitos. Além disso, seguindo o terceiro princípio *Linked Data*, os URIs dos requisitos são dereferenciáveis, ou seja, ao serem acessados retornam um arquivo descrevendo o requisito. Dependendo do tipo de acesso é retornado um arquivo RDF para máquinas ou um arquivo HTML para humanos. Com relação ao quarto princípio *Linked Data*, sempre que possível devemos enriquecer a descrição dos requisitos, os relacionando com recursos (URIs) de outras fontes de dados consagradas da Web Semântica (p.ex. DBpedia), perfazendo o chamado *Linked Data manshup*.

A base de conhecimento é extensível, o que permite que novos conceitos, relações ou propriedades sejam inseridos ou modificados a qualquer momento. Conforme a essência da Web, para referenciar um requisito a partir de qualquer lugar do mundo, seja para estendê-lo ou para (re)usá-lo, basta estabelecer um link com seu respectivo URI, estabelecendo uma tripla RDF.

Uma aplicação com interface de usuário amigável (ou um *plug-in* para uma ferramenta

existente) deve ser criada para dar suporte ao cadastro e definição das relações entre os requisitos, de modo que o grafo RDF subjacente seja automaticamente criado e armazenado no banco de dados. Ao inserir um novo conceito de requisito, com base nos axiomas da ontologia SKOS, o módulo raciocinador (*reasoner*) do banco RDF infere automaticamente mais informações, contribuindo para que o tesauro se torne cada vez mais robusto e completo.

4.3 Considerações Finais

Este capítulo apresentou a proposta de documentação de requisitos por meio da criação de tesauros semânticos. O *template* proposto é constituído de um híbrido de modelos existentes na literatura. O principal objetivo é agregar ao contexto semântico os atributos já utilizados nesses modelos, que passaram por processos de validação e apresentam uma estrutura que contém elementos importantes para a documentação de requisitos.

Outras características para atributos do *template* são sugeridas nesta proposta de modo que facilite a criação de relacionamentos entre requisitos, principalmente de generalização e especificação. Com isso, é possível construir a estrutura de tesauros no universo dos requisitos e proporcionar a navegabilidade entre os atributos e as informações interrelacionados. Dessa forma, os requisitos podem ser recuperados e interpretados corretamente para o reúso.

Com o *template* definido, são definidas as diretrizes metodógicas baseada no plano de Maculan (2015), onde por meio de três etapas são descritas as principais atividades para construir as triplas RDF, com o auxílio da ontologia SKOS, e compartilhar os tesauros semânticos. O próximo capítulo apresenta a aplicação desse método para o desenvolvimento do protótipo, demonstrando o mapeamento do modelo de especificação de requisitos proposto e exemplificando com os tesauros ilustrados em forma de grafos.

5 PROTÓTIPO DO WEB SERVICE RESTFUL

Este capítulo apresenta um protótipo desenvolvido para avaliar o método proposto para criação de tesauros. Primeiramente, é demonstrada a aplicação do método com o planejamento e a estruturação dos tesauros. Em seguida, é apresentada a arquitetura e as funcionalidades do protótipo, um web service RESTful, que funciona em conjunto com um BD de grafos como repositório para os tesauros. Por fim, uma aplicação cliente para consumir o web service é apresentada, sendo a mesma utilizada para a validação do método.

5.1 Aplicação do método

Nesta seção é demonstrada, na prática, a aplicação do método descrito no Capítulo 4. Para os exemplos de estruturas semânticas apresentadas ao longo desta seção, são considerados os seguintes prefixos (*namespaces*) para abreviar as respectivas URIs:

- prefixo rt: <http://localhost:8080/requirementsThesauri/> (*host* local)
- prefixo rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
- prefixo rdfs: <http://www.w3.org/2000/01/rdf-schema#>
- prefixo skos: <http://www.w3.org/2004/02/skos/core#>
- prefixo owl: <http://www.w3.org/2002/07/owl#>
- prefixo dbr: <http://dbpedia.org/resource/>

Etapa 1: Planejamento Geral do Tesauro

Para o planejamento inicial do tesauro é considerado um requisito de exemplo, “Confidencialidade de Dados de Servidores”. O Quadro 10 apresenta os atributos desse requisito utilizando o *template* proposto neste trabalho.

Quadro 10. *Template* proposto preenchido com os atributos de um requisito.

Campo	Descrição
Identificador	rt:requirements/confidencialidadeDeDadosDeServidores
Nome	Confidencialidade de Dados de Servidores
Nome Preferencial	Confidencialidade de Dados de Servidores
Nome Alternativo	Sigilo de Dados de Servidores
Idioma	pt-BR
Problema	Capacidade de permitir acesso aos dados somente por pessoas devidamente autorizadas.
Contexto	Informações pessoais ou classificadas conforme a lei de acesso a informação - LEI Nº 12.527, DE 18 DE NOVEMBRO DE 2011.

Fonte: Elaboração Própria

Quadro 18. *Template* proposto preenchido com os atributos de um requisito.

Campo	Descrição
Modelo	O canal de comunicação deve utilizar tecnologias reconhecidamente seguras para evitar captura das seguintes informações: [< especificar as informações que são confidenciais >] .
Exemplo	O canal de comunicação deve utilizar tecnologias reconhecidamente seguras para evitar captura das seguintes informações: dados pessoais dos servidores.
Domínio(s)	Gestão de Servidores
Tipo(s) de Sistema	Sistema de Recursos Humanos
Tipo de Requisito	Confidencialidade
Requisito Generalista	Confidencialidade de Dados
Requisitos Específicos	Não há.

Fonte: Elaboração Própria

Com base nele, é necessário definir o formato de entrada dessas informações para o repositório de dados. Os campos podem ser informados pelo usuário ou serem recuperados do repositório. Analisando os atributos, verificou-se que os atributos “Domínio”, “Tipo de Requisito”, “Tipo de Sistema” e “Requisitos Generelista e Específicios”, possuem características que podem ser compartilhadas entre diversos requisitos do tesauro. Por exemplo: “O domínio Gestão pode conter diversos requisitos” ou “Vários requisitos podem estar vinculados ao mesmo tipo de sistema”.

Além disso, eles também podem ser hierarquizados. Por exemplo: “O dominío Gestão de Pessoas pode apresentar Gestão de Servidores e Gestão de Contratados como subdomínios”. Logo, é necessário que esses atributos também sejam tratados como conceitos pela Ontologia SKOS. Logo, suas informações também devem ser mapeadas. Um simples *template* para coletar dados para construir os tesauros de “Domínio” é mostrado no Quadro 11.

Esse *template* também pode ser utilizado para construir tesauros para os atributos “Tipo de Requisito” e “Tipo de Sistema” com pequenas adaptações. “Requisitos Generelista e Específicios” são recuperados por meio do tesauro de requisitos. Os demais atributos podem ser informados por meio de texto digitado pelo usuário.

Além dos atributos já utilizados no *template* de requisitos, é acrescido o atributo “Link DBpedia”, a fim de estabelecer uma conexão com essa base de conhecimento. Nesse sentido, ao recuperar um “Domínio”, “Tipo de Requisito”e “Tipo de Sistema”, o usuário pode acessar a DBpedia para obter mais informações sobre aquele dado, se necessário. Isso vai de encontro

ao quarto princípio *Linked Data*, que descreve que sempre que possível deve-se enriquecer a descrição dos requisitos, relacionado-os com recursos (URIs) de outras fontes de dados consagradas da web semântica.

Quadro 11. *Template* com atributos para construção de tesauro que pode ser utilizado para "Domínio", "Tipo de Requisito"e "Tipo de Sistema".

Campo	Descrição
Identificador	rt:domains/gestaoDePessoas
Nome	Gestão de Pessoas
Nome Preferencial	Gestão de Pessoas
Nome Alternativo	Gestão de Recursos Humanos
Link DBpedia	http://dbpedia.org/resource/Human_resource_management
Descrição	Gestão de Pessoas é a aplicação um conjunto de conhecimentos e técnicas administrativas especializadas no gerenciamento das relações das pessoas com as organizações, com o objetivo de atingir os objetivos organizacionais, bem como proporcionar a satisfação e a realização das pessoas envolvidas.
Domínio Generalista	Gestão
Domínios Específicos	Gestão de Servidores e Gestão de Contratados
Requisitos	Cadastro de pessoas.

Fonte: Elaboração Própria

Etapa 2: Estruturação do Tesauro

Nesse momento é realizado o mapeamento dos atributos do *template* com o auxílio de ontologias para transformar o requisito em triplas recurso-propriedade-valor para que, assim, seja possível descrevê-los e estruturá-los em RDF.

O vocabulário SKOS é ontologia principal utilizada na construção das taxonomias, por isso, há destaque nos grafos para os relacionamentos que utilizam suas propriedades, baseadas no Quadro 9. Cabe ressaltar que a construção da taxonomia não está restrita a SKOS, pois outras ontologias complementares também podem ser usadas, como, por exemplo, RDF Schema[4] e a ontologia OWL[5].

O requisito "Confidencialidade de Dados de Servidores" é utilizado para exemplificar a composição de triplas RDF do requisito no tesauro estruturado pelo SKOS, conforme o Quadro 12. Cada linha representa uma tripla que descreve o requisito cujo URI é *rt:requirements/confidencialidadeDeDadosDeServidores*.

[4] https://www.w3.org/2000/01/rdf-schema#
[5] http://www.w3.org/2002/07/owl#

Quadro 12. Estrutura das triplas RDF relacionadas ao conceito de “Confidencialidade de Dados de Servidores”.

Recurso	**Propriedade**	**Valor**	**Equivalência ao *Template***
rt:requirements/ confidencialidadeDe DadosDeServidores	rdf:type	skos:Concept	Nome (Identificação)
rt:requirements/ confidencialidadeDe DadosDeServidores	rdfs:label	"Confidencialidade de Dados de Servidores"	
rt:requirements/ confidencialidadeDe DadosDeServidores	skos:prefLabel	"Confidencialidade de Dados de Servidores"	Nome Preferencial
rt:requirements/ confidencialidadeDe DadosDeServidores	skos:altLabel	"Sigilo de Dados de Servidores"	Nome Alternativo
rt:requirements/ confidencialidadeDe DadosDeServidores	skos:scopeNote	“Capacidade de permitir acesso aos dados somente por pessoas devidamente autorizadas.”	Problema
rt:requirements/ confidencialidadeDe DadosDeServidores	skos:scopeNote	“Informações pessoais ou classificadas conforme a lei de acesso à informação - LEI Nº 12.527, DE 18 DE NOVEMBRO DE 2011.”	Contexto
rt:requirements/ confidencialidadeDe DadosDeServidores	skos:definition	“O canal de comunicação deve utilizar tecnologias reconhecidamente seguras para evitar captura das seguintes informações: [< especificar as informações que são confidenciais >].”	Modelo
rt:requirements/ confidencialidadeDe DadosDeServidores	skos:scopeNote	“O canal de comunicação deve utilizar tecnologias reconhecidamente seguras para evitar captura das seguintes informações: dados pessoais dos servidores.”	Exemplo
rt:requirements/ confidencialidadeDe DadosDeServidores	skos:broader	rt:domains/gestaoDeServidores	Domínio(s)
rt:requirements/ confidencialidadeDe DadosDeServidores	skos:broader	rt:systemTypes/sistemaDeRecursos Humanos	Tipo(s) de Sistema
rt:requirements/ confidencialidadeDe DadosDeServidores	skos:broader	rt:requirementTypes/ confidencialidade	Tipo de Requisito
rt:requirements/ confidencialidadeDe DadosDeServidores	skos:broader	rt:requirements/confidencialidade DeDados	Requisito Generalista
rt:requirements/ confidencialidadeDe DadosDeServidores	skos:broader	null	Requisitos Específicos

Fonte: Elaboração Própria

A última coluna traz a equivalência de cada tripla ao *template* proposto. Da mesma maneira, o Quadro 13 apresenta as triplas do domínio “Gestão de Pessoas”, identificado pelo

URI: *rt:domains/gestaoDePessoas*. Esse tipo de mapeamento também é realizado para os tesauros de Tipos de Sistema e Tipos de Requisito.

Quadro 13. Estrutura das triplas RDF relacionadas ao conceito de “Gestão de Pessoas”.

Recurso	**Propriedade**	**Valor**	**Equivalência ao *Template***
rt:domains/gestaoDePessoas	rdf:type	skos:Concept	Nome (Identificação)
rt:domains/gestaoDePessoas	rdfs:label	“Gestão de Pessoas”	
rt:domains/gestaoDePessoas	skos:prefLabel	“Gestão de Pessoas”	Nome Preferencial
rt:domains/gestaoDePessoas	skos:altLabel	“Getsão de Recursos Humanos”	Nome Alternativo
rt:domains/gestaoDePessoas	owl:sameAs	http://dbpedia.org/resource/Human_resource_management	Link DBpedia
rt:domains/gestaoDePessoas	skos:definition	“Gestão de Pessoas é a aplicação um conjunto de conhecimentos e técnicas administrativas especializadas no gerenciamento das relações das pessoas com as organizações, com o objetivo de atingir os objetivos organizacionais, bem como proporcionar a satisfação e a realização das pessoas envolvidas.”	Descrição
rt:domains/gestaoDePessoas	skos:broader	rt:domains/gestao	Domínio Generalista
rt:domains/gestaoDePessoas	skos:broader	rt:domains/gestaoDeServidores rt:domains/gestaoDeContratados	Domínios Específicos
rt:domains/gestaoDePessoas	skos:broader	rt:requiremts/cadastroDePessoa	Requisitos

Fonte: Elaboração Própria

Sob o ponto de vista hierárquico (Figura 18), o requisito “Confidencialidade de Dados de Servidores” é um subconceito do conceito de domínio “Gestão de Servidores”, bem como é um subconceito do conceito de tipo de requisito "Confidencialidade" e do conceito de tipo de sistema “Sistema de Recursos Humanos”. Dessa forma, o requisito possuirá uma conexão direta com conceitos dos outros tesauros (Domínios, Tipos de Requisito e Tipos de Sistema). Os grafos das versões iniciais desses tesauros, que foram criados para estudo de viabilidade da proposta, podem ser consultados no Apêndice B.

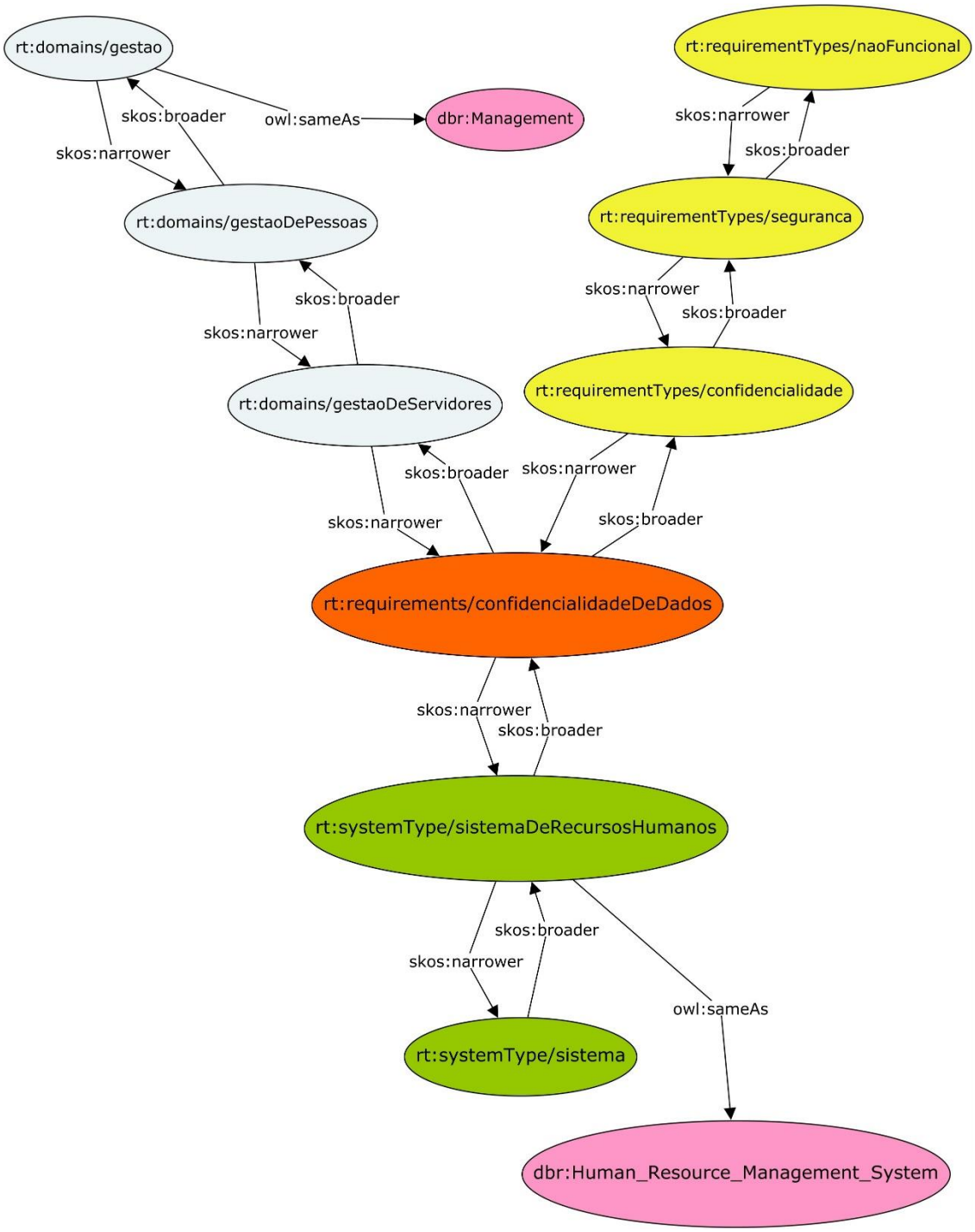


Figura 18. Relacionamentos hierárquicos do requisito “Confidencialidade de Dados de Servidores” com os tesauros de Domínios (Azul), de Tipos de Requisito (Amarelo) e de Tipos de Sistema (Verde). Fonte: Elaboração Própria.

Na Figura 18, ao relacionar, por meio da propriedade *skos:narrower*, o requisito "Confidencialidade de Dados de Servidores" ao tipo de requisito “Confidencialidade”, a máquina infere automaticamente que o requisito também é do tipo "Segurança" e "Não Funcional", o relacionado diretamente com estes supertipos, por meio da propriedade transitiva *skos:narrowerTransitive*, configurando uma rastreabilidade via inferência ontológica (este tipo

de inferência vale para todos os relacionamentos *skos:narrower* e *skos:broader*). Futuramente, demais artefatos produzidos por um projeto de software, tais como casos de uso, diagramas ou componentes de código, podem ser relacionados, direta ou indiretamente, ao requisito, por meio de propriedades de outras ontologias, como, por exemplo, a propriedade *http://purl.org/dc/terms/isRequiredBy* da ontologia *Dublin Core*[6].

Etapa 3: Edição e extensão do tesauro

Para o compartilhamento dos tesauros, visando edição e extensão, foi decidida a criação de um web service RESTful conectado a um BD de grafos RDF, neste caso, o AllegroGraph[7]. A decisão por um web service se deve à possibilidade de compartilhamento do seu conteúdo com diversos dispositivos clientes (computadores, celulares, tablets etc) e pela simplicidade de implementação dos métodos REST para manipulação dos recursos (GET – consulta, POST – inserção, PUT – alteração e DELETE – exclusão) com o *framework* Jena, responsável pela interação com grafos RDF.

Na próxima seção são apresentados detalhes do desenvolvimento de um protótipo desse web service, descrevendo sua arquitetura e as funcionalidades implementadas. Além disso, foi criada uma interface para funcionar como cliente e consumir os recursos gerenciados pelo web service, sendo utilizada no estudo de viabilidade e validação da proposta.

5.2 Desenvolvimento do Protótipo

O protótipo foi desenvolvido utilizando a linguagem Java através do Ambiente de Desenvolvimento Integrado (*Integrated Development Environment* – IDE) IntelliJ[8]. O gerenciamento de dependências e configuração do IDE foi realizado pelo Apache Maven[9]. Para o desenvolvimento do servidor RESTful, foi utilizado o *framework* de código aberto Spring Boot[10].

O banco de dados utilizado é o AllegroGraph, com um SPARQL *Endpoint*. O AllegroGraph é um banco de dados de alta performance que realiza persistência de grafos RDF, combinando o uso eficiente de memória e armazenamento em disco. Também trabalha com inferência RDF++ e o raciocínio Prolog de inúmeros aplicativos clientes, além da linguagem SPARQL (FANZINC, 2020).

[6] https://www.dublincore.org/specifications/dublin-core/dcmi-terms
[7] https://franz.com/agraph/allegrograph/
[8] https://www.jetbrains.com/idea/
[9] https://maven.apache.org/
[10] https://spring.io/projects/spring-boot

O Apache Jena foi utilizado para manipulação dos grafos RDF, que é um framework Java que trabalha com web semântica e *Linked Data*. Permite escrever grafos e extrair dados RDF. Os grafos são representados por meio de um modelo abstrato que pode combinar dados de arquivos, bancos de dados ou URLs (THE APACHE SOFTWARE FOUNDATION, 2011).

As funcionalidades implementadas no protótipo do web service são demonstradas pelo diagrama de casos de uso representado na Figura 19. Manter Requisito, Manter Domínio, Manter Tipo de Requisito e Manter Tipo de Sistema são as funcionalidades CRUD (*Create, Read, Update* e *Delete*).

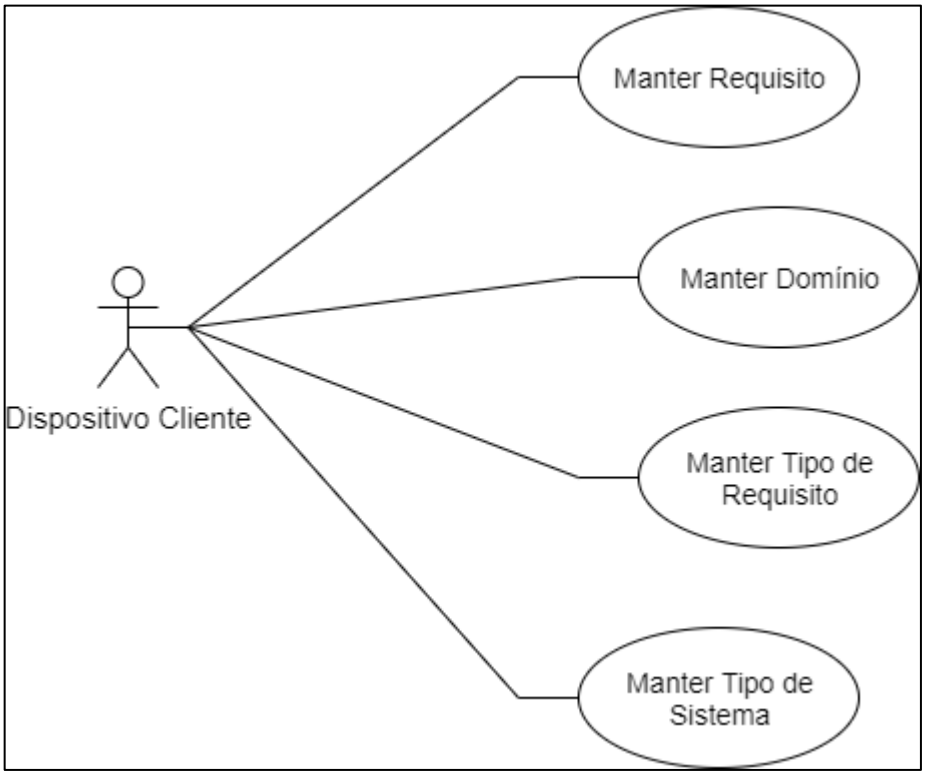


Figura 19. Diagrama de caso de uso do Web Service.
Fonte: Elaboração Própria.

A Figura 20 apresenta, graficamente, a arquitetura do protótipo de web service RESTful que está conectado a um banco de dados RDF, o Allegrograph, com o auxílio do *framework* Jena. O web service é composto por três camadas:

- **Controller**: camada responsável por conter classes que gerenciam os métodos REST (GET, POST, PUT e DELETE) que acessam as funções responsáveis pela manipulação das triplas RDF.
- **Model**: camada onde se encontram classes que representam os *templates* de Requisito, Domínio, Tipo de Requisito e Tipo de Sistema.
- **Service**: camada responsável pela interação com o banco de dados RDF. Essa interação é feita por meio de métodos fornecido pelo *framework* Jena que, além de manipular os

grafos utilizando a linguaguem SPARQL, auxilia na interação com as ontologias (SKOS, RDFs, RDF, OWL etc).

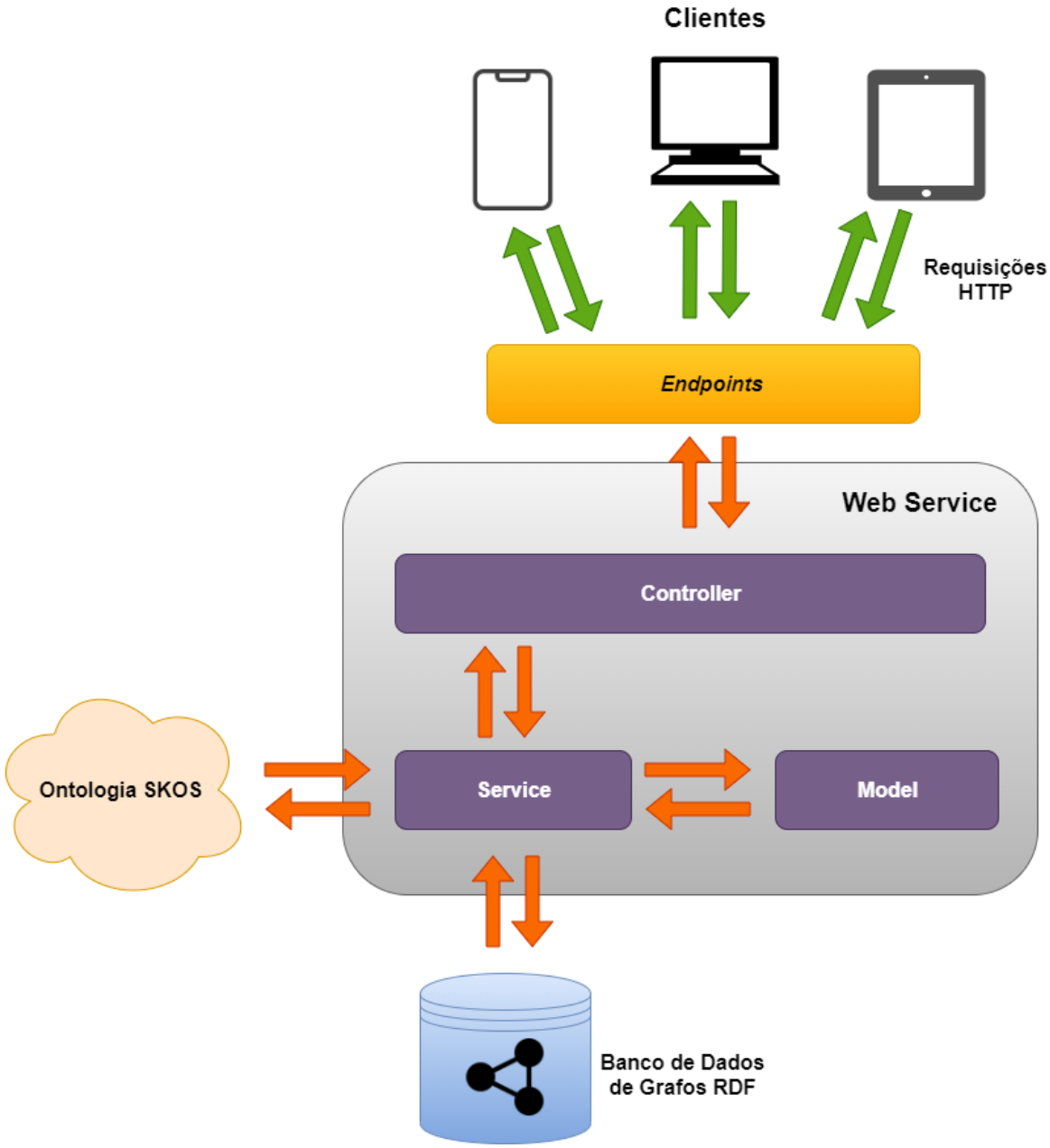


Figura 20. Arquitetura do protótipo de Web Service RESTful para manipulação dos tesauros.
Fonte: Elaboração Própria.

Os dispositivos clientes acessam o web service por meio requisições HTTP aos *Endpoints*, um conjunto de URIs que identificam as ações RDF e os recursos armazenados. As classes da camada Controller processam as requisições e direcionam à classe correspondente na camada Service. Essa classe, interagindo com os objetos da camada Model e aplicando o processamento do *framework* Jena, extrai os dados RDF do grafo recuperado no banco, gerenciando as inferências com as ontologias, e retorna no formato desejado para o cliente (JSON, RDF, TURTLE etc).

De acordo com o estilo da estrutura REST e sua restrição de interface uniforme, o Quadro 14 apresenta uma síntese dos métodos HTTP disponíveis no protótipo e seus URIs de acesso.

Quadro 14. Descrição dos Recursos, URIs e Serviços REST.

Recurso	Método	URI	Produz	Consome
Requisito	GET	/requirements/getAllRequirements	application/json	
	GET	/requirements/{requirementID}	application/json, application/ld+json, application/n-triples, application/rdf+xml, application/turtle, application/rdf+json	
	POST	/requirements/createRequirement		application/json
	POST	/requirements/createRequirementsList		application/json
	PUT	/requirements/{requirementID}		application/json
	DELETE	/requirements/{requirementID}		
Domínio	GET	/domains/getAllDomains	application/json	
	GET	/domains/{domainID}	application/json, application/ld+json, application/n-triples, application/rdf+xml, application/turtle, application/rdf+json	
	POST	/domains/createDomain		application/json
	POST	/domains/createDomainsList		application/json
	PUT	/domains/{domainID}		application/json
	DELETE	/domains/{domainID}		
Tipo de Requisito	GET	/requirementTypes/getAllRequirementTypes	application/json	
	GET	/requirementTypes/{requirementTypeID}	application/json, application/ld+json, application/n-triples, application/rdf+xml, application/turtle, application/rdf+json	
	POST	/requirementTypes/createRequirementType		application/json
	POST	/requirementTypes/createRequirementTypesList		application/json
	PUT	/requirementTypes/{requirementTypeID}		application/json
	DELETE	/requirementTypes/{requirementTypeID}		

Fonte: Elaboração Própria.

Quadro 14. Descrição dos Recursos, URIs e Serviços REST.

Recurso	Método	URI	Produz	Consome
Tipo de Sistema	GET	/systemTypes/getAllSystemTypes	application/json	
	GET	/systemTypes/{systemTypeID}	application/json, application/ld+json, application/n-triples, application/rdf+xml, application/turtle, application/rdf+json	
	POST	/systemTypes/createSystemType		application/json
	POST	/systemTypes/createSystemTypesList		application/json
	PUT	/systemTypes/{systemTypeID}		application/json
	DELETE	/systemTypes/{systemTypeID}		

Fonte: Elaboração Própria.

O código de implementação do protótipo está disponibilizado no GitHub e pode ser acessado pelo endereço https://github.com/rodrigozacarias/BaseTesaurosRequisitos.

5.3 Apresentação do Protótipo

Esta seção apresenta o conteúdo dos serviços REST implementados no protótipo. Para isso, é utilizado exemplos realistas de requisitos, domínios, tipos de requisitos e tipos de sistemas. As requisições HTTP são realizadas com o auxílio do software Postman[11].

Os requisitos são inseridos no banco de dados orientado a grafos por meio do método POST no URI */requirements/createRequirementsList*. Através dessa requisição é possível cadastrar uma lista de requisitos em uma única vez, diferentemente do URI */requirements/createRequirement*, onde somente um requisito é inserido por vez. Esse padrão de inserção é utilizados para todos os demais tesauros.

A Figura 21 apresenta o corpo da requisição contendo dois exemplos realistas de requisitos: “Confidencialidade de Dados de Servidores” e “Cadastro de Servidores”. Esses requisitos estão descritos em JSON e cada linha corresponde a um atributo do *template* proposto nesta dissertação: *requirementID* – Identificador, *label* – Nome, *language* – Idioma, *prefLabel* – Nome preferencial, *altLabel* – Nome alternativo, *problem* – Problema, *context* – Contexto, *template* – Modelo, *example* – Exemplo, *broaderRequirementTypeID* – Tipo de Requisito, *broaderRequirementID* – Requisito Generalista, *broaderDomainID* – Domínios, *broaderSystemTypeID* – Tipos de Sistemas e *narrowerRequiremetID* – Requisitos Específicos.

[11] https://www.getpostman.com/

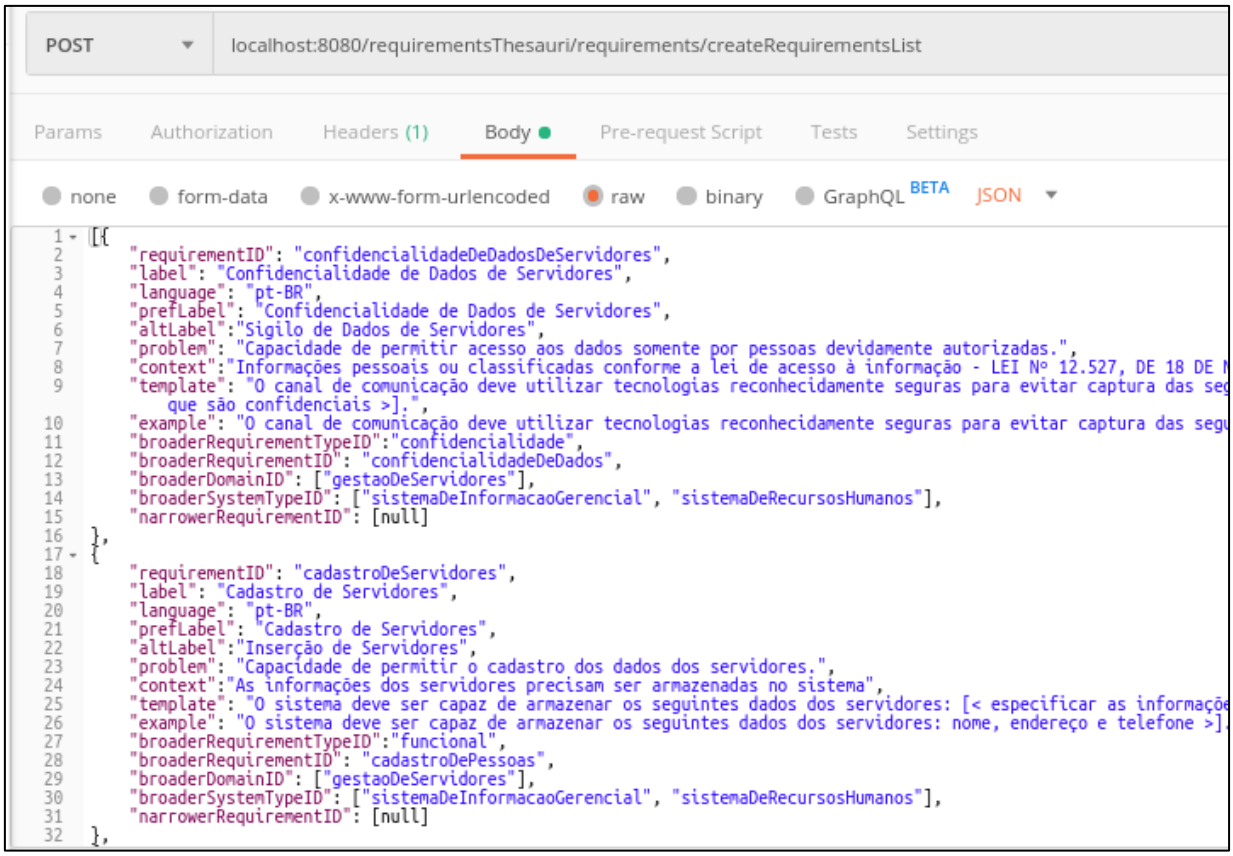


Figura 21. Requisição POST para inserção de uma lista de requisitos.
Fonte: Elaboração Própria.

A notação ID no fim dos atributos simboliza que é informado o identificador do recurso. Após o POST, são exibidos os URIs dos requisitos criados conforme a

Figura 22.

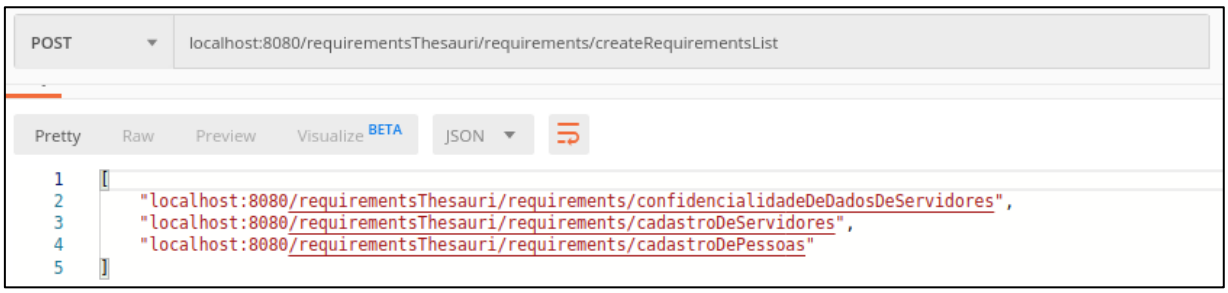


Figura 22. URIs dos requisitos criados pela requisição POST.
Fonte: Elaboração Própria.

Para a criação dos recursos de domínios, tipos de requisitos e tipos de sistemas, o corpo do método POST também recebe os atributos do *template* definido. Conforme a Figura 23, nesse caso os recursos de domínios também são descritos em JSON e cada linha corresponde: *domainID* – Identificador, *label* – Nome, *language* – Idioma, *prefLabel* – Nome preferencial, *altLabel* – Nome alternativo, *description* – Descrição, *linkDbpedia* – Link DBpedia,

broaderDomainID – Domínio Generalista, *narrowerDomainID* – Domínios Específicos e *narrowerRequirementID* – Requisitos.

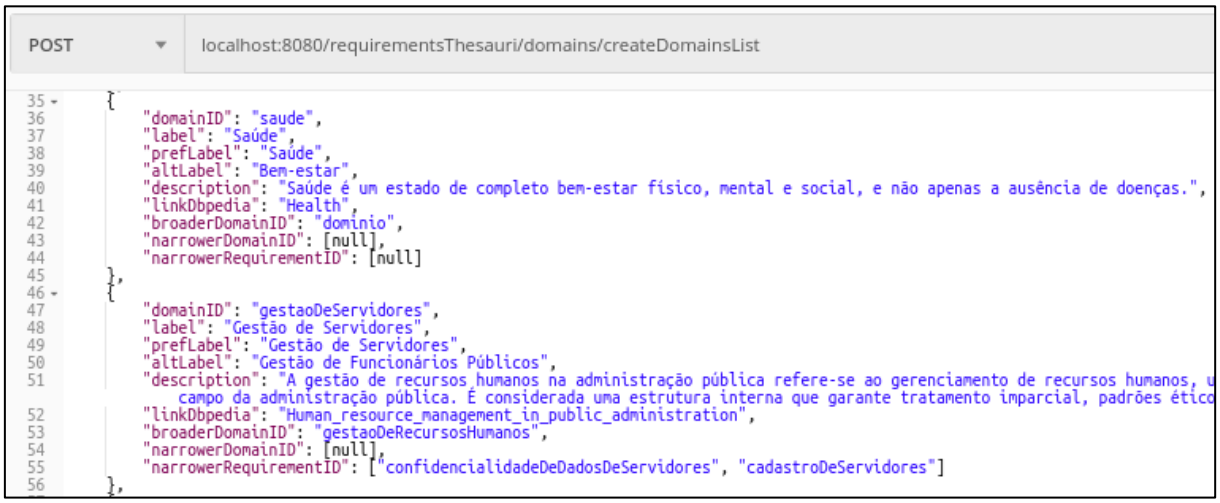


Figura 23. Requisição POST para inserção de uma lista de domínios.
Fonte: Elaboração Própria.

As requisições POST para Tipos de Requisitos e Tipos de Sistema também seguem essa mesma estrutura de Domínios. Após a relização dos POST, também são exibidos os URIs dos recursos criados.

A requisição do tipo GET recupera os recursos armazenados por meio do acesso ao respectivo URI. Os dados podem ser recuperados em diversos formatos ou sintaxes, a depender do parâmetro *Accept* que é passado no *Header* da requisição, conforme a Figura 24. Os formatos podem ser: *application/json*, *application/ld+json*, *application/n-triples*, *application/rdf+xml*, *application/turtle* e *application/rdf+json*.

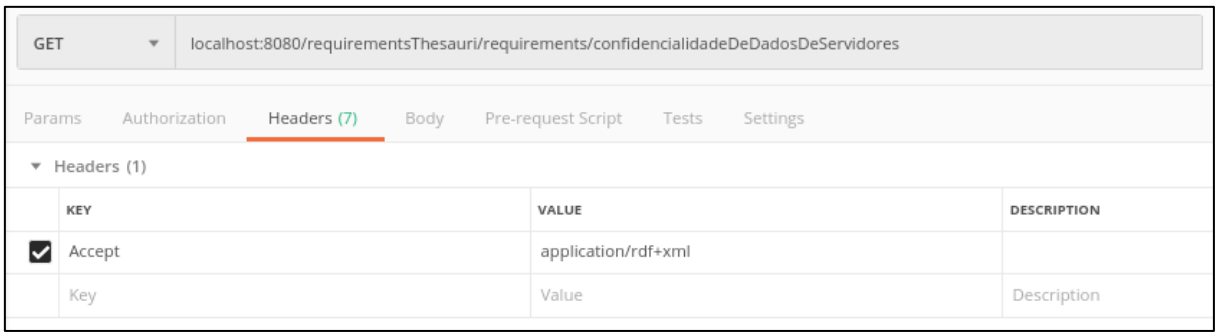


Figura 24. Parâmetro *Accept* para definir o formato do conteúdo de retorno da requisição GET.
Fonte: Elaboração Própria.

A Figura 25 e a Figura 26 apresentam os grafos do requisito “Confidencialidade de Dados dos Servidores” nas sintaxes RDF/XML e Turtle, respectivamente. Esses formatos são lidos e interpretados pelas máquinas.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
   xmlns:j.0="http://purl.org/dc/dcmitype/"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:schema="http://schema.org/"
   xmlns:skos="http://www.w3.org/2004/02/skos/core#"
>
  <rdf:Description rdf:about="localhost:8080/requirementsThesauri/requirements/confidencialidadeDeDadosDeServidores">
    <schema:url rdf:resource="localhost:8080/requirementsThesauri/requirements/confidencialidadeDeDadosDeServidores"/>
    <skos:definition>O canal de comunicação deve utilizar tecnologias reconhecidamente seguras para evitar captura das segui
    <rdfs:subClassOf rdf:resource="localhost:8080/requirementsThesauri/requirementTypes/seguranca"/>
    <skos:broader rdf:resource="localhost:8080/requirementsThesauri/systemTypes/sistemaDeRecursosHumanos"/>
    <skos:broader rdf:resource="http://dbpedia.org/resource/Management_information_system"/>
    <rdfs:subClassOf rdf:resource="http://dbpedia.org/resource/Application_security"/>
    <skos:broader rdf:resource="localhost:8080/requirementsThesauri/domains/gestaoDeServidores"/>
    <rdfs:subClassOf rdf:resource="http://dbpedia.org/resource/Confidentiality"/>
    <rdfs:subClassOf rdf:resource="localhost:8080/requirementsThesauri/requirementTypes/confidencialidade"/>
    <rdfs:subClassOf rdf:resource="http://dbpedia.org/resource/Requirement"/>
    <rdfs:subClassOf rdf:resource="localhost:8080/requirementsThesauri/requirementTypes/null"/>
    <skos:scopeNote>Informações pessoais ou classificadas conforme a lei de acesso à informação - LEI Nº 12.527, DE 18 DE NO
    <skos:note>Capacidade de permitir acesso aos dados somente por pessoas devidamente autorizadas.</skos:note>
    <skos:altLabel>Sigilo de Dados de Servidores</skos:altLabel>
    <rdf:type rdf:resource="http://www.w3.org/2004/02/skos/core#Concept"/>
    <j.0:language>pt-BR</j.0:language>
    <rdfs:label>Confidencialidade de Dados de Servidores</rdfs:label>
    <rdfs:subClassOf rdf:resource="localhost:8080/requirementsThesauri/requirements/confidencialidadeDeDados"/>
    <skos:broader rdf:resource="http://dbpedia.org/resource/Human_resource_management_in_public_administration"/>
    <rdfs:subClassOf rdf:resource="localhost:8080/requirementsThesauri/requirementTypes/naoFuncional"/>
    <rdfs:subClassOf rdf:resource="http://dbpedia.org/resource/Non-functional_requirement"/>
    <skos:broader rdf:resource="localhost:8080/requirementsThesauri/systemTypes/sistemaDeInformacaoGerencial"/>
    <skos:example>O canal de comunicação deve utilizar tecnologias reconhecidamente seguras para evitar captura das seguinte
    <skos:broader rdf:resource="localhost:8080/requirementsThesauri/requirements/confidencialidadeDeDados"/>
    <skos:preLabel>Confidencialidade de Dados de Servidores</skos:preLabel>
    <skos:broader rdf:resource="localhost:8080/requirementsThesauri/requirementTypes/confidencialidade"/>
    <skos:narrower rdf:resource="localhost:8080/requirementsThesauri/requirements/null"/>
    <rdfs:subClassOf rdf:resource="localhost:8080/requirementsThesauri/requirementTypes/requisito"/>
    <skos:broader rdf:resource="http://dbpedia.org/resource/Human_resource_management_system"/>
    <skos:broader rdf:resource="http://dbpedia.org/resource/Confidentiality"/>
  </rdf:Description>
</rdf:RDF>
```

Figura 25. Grafos do requisito “Confidencialidade de Dados de Servidores” em sintaxe RDF/XML.
Fonte: Elaboração Própria.

```
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix dcmit: <http://purl.org/dc/dcmitype/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix uriDom: <localhost:8080/requirementsThesauri/domains/> .
@prefix uriReq: <localhost:8080/requirementsThesauri/requirements/> .
@prefix uriRqt: <localhost:8080/requirementsThesauri/requirementTypes/> .
@prefix uriSys: <localhost:8080/requirementsThesauri/systemTypes/> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

uriReq:confidencialidadeDeDadosDeServidores a skos:Concept ;
    rdfs:label "Confidencialidade de Dados de Servidores" ;
    dcmit:language "pt-BR" ;
    schema:url uriReq:confidencialidadeDeDadosDeServidores ;
    rdfs:subClassOf dbpedia:Application_security,
        dbpedia:Confidentiality,
        dbpedia:Non-functional_requirement,
        dbpedia:Requirement,
        uriRqt:confidencialidade,
        uriRqt:naoFuncional,
        uriRqt:null,
        uriRqt:requisito,
        uriRqt:seguranca,
        uriReq:confidencialidadeDeDados ;
    skos:altLabel "Sigilo de Dados de Servidores" ;
    skos:broader dbpedia:Confidentiality,
        dbpedia:Human_resource_management_in_public_administration,
        dbpedia:Human_resource_management_system,
        dbpedia:Management_information_system,
        uriDom:gestaoDeServidores,
        uriRqt:confidencialidade,
        uriReq:confidencialidadeDeDados,
        uriSys:sistemaDeInformacaoGerencial,
        uriSys:sistemaDeRecursosHumanos ;
    skos:definition "O canal de comunicação deve utilizar tecnologias reconhecidamente seguras para evitar captura das seguii
    skos:example "O canal de comunicação deve utilizar tecnologias reconhecidamente seguras para evitar captura das seguinte
    skos:narrower uriReq:null ;
    skos:note "Capacidade de permitir acesso aos dados somente por pessoas devidamente autorizadas." ;
    skos:preLabel "Confidencialidade de Dados de Servidores" ;
    skos:scopeNote "Informações pessoais ou classificadas conforme a lei de acesso à informação - LEI Nº 12.527, DE 18 DE NO
```

Figura 26. Grafos do requisito “Confidencialidade de Dados de Servidores” em sintaxe Turtle.
Fonte: Elaboração Própria

Já a Figura 27 e a Figura 28 apresentam os grafos do domínio “Gestão de Recursos Humanos” nas sintaxes RDF/XML e Turtle, respectivamente.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:schema="http://schema.org/"
   xmlns:skos="http://www.w3.org/2004/02/skos/core#"
>
  <rdf:Description rdf:about="localhost:8080/requirementsThesauri/domains/gestaoDeRecursosHumanos">
    <rdfs:subClassOf rdf:resource="localhost:8080/requirementsThesauri/domains/null"/>
    <rdfs:subClassOf rdf:resource="http://dbpedia.org/resource/Administration"/>
    <rdfs:subClassOf rdf:resource="http://dbpedia.org/resource/Domain_knowledge"/>
    <skos:altLabel>Gestão de RH</skos:altLabel>
    <skos:broader rdf:resource="localhost:8080/requirementsThesauri/domains/gestao"/>
    <rdf:type rdf:resource="http://www.w3.org/2004/02/skos/core#Concept"/>
    <skos:broader rdf:resource="http://dbpedia.org/resource/Administration"/>
    <rdfs:subClassOf rdf:resource="localhost:8080/requirementsThesauri/domains/gestao"/>
    <rdfs:subClassOf rdf:resource="localhost:8080/requirementsThesauri/domains/dominio"/>
    <skos:note>É a aplicação um conjunto de conhecimentos e técnicas administrativas especializadas no gerenciamento das rel
    <skos:preLabel>Gestão de Recursos Humanos</skos:preLabel>
    <skos:narrower rdf:resource="localhost:8080/requirementsThesauri/domains/gestaoDeContratados"/>
    <schema:url rdf:resource="http://dbpedia.org/resource/Human_resource_management"/>
    <skos:narrower rdf:resource="localhost:8080/requirementsThesauri/requirements/cadastroDePessoas"/>
    <rdfs:label>Gestão de Recursos Humanos</rdfs:label>
    <schema:url rdf:resource="localhost:8080/requirementsThesauri/domains/gestaoDeRecursosHumanos"/>
    <owl:sameAs rdf:resource="http://dbpedia.org/resource/Human_resource_management"/>
    <skos:narrower rdf:resource="localhost:8080/requirementsThesauri/domains/gestaoDeServidores"/>
    <skos:narrower rdf:resource="http://dbpedia.org/resource/Human_resource_management_in_public_administration"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://dbpedia.org/resource/Human_resource_management">
    <rdf:type rdf:resource="http://www.w3.org/2004/02/skos/core#Concept"/>
  </rdf:Description>
</rdf:RDF>
```

Figura 27. Grafos do domínio “Gestão de Recursos Humanos” em sintaxe RDF/XML.
Fonte: Elaboração Própria.

```
@prefix dbpedia: <http://dbpedia.org/resource/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix uriDom: <localhost:8080/requirementsThesauri/domains/> .
@prefix uriReq: <localhost:8080/requirementsThesauri/requirements/> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

uriDom:gestaoDeRecursosHumanos a skos:Concept ;
    rdfs:label "Gestão de Recursos Humanos" ;
    schema:url dbpedia:Human_resource_management,
        uriDom:gestaoDeRecursosHumanos ;
    rdfs:subClassOf dbpedia:Administration,
        dbpedia:Domain_knowledge,
        uriDom:dominio,
        uriDom:gestao,
        uriDom:null ;
    = dbpedia:Human_resource_management ;
    skos:altLabel "Gestão de RH" ;
    skos:broader dbpedia:Administration,
        uriDom:gestao ;
    skos:narrower dbpedia:Human_resource_management_in_public_administration,
        uriDom:gestaoDeContratados,
        uriDom:gestaoDeServidores,
        uriReq:cadastroDePessoas ;
    skos:note "É a aplicação um conjunto de conhecimentos e técnicas administrativas especializadas no gerenciamento das rel
    skos:preLabel "Gestão de Recursos Humanos" .

dbpedia:Human_resource_management a skos:Concept .
```

Figura 28. Grafos do domínio “Gestão de Recursos Humanos” em sintaxe Turtle.
Fonte: Elaboração Própria.

As modificações e atualizações dos recursos são realizadas por meio das requisições do

tipo PUT. A Figura 29 apresenta um exemplo de requisição PUT para atualização do requisito "Confidencialidade de Dados de Servidores". No corpo da requisição, devem ser passados todos os atributos do recurso em formato JSON como se fosse realizar um novo cadastro, uma vez que o registro anterior é excluído durante a atualização/modificação. A diferença nessa requisição está no URI (*/requirements/{requirementID}*), onde é informado o URI do recurso já existente. O procedimento é o mesmo para qualquer recurso armazenado na base de dados.

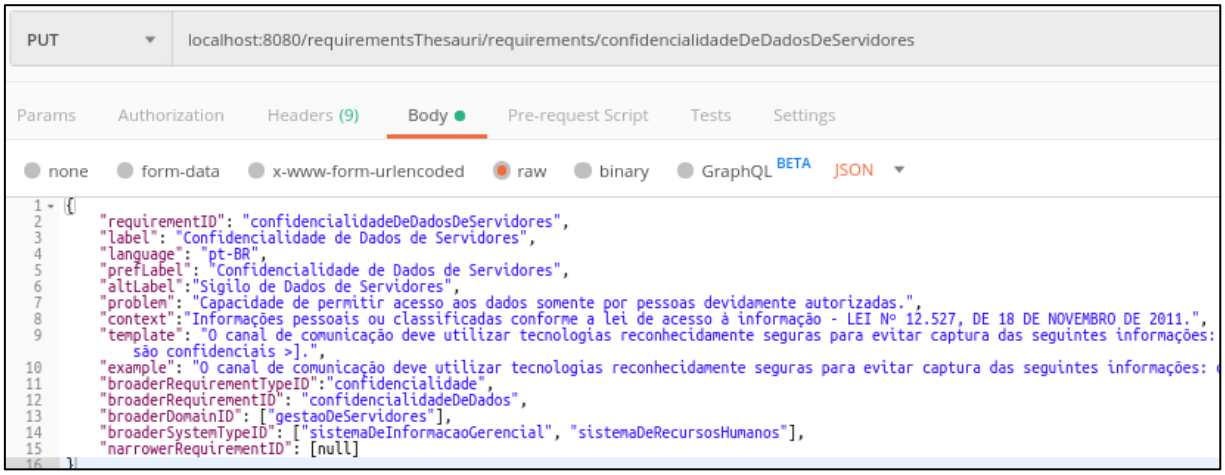


Figura 29. Requisição PUT para atualização do requisito "Confidencialidade de Dados de Servidores".
Fonte: Elaboração Própria.

Por fim, a Figura 30 apresenta um exemplo da requisição DELETE para o domínio "Gestão de Servidores". Nesse caso, não há corpo para a requisição, sendo o único parâmetro o URI do recurso a ser excluído da base. O procedimento é o mesmo para qualquer tipo de recurso da base de dados.

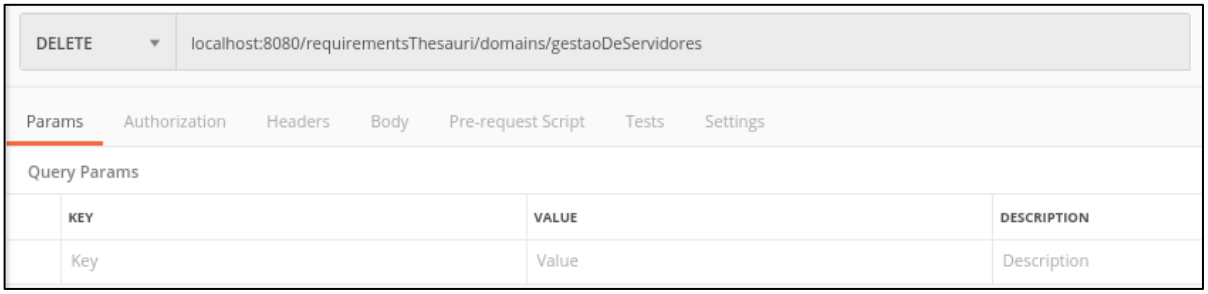


Figura 30. Exemplo de requisição DELETE para um domínio.
Fonte: Elaboração Própria.

5.4 Considerações Finais

Corroborando com Altounian e Gomes (2016), o mapeamento das informações dos requisitos com o auxílio de ontologias para estruturá-los em tesauros trouxe benefícios ao procedimento de consulta e recuperação dos requisitos. A partir da inferência realizada sobre tesauros semânticos, o protótipo conseguiu sugerir novos recursos como respostas para a

consulta do usuário, tornando a busca por requisitos mais inteligente e ágil.

A criação dos grafos dos tesauros semânticos teve como um dos embasamentos a estrutura taxonômica de Motta *et al.* (2017), o que facilitou bastante a implementação com o *framework* Jena. A adoção do padrão de descrição de requisitos de Alexander e Stevens (2002, *apud* WIEGERS; BEATTY, 2013) para o atributo modelo do *template* facilita a leitura e contribui para a completude do requisito.

Quando comparado aos repositórios RCR Editor (CHANG *et al.*, 2011), biblioteca Brechó (SANTOS *et al.*, 2013), Hunter (WANG *et al.*, 2016) e CodeEase (ABID *et al.*, 2017), o principal diferencial do protótipo é trazer as tecnologias das web semântica para o ambiente de reúso e, com isso, pontencializar os mecanismos de buscas. Outro fator diferencial é em relação ao formato de armazenamento. Os repositórios citados estão conectados a bancos de dados comuns, o que pode dificultar a inferência de informações extras em uma consulta, algo que é facilmente obtido quando os dados são estruturados em grafos RDF.

Futuramente, há a intenção de aprimorar os estudos de mecanismos para aferição da qualidade dos requisitos armazenados pelo web service. A biblioteca Brechó pode ser uma importante fonte de embasamento, uma vez que já possui um mecanismo baseado no *feedback* pelos usuários.

6 ESTUDO DE VIABILIDADE

Neste capítulo é descrito um estudo para verificar a viabilidade da proposta desta dissertação com profissionais da área de TI. Inicialmente, são descritos os detalhes da organização e execução do estudo. Em seguida, é feita a análise e discussão dos resultados.

6.1 Descrição do estudo de viabilidade

Este estudo é de natureza qualitativa e o objetivo é avaliar, junto a profissionais de TI, a viabilidade de implementação da proposta desta dissertação. O estudo foi organizado em duas etapas:

-Etapa 1: Apresentação da proposta e do protótipo

Esta etapa é planejada para funcionar como uma apresentação interativa e conduzida da seguinte forma:

1- **Apresentação do Projeto:** Uma pequena apresentação é realizada sobre os conceitos envolvidos na construção do web service semântico: requisitos (tipos e especificação), reúso de requisitos, tesauro, web semântica (ontologia SKOS) e web service RESTful.

2- **Demonstração do funcionamento do protótipo:** Utilizando um conjunto de requisitos previamente descritos, é demonstrado o funcionamento das telas da aplicação cliente e dos métodos REST desenvolvidos, com análise dos formatos de inserção e recuperação das informações dos requisitos.

3- **Interação:** Durante a demonstração, os participantes são estimulados a dialogarem sobre sua percepção em relação:

- aos métodos REST desenvolvidos e o web service de forma geral;
- ao modo como as informações são inseridas e recuperadas;
- aos possíveis ganhos na realização de consultas similares às demonstradas e à possibilidade de uma infinidade de consultas que podem ser executadas na base;
- ao armazenamento da especificação de requisitos de forma semântica;
- à disponibilização de uma base semântica para uso similar à demonstrada;
- à comparação com outras formas de reúso de requisitos;
- às considerações gerais sobre o projeto como um todo.

4- **Documentação:** São realizadas anotações acerca do *feedback* dos profissionais, com ênfase nos pontos que mais geraram discussão.

-Etapa 2: Preenchimento de questionário para avaliação

Após a apresentação interativa com os profissionais, um questionário elaborado na

plataforma Google® é encaminhado por e-mail para preenchimento individual, sem identificação e sem a presença do condutor do estudo, a fim de proporcionar maior liberdade para avaliação da proposta.

O questionário é divido em duas etapas. A primeira parte tem o objetivo de coletar informações para traçar o perfil do participante. A segunda parte apresenta um conjunto de afirmações e hipóteses relacionadas aos objetivos específicos da proposta desta dissertação que devem ser avaliadas pelos participantes utilizando a Escala de Likert[12] (Concordo Totalmente, Concordo Parcialmente, Indiferente, Discordo Parcialmente e Discordo Totalmente) e, em seguida, devem responder a perguntas abertas que abrem espaço para críticas, sugestões e demais considerações sobre a proposta. O questionário completo pode ser consultado no Apêndice D desta dissertação.

6.2 Análise e Discussão de Resultados

O estudo foi realizado com dois grupos de profissionais de TI. O primeiro grupo (P1 ao P4) foi composto por profissionais da iniciativa privada em Itaperuna-RJ e realizado em dezembro de 2019. Já o segundo grupo (P5 ao P8) por profissonais atuantes da administração pública federal em Niterói-RJ e realizado em janeiro de 2020. A Tabela 1 apresenta os perfis dos profissionais a partir dos dados coletados no questionário.

Tabela 1. Perfil do profissionais participantes.

Identificador	Formação Acadêmica	Experiência com projetos de desenvolvimento de software:	Experiência com levantamento e análise de requisitos:	Experiência com reúso de requisitos:
P1	Graduação	De 1 a 5 anos	Menos 1 de ano	Menos 1 de ano
P2	Graduação	Menos 1 de ano	Menos 1 de ano	De 1 a 5 anos
P3	Graduação	De 1 a 5 anos	De 1 a 5 anos	De 1 a 5 anos
P4	Técnico ou Graduando	De 1 a 5 anos	De 1 a 5 anos	De 1 a 5 anos
P5	Técnico ou Graduando	De 1 a 5 anos	De 1 a 5 anos	Nenhuma
P6	Mestrado	Mais de 5 anos	De 1 a 5 anos	Nenhuma
P7	Técnico ou Graduando	De 1 a 5 anos	Nenhuma	Nenhuma
P8	Técnico ou Graduando	Menos 1 de ano	Nenhuma	Menos 1 de ano

Fonte: Elaboração Própria.

Durante a primeira etapa do estudo, os profissionais de ambos os grupos interagiram

[12] É um tipo de escala de resposta psicométrica muito usada questionários e, principalmente, em pesquisas de opinião. Ao responderem esta escala, os perguntados especificam seu nível de concordância com relação a uma afirmação (DALMORO; VIEIRA, 2013).

bem durante a apresentação formal dos conceitos relacionados a este trabalho. A grande maioria não tinha conhecimento em relação a tesauros, mas conseguiram entender facilmente o conceito após a explicação.

Na parte de apresentação do protótipo, o primeiro grupo realizou uma avaliação da versão perliminar do protótipo e o segundo grupo já teve contato com uma versão evoluída com base no *feedback* do grupo anterior. A evolução do projeto no intervalo entre os estudos realizados atende a alguns apontamentos feitos pelo primeiro grupo em relação à necessidade de incluir uma interface gráfica mínima para manipulação dos recursos. A seguir são descritos os pontos de discussão em cada grupo durante a apresentação do protótipo.

-Grupo 1

No primeiro grupo, só foram apresentadas consultas e inserções de dados via requisições HTTP utilizando o software Postman, que simula um cliente para testar serviços RESTful de APIs e web services. Apesar de considerarem o web service uma boa opção para armazenamento e compartilhamento do conteúdo semântico dos requisitos, eles pautaram que uma interface gráfica iria atrair mais usuários, pois somente com os métodos REST, o usuário seria obrigado a desenvolver uma aplicação cliente para acessar os recursos, o que poderia gerar resistência.

Ainda no primeiro grupo, houve uma breve discussão sobre desenvolver aplicações para o ambiente semântico. O profissional P3 utilizou o termo “burocratizar” o processo de desenvolvimento para expressar que talvez alguns profissionais considerem um pouco trabalhoso a aplicação de conceitos semânticos em projetos de software, o que pode gerar certa resistência por parte da comunidade.

-Grupo 2

No segundo grupo, com a apresentação de um protótipo tendo seu conteúdo acesso por uma aplicação cliente com interface gráfica (apresentada no Apêndice C), a aceitação foi bem positiva. Eles acharam bem intuitivo os modos de inserção e consultas aos dados com a interface, além de destacarem a fácil navegabilidade entre as informações cadastradas, podendo acessar um recurso relacionado a um requisito rapidamente.

O principal ponto de discussão no segundo grupo foi em relação ao conteúdo inserido no web service, ressaltando sobre a necessidade ou não de restrição de acesso. Nesse momento, o condutor do estudo relatou que a proposta do web service não se limita a ambientes de livre acesso, e que a tecnologia poderia facilmente ser adaptada para uso em Intranets, seguindo as políticas de segurança de uma organização.

Ainda fizeram ressalvas sobre a qualidade dos requisitos e demais informações que são armazenadas no web service. Uma sugestão dada pelo profissional P7 é a inclusão de um sistema de avaliação do requisito, onde o usuário poderia avaliar, como positivo ou negativo, a experiência de reúso do requisito, seja por meio de votação ou por comentários. Como esta proposta ainda não avançou para o estudo sobre a qualidade do requisito armazenado, essa sugestão foi bem aceita, e será considerada durante a execução de trabalhos futuros.

-Considerações dos grupos

Por fim, ambos os grupos acreditam que o projeto pode trazer novas perspectivas para área de reúso de requisitos, destacando, principalmente, o poder de inferência da máquina sobre os dados. Eles consideraram de grande relevância o fato da máquina sugerir informações ou recursos extras durante uma consulta, o que pode agilizar o processo de documentação dos requisitos ou também no descobrimento de outros requisitos não levantados previamente.

-Questionário de Avaliação

Após o fim da apresentação e do grupo de discussão, foi encaminhado para os e-mails dos profissionais o questionário para avaliação da proposta da dissertação, como a segunda etapa do estudo. A seguir é realizada uma análise geral das respostas coletadas. A íntegra das respostas do questionário pode ser consulta no Apêndice D.

A Figura 31 traz uma nova perspectiva, em forma de Diagrama de Pareto[13], da frequência das respostas. O eixo vertical esquerdo apresenta a escala do número de ocorrências para uma resposta, sendo a referência para o gráfico de barras. Já o eixo vertical direito apresenta a escala de percentual de frequência acumulada das respostas, sendo a referência para o gráfico de linha.

Pode-se destacar que a opção “Concordo Totalmente” se repetiu por 88 vezes o que corresponde a quase 69% das respostas. Já em relação ao extremo da escala, a opção “Discordo Totalmente”, não houve nenhuma ocorrência.

[13] O Diagrama de Pareto é uma ferramenta de qualidade que, por meio de um gráfico de barras, ordena as frequências das ocorrências, da maior para a menor, muito utilizado para a priorização dos problemas (COELHO; SILVA; MANIÇOBA, 2016).

Figura 31. Diagrama de Pareto sobre as frequências das opções de resposta escolhidas no questionário.
Fonte: Elaboração Própria.

A seguir são destacados alguns gráficos de respostas sobre os pontos mais importantes do questionário que estão alinhados com os objetivos específicos do trabalhos e perfazendo o processo de avaliação da proposta.

A Figura 32 ressalta que 75% dos participantes consideram que a especificação estruturada de requisitos facilita a documentação. Nesse sentido, propor um *template* para especificação de requisitos de forma estruturada para o contexto semântico se faz uma escolha acertada.

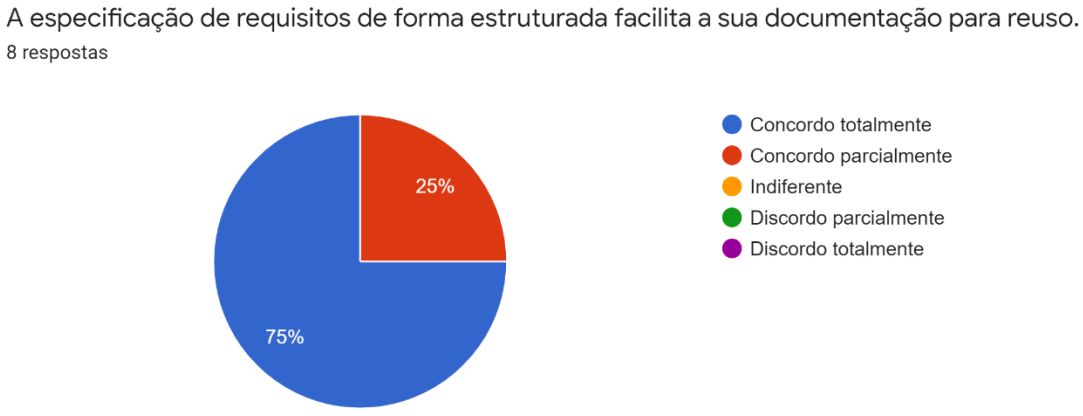


Figura 32. Gráfico de resposta para A5.
Fonte: Elaboração Própria

A Figura 33 e a Figura 34 atestam que, num primeiro momento, a organização de requisitos em forma de tesauros pode parecer complexa, mas a maior parte dos profissionais também consideram que a estrutura hierárquica desse SOC facilita a localização de requisitos para o reúso em um domínio.

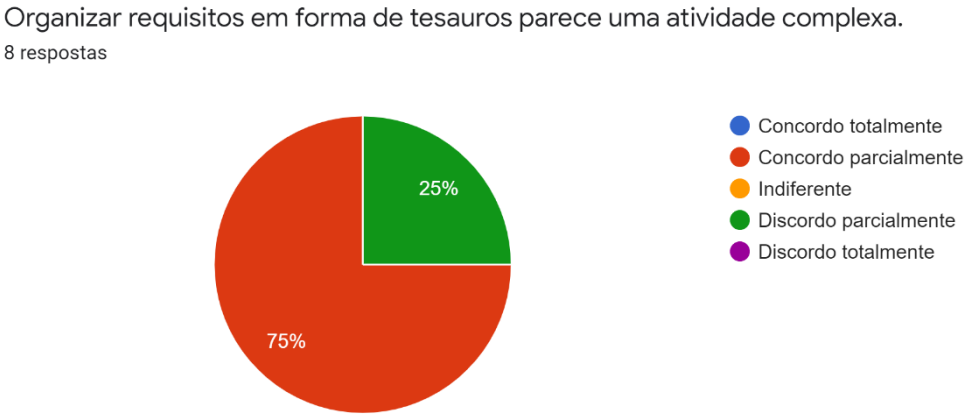


Figura 33. Gráfico de resposta para A7.
Fonte: Elaboração Própria

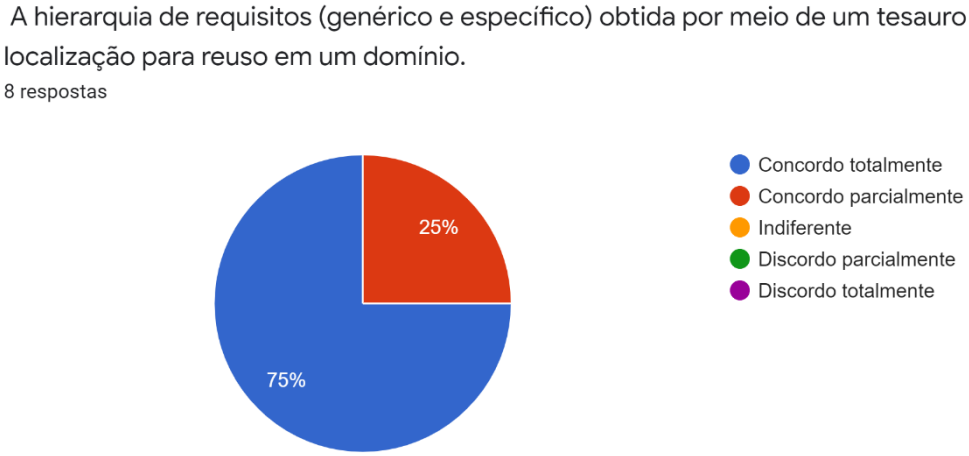


Figura 34. Gráfico de resposta para A8.
Fonte: Elaboração Própria.

A Figura 35 e a Figura 36 destacam que a maioria dos profissionais afirmam que a web semântica pode potencializar a facilitar a atividade de reúso de requisitos, principalmente, em função das inferências realizadas pelas máquinas sobre os requisitos estruturados em grafos RDF. Isso reforça as opiniões dos grupos expressadas durante a discussão na primeira etapa.

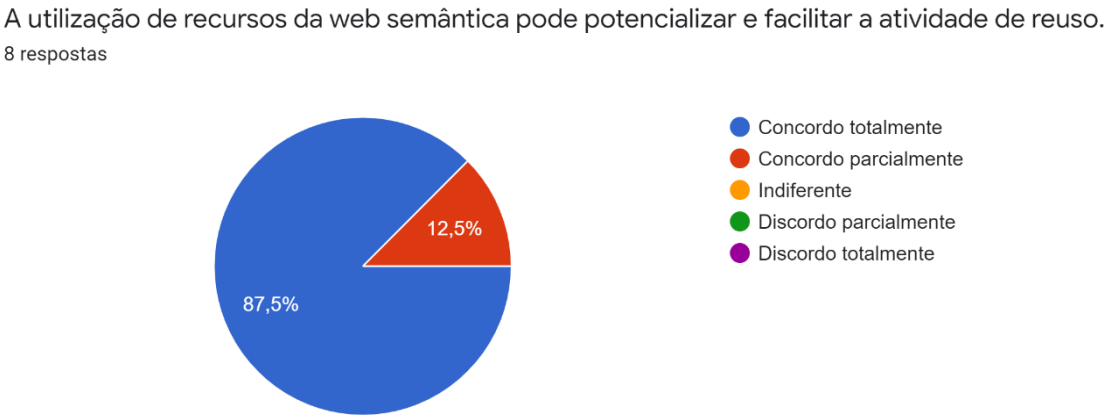


Figura 35. Gráfico de resposta para A9.
Fonte: Elaboração Própria

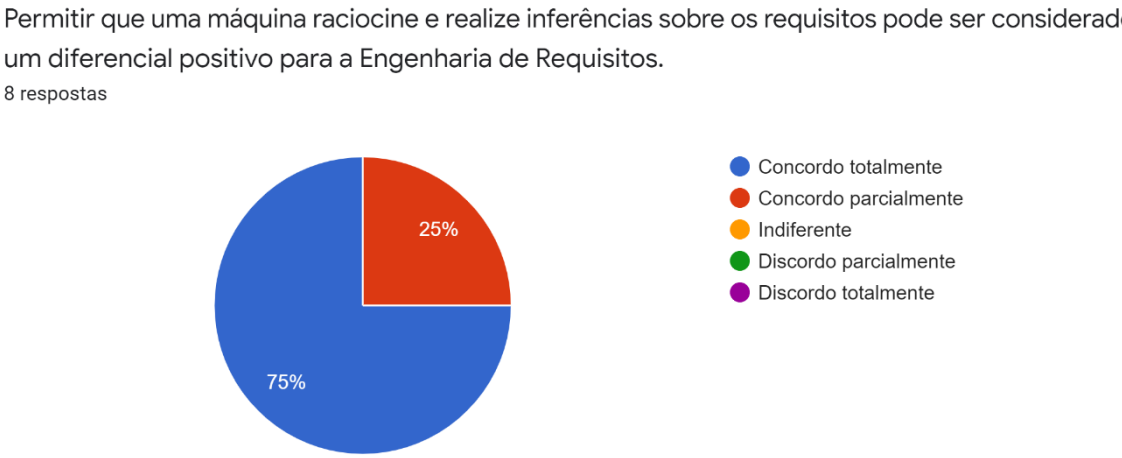


Figura 36. Gráfico de resposta para A10.
Fonte: Elaboração Própria

Segundo a Figura 37 e a Figura 38, a ideia de utilizar um web service como um repositório de conhecimento para reúso é bem aceita por grande parte dos profissionais. A maioria deles ainda concorda que o web service é uma boa opção para compartilhar o conteúdo dos requisitos na web, pois permite o acesso geograficamente distribuído, possui métodos de facíl entendimento e implementação, além de facilitar a interoperabilidade com diferentes dispositivos.

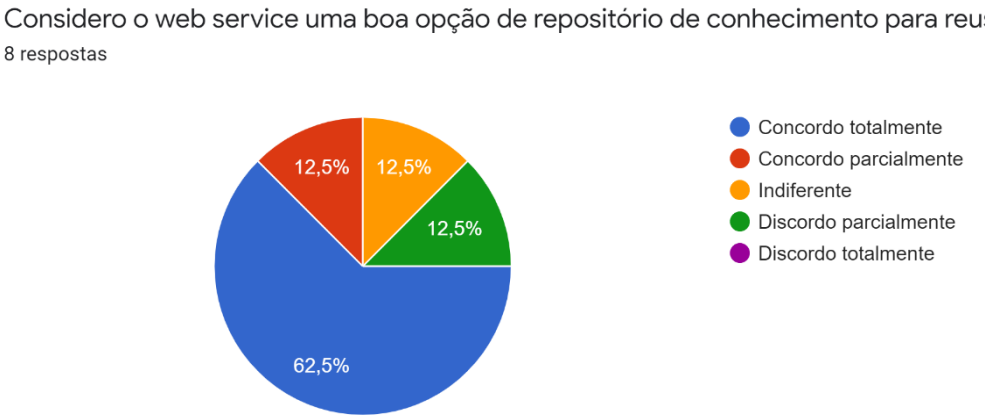


Figura 37. Gráfico de resposta para A12.
Fonte: Elaboração Própria

Figura 38. Gráfico de resposta para A15.
Fonte: Elaboração Própria

Por fim, a Figura 39 comprova que todos os profissionais também acreditam que a proposta pode ser expandida com novas formas de consultas e inferências, novos atributos podem ser acrescentados aos requisitos e novos componentes e artefatos de um projeto de software podem ser mapeados e disponibilizados para reúso. Desta forma, esta proposta pode ser considerada um pontapé inicial para essa infinidade de possibilidades.

Figura 39. Gráfico de resposta para A16.
Fonte: Elaboração Própria.

As perguntas abertas constantes no fim do formulário foram as que mais trouxeram pontos de discussão. Quando questionados se considerariam a utilização do web service apresentado na sua rotina como fonte para reúso de requisitos, o seguinte *feedback* fez com que o protótipo fosse reavaliado e evoluído.

P3:"*A princípio não me parece interessante, pois eu iria precisar uma interface gráfica própria para poder operar os requisitos do tesauros. Um webservice só se torna interessante quando vou desenvolver um software que vai tirar algum proveito dos dados, nesse caso só imagino um software que atenderia aos próprios desenvolvedores de software, ou seja, algo que a própria ferramenta proposta no trabalho poderia fazer.*"(sic)

Esse fato já havia sido relatado no primeiro grupo de discussão presencial e fez com que o protótipo fosse revisto e implementada uma interface gráfica por meio de uma aplicação cliente para testar os serviços REST, que foi bem aceita pelo segundo grupo.

Quando questionados sobre impedimentos para a proposta, pode-se destacar as seguintes respostas:

P3:"*O impedimento principal acredito que seja a familiaridade dos desenvolvedores de software com os conceitos de tesauros e websemântica. Ambos se tratam de áreas de predominância do ambiente acadêmico, e no âmbito comercial pode haver uma resistência no uso dos mesmos devido ao pensamento de alguns desenvolvedores acharem que isso irá "burocratizar" o processo todo.*"(sic)

P4: "*A curva de aprendizado, seria mais um item para estudo num ramo que sempre há com que se atualizar e como é algo novo, se não for bem apresentado os profissionais optaram por darem foco em algo mais próximo do que algo novo.*"(sic)

P7:“*Talvez um projeto tenha algum requisito que não possa divulgar (por tratar-se de segredo de uma empresa) para os usuários do web service.*”(sic)

Diante dessas respostas, um dos próximos passos do trabalho é traçar estratégias para evitar uma possível resistência por parte da comunidade em relação à utilização do web service ou ao desenvolvimento de novas aplicações para utilizar seus serviços por consideram algo mais trabalhoso ou “burocratizado”, talvez pela necessidade de mínimo conhecimento sobre as ontologias ou tecnologias da web semântica.

Em relação ao comentário de P7, o código do web service será compartilhado em plataformas de hospedagem, como GitHub, dessa forma sendo possível reproduzir uma versão que se adeque às políticas de segurança da organização e para funcionar em uma Intranet.

Por fim, quando questionados sobre sugestões para o projeto, pode-se destacar as seguintes:

P4:“*Fazer um manual ensinando como mexer e como implantá-lo, além de construir uma interface simples para uso visual. No futuro fazer com que a máquina trasse e interligue os requisitos, montando e indicando aos profissionais templates prontos com uma gama de requisitos que a máquina "ache" que o projeto apresentará*.”(sic)

P6:“*Estender usando ou criando ontologias de requisitos ligadas a diferentes metodologias de desenvolvimento de software; Verificar como seria a gerência de configuração dos requisitos e sua relação com as ontologias. Inclusive como seria o caso de requisitos que migram entre ontologias conforme o tempo*.”(sic)

P7:“*Permitir que os utilizadores possam avaliar a qualidade dos requisitos exibidos pelo web service.*”(sic)

A sugestão de P4 se faz realmente necessária, pois a criação de um manual ou dicionário para o entendimento da estrutura do web service e das propriedades utilizadas nos mapeamentos também pode diminuir possíveis resistências. Além disso, deixar a máquina mais inteligente nos processos de inferência e criar mecanismos para lidar com a gerência de configuração e requisitos em diferentes metodologias de desenvolvimento, com bem ressaltado por P6, irão agregar robutez ao processo de reúso.

Por fim, a questão de aferição qualidade do requisitos armazenados é a prioridade para trabalhos futuros, pois é um ponto crítico para o reúso. Como destacado na primeira etapa do estudo, um esquema para coletar o *feedback* do usuário, seja por votação ou comentário, sobre a experiência de reúso do requisito pode ser uma alternativa viável para o projeto.

6.3 Considerações Finais

O estudo de viabilidade apresentado neste capítulo permitiu avaliar, de forma abrangente, a proposta desta dissertação. Os grupos de profissionais de TI expuseram suas percepções sobre o tema e identificaram os principais benefícios da proposta, bem como as limitações e fragilidades.

Dentre os benefícios, é possível destacar a organização do conhecimento relativo aos requisitos durante a documentação em formato de tesauros e, a partir disso, a possibilidade de inferências que a máquina pode realizar a partir dos relacionamentos estabelecidos. Além disso, eles também concordam que a estrutura facilita a navegabilidade entre as informações agregadas aos requisitos que, consequentemente, agilizam seu processo de recuperação.

Em relação às fragilidades e limitações, foram ressaltados dois pontos: a qualidade do requisito disponibilizado para reúso e uma possível resistência por parte da comunidade para desenvolver no ambiente da web semântica. Esses aspectos trazem uma reflexão sobre a proposta e auxiliam na elaboração de metas futuras e próximos passos para o trabalho.

7 CONSIDERAÇÕES FINAIS

Organização, armazenamento e compartilhamento de conhecimento relativo a requisitos para reutilização em projetos de desenvolvimento de software em um determinado domínio ainda são um desafio e um tópico relevante de pesquisa na atualidade. Nesse sentido, esta dissertação teve como objetivo o desenvolvimento de uma proposta de documentação para requisitos estruturada em tesauros semânticos e, com isso, permirtir sua publicação na web para pontencializar o seu reúso em outros projetos de um mesmo domínio.

Em relação aos objetivos específicos, este trabalho conseguiu identificar padrões de descrição e especificação de requisitos e estabeleceu relacionamentos entre os mesmos. A partir disso, foi proposto um *template* para especificar requisitos de forma estruturada no contexto semântico, sendo utilizada, como base principal, a ontologia SKOS para o mapeamento das relações dos requisitos com seus atributos, transformando-os em grafos RDF, e estabelecendo a taxonomia que representa os tesauros semânticos.

Outro objetivo específico atingido foi a proposição de diretrizes para a criação dessa taxonomia, que permite classificar, buscar e recuperar a definição de requisitos de software em conceito de tesauros, com auxílio de ontologias e de acordo com os princípios *Linked Data*.

Em relação ao desenvolvimento de um web service RESTful, este trabalho apresentou um protótipo, conectado a um BD de grafos, para funcionar como um repositório e demonstrar, na prática, a aplicabilidade da proposta. Esse protótipo passou por um estudo de viabilidade com profissionais de TI, sendo possível checar a organização do conhecimento de requisitos em forma de tesauros e rastreabilidade entre as informações. O protótipo obteve avaliações, em sua grande maioria, favoráveis a sua implementação.

O trabalho atingiu parcialmente o objetivo específico relacionado ao acesso geograficamente distribuído, ou seja, de diversos lugares aos tesauros de requisitos, uma vez que o web service ainda não foi hospedado na web e somente o seu código de implementação foi compartilhado no GitHub. O único objetivo específico ainda não atingido foi em relação à criação de um mecanismo adequado para aferir e documentar aspectos relacionados à qualidade dos requisitos, como detalhado na seção de limitações.

7.1 Contribuições

Esta dissertação apresenta as seguintes contribuições para área de ER e organização de conhecimento:

- Propõe um *template* para a especificação de requisitos de forma estruturada para o contexto semântico. Isso facilita o mapeamento das informações de um requisito por ontologias, bem

como a implementação de mecanismo de busca e recuperação de seu conteúdo.

- Propõe diretrizes, por meio de um método apresentado no Capítulo 4, para a construção de tesauros semânticos de requisitos baseado na metologia de Maculan (2015), utilizada para a construção de tesauros na área de Ciência da Informação. Com isso, o intuito é orientar a organização desse conhecimento e permitir que o tesauro possua uma estrutura passível de navegabilidade entre os conceitos e processamento de inferências de informações semânticas.

7.2 Limitações

Algumas limitações iniciais detectadas durante o desenvolvimento desta dissertação foram:

- A ausência de implementação de mecanismo para a aferição do qualidade dos requisitos armazenados, fator de grande relevância para atividade de reúso. Ainda é necessário maior aprofundamento nesse tema para chegar na proposta de um mecanismo ideal para tal ação, sendo esse um objetivo específico ainda não atingido.
- Uma questão levantada durante o estudo de viabilidade foi sobre uma possível resistência da comunidade em desenvolver para a web semântica, talvez por considerar que seja um processo mais trabalhoso ter que adquirir um conhecimento mínimo das tecnologias semânticas, algo mais distante do cotidiano da maioria dos desenvolvedores.

7.3 Trabalhos Futuros

Como trabalhos futuros, será dada continuidade à pesquisa realizada sobre mecanismos de aferição da qualidade do requisito para que sejam implementados o mais breve possível. Nesse processo, serão consideradas as sugestões dos profissionais participantes do estudo de viabilidade e consultado como essa ação é realizada em outros repositórios.

É prevista a evolução do *template* proposto, podendo surgir variações de acordo com tipo de tipo de requisito a ser mapeado. Novas taxonomias também podem ser criadas para outras categorias de requisitos não abordadas nesta dissertação, tais como requisitos de negócio, de *stakeholder*, de transição, de projeto ou de qualidade, como definidos no PMBOK (2013).

É prevista a evolução da interface gráfica da aplicação cliente, de forma a torná-la cada vez mais amigável e intuitiva, baseando-se em princípios para melhoria de usabilidade. Outra sugestão proveniente do estudo de viabilidade que será acatada é a criação de um manual para desenvolvedores e usuários do web service RESTful para que haja uma rápida familiarização com seu funcionamento e, com isso, diminuir possíveis resistências.

Também será dado prosseguimento aos procedimentos de disponibilização de uma versão aprimorada do web service RESTful na web, com novas formas de inferência e de

verificação da rastreabilidade entre as informações dos requisitos armazenados, monitorando seu comportamento durante sua utilização. Também serão estudados procedimentos de gerenciamento das sugestões da comunidade desenvolvedora para melhorias no código do web service.

À medida que resultados positivos forem sendo obtidos, a proposta poderá evoluir para novos estágios, com acréscimo de mapeamento dos outros artefatos de software, tais como diagramas, código-fonte, documentação de testes etc. Assim, espera-se que está proposta seja considerada um passo inicial e motivação para a implementação de uma documentação semântica completa de um projeto de software para reúso.

REFERÊNCIAS BIBLIOGRÁFICAS

ABID, S. et al. **Codeease: harnessing method clone structures for reuse**. 2017 IEEE 11th International Workshop on Software Clones (IWSC). **Anais**... In: 2017 IEEE 11TH INTERNATIONAL WORKSHOP ON SOFTWARE CLONES (IWSC). fev. 2017

AL BALUSHI, T. H.; SAMPAIO, P. R. F.; LOUCOPOULOS, P. Eliciting and prioritizing quality requirements supported by ontologies: a case study using the ElicitO framework and tool. **Expert Systems**, v. 30, n. 2, p. 129–151, maio 2013.

ALTOUNIAN, M. M. DE A.; GOMES, B. P. DE M. A recuperação semântica da informação no contexto do controle externo. **Revista do TCU**, n. 137, p. 31–41, dez. 2016.

ANGRISANI, R. R. **Aplicação de Ontologias à Engenharia de Requisitos em Ambientes de DDS**. Mestrado em Ciência da Computação—Porto Alegre: Pontifícia Universidade Católica do Rio Grande do Sul, 2006.

BARCELOS, L. V. **ESPECIFICAÇÃO DE REQUISITOS NO DOMÍNIO DE SISTEMAS DE INFORMAÇÃO COM O USO DE PADRÕES**. Dissertação de Mestrado em Ciência da Computação—São Carlos: Universidade Federal de São Carlos (UFSCAR), 2016.

BERNERS-LEE, T. et al. Tabulator: Exploring and Analyzing linked data on the Semantic Web. p. 16, 2006.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The Semantic Web. **Scientific American**, p. 4, maio 2001.

BIGIO, M. T.; MOTTA, R. W.; SRTREIT, R. E. Métricas de qualidade na coleta de requisitos. **Revista Gestão do Conhecimento e Tencologia da Informação**, v. 1, n. 1, p. 88–97, 2017.

BITENCOURT, A. S.; PAIVA, D. M. B.; CAGNIN, M. I. **Elicitação de Requisitos a partir de Modelos de Processos de Negócio em BPMN: Uma Revisão Sistemática**. Anais do Simpósio Brasileiro de Sistemas de Informação (SBSI). **Anais**... In: SIMPÓSIO BRASILEIRO DE SISTEMAS DE INFORMAÇÃO (SBSI). Florianópolis: 17 maio 2016. Disponível em: <https://sol.sbc.org.br/index.php/sbsi/article/view/5963>. Acesso em: 24 jan. 2020

BRICKLEY, D.; GUHA, R. V.; MCBRIDE, B. **RDF Schema 1.1**. Disponível em: <https://www.w3.org/TR/rdf-schema/>. Acesso em: 21 jul. 2019.

BROWN, R. B. K. et al. Computationally efficient ontology selection in software requirement planning. **Information Systems Frontiers**, v. 18, n. 2, p. 349–358, abr. 2016.

CHANG, C.-H. et al. **XML-Based Reusable Component Repository for Embedded Software.** IEEE, 2011. Disponível em: <http://ieeexplore.ieee.org/document/6032262/>. Acesso em: 19 fev. 2018

COELHO, F. P. DE S.; SILVA, A. M.; MANIÇOBA, R. F. APLICAÇÃO DAS FERRAMENTAS DA QUALIDADE: ESTUDO DE CASO EM PEQUENA EMPRESA DE PINTURA. **REFAS - Revista FATEC Zona Sul**, v. 3, n. 1, p. 31–45, out. 2016.

CRUZ, J. A. G. DA. **Mapeamento de Bancos de Dados para Domínios Semânticos**. Mestrado em Ciência da Computação—Goiânia: Universidade Federal do Goiás (UFG), 2015.

DABBAGH, M.; LEE, S. P.; PARIZI, R. M. Functional and non-functional requirements prioritization: empirical evaluation of IPA, AHP-based, and HAM-based approaches. **Soft Computing**, v. 20, n. 11, p. 4497–4520, nov. 2016.

DALMORO, M.; VIEIRA, K. M. Dilemas na Construção de Escalas Tipo Likrt: o número de itens e a disposição influenciam nos resultados? **RGO - REVISTA GESTÃO ORGANIZACIONAL**, v. 6, p. 161–174, 2013.

DERMEVAL, D. et al. Applications of ontologies in requirements engineering: a systematic review of the literature. **Requirements Engineering**, v. 21, n. 4, p. 405–437, nov. 2016.

DIAMANTOPOULOS, T.; SYMEONIDIS, A. Enhancing requirements reusability through semantic modeling and data mining techniques. **Enterprise Information Systems**, v. 12, n. 8–9, p. 960–981, 21 out. 2018.

FANZINC. **AllegroGraph - Semantic Graph Database**. Disponível em: <https://franz.com/agraph/allegrograph/>. Acesso em: 26 jan. 2020.

FARFELEDER, S. et al. Ontology-Driven Guidance for Requirements Elicitation. In: ANTONIOU, G. et al. (Eds.). **The Semanic Web: Research and Applications**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. v. 6644p. 212–226.

FAWCETT, J.; AYERS, D.; QUIN, L. R. E. **Beginning XML**. Indianapolis, USA: John Wiley & Sons, 2012.

FERREIRA, H. N. M.; NAVES, T. F. **REUSO DE SOFTWARE: SUAS VANTAGENS, TÉCNICAS E PRÁTICAS**. IX Enacomp. **Anais**... In: ENCONTRO ANUAL DE COMPUTAÇÃO. Catalão/GO: 2011

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. Doctoral dissertation—Irvine: University of California, 2000.

FRAKES, W. B.; KANG, K. Software Reuse Research: Status and Future. **IEEE Transactions on Software Engineering**, v. 31, p. 529–536, 2005.

FREITAS JUNIOR, N.; JACYNTO, M. D. DE A. Um protótipo Linked data para catalogação semântica de publicações. **Perspectivas em Ciência da Informação**, v. 21, n. 4, p. 48–65, dez. 2016.

GAMBO, I. P.; SORIYAN, A. H.; IKONO, R. N. A Proposed Process Model for Requirements Engineering using Delphi Techniques for Prioritisation. **International Journal of Information Technology and Computer Science**, v. 7, n. 1, p. 73–80, 8 dez. 2014.

GÓES, A. DE S.; SILVA, J. P. DA; BARROS, R. M. DE. Melhoria no Processo de Levantamento de Requisitos para Software de Gestão Pública: Um Estudo de Caso utilizando Instruções de Trabalho. **Revista de Sistemas de Informação da FSMA**, n. 12, p. 21–32, 2013.

GOLDIN, L.; BERRY, D. M. Reuse of requirements reduced time to market at one industrial shop: a case study. **Requirements Engineering**, v. 20, n. 1, p. 23–44, 1 mar. 2015.

GUALHANO, M. A.; VERA, A. S. C.; MEDEIROS JUNIOR, R. C. **Biblioteca de Requisitos sobre propriedades de segurança em Sistemas Computacionais**. ENCompIF - II Encontro

Nacional de Computação dos Institutos Federais. **Anais**... In: XXXIV CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. Brasília: Sociedade Brasileira de Computação, 2014Disponível em: <http://rgdoi.net/10.13140/2.1.3434.6566>. Acesso em: 10 jan. 2019

HEATH, T.; BIZER, C. **Linked data: Evolving the web into a global data space. Synthesis lectures on the semantic web: theory and technology**. 1ª ed. Ebook: Morgan & Claypool, 2011. v. 1

HEFLIN, J. **OWL Web Ontology Language Use Cases and Requirements**. Disponível em: <https://www.w3.org/TR/2004/REC-webont-req-20040210/>. Acesso em: 20 jul. 2019.

HITZLER, P. et al. **OWL 2 Web Ontology Language Primer (Second Edition)**. Disponível em: <https://www.w3.org/TR/2012/REC-owl2-primer-20121211/#What_is_OWL_2.3F>. Acesso em: 21 jul. 2019.

IEEE. **IEEE standard glossary of software engineering terminology**. New York, N.Y: Institute of Electrical and Electronics Engineers, 1990.

IEEE. **IEEE 830-1998 - IEEE Recommended Practice for Software Requirements Specifications**. Disponível em: <https://standards.ieee.org/standard/830-1998.html>. Acesso em: 10 jul. 2019.

ISO. **ISO/IEC/IEEE 29148:2018(en), Systems and software engineering — Life cycle processes — Requirements engineering**. Disponível em: <https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:29148:ed-2:v1:en>. Acesso em: 20 jul. 2019.

ISO/IEC 9126-1. **NBR ISO/IEC 9126-1: Engenharia de software - Qualidade de produto**, 2011.

ISO/IEC-25010. **ISO/IEC 25010. Systems and software engineering / Systems and software Quality Requirements and Evaluation (SQuaRE) / System and software quality models.** Geneva, Switzerland: ISO/IEC, 2011.

ISOTANI, S. et al. Ontology Driven Software Engineering: A Review of Challenges and Opportunities. **IEEE Latin America Transactions**, v. 13, n. 3, p. 863–869, mar. 2015.

JURISTO, N.; MORENO, A.; SANCHEZ-SEGURA, M.-I. Guidelines for Eliciting Usability Functionalities. **IEEE Transactions on Software Engineering**, v. 33, n. 11, p. 744–758, nov. 2007.

KAIYA, H. et al. **Enhancing Domain Knowledge for Requirements Elicitation with Web Mining**. 2010 Asia Pacific Software Engineering Conference. **Anais**... In: 2010 ASIA PACIFIC SOFTWARE ENGINEERING CONFERENCE. Sydney, NSW, Australia: IEEE, nov. 2010

KAIYA, H.; SAEKI, M. **Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach**. Fifth International Conference on Quality Software (QSIC'05). **Anais**... In: FIFTH INTERNATIONAL CONFERENCE ON QUALITY SOFTWARE (QSIC'05). Melbourne, Australia: IEEE, 2005. Disponível em: <http://ieeexplore.ieee.org/document/1579139/>. Acesso em: 21 jul. 2019

KOPCZYNSKA, S.; NAWROCKI, J. **Using non-functional requirements templates for elicitation: A case study**. 2014 IEEE 4th International Workshop on Requirements Patterns (RePa). **Anais**... In: 2014 IEEE 4TH INTERNATIONAL WORKSHOP ON REQUIREMENTS PATTERNS (REPA). Karlskrona, Sweden: IEEE, ago. 2014. Disponível em: <http://ieeexplore.ieee.org/document/6894844/>. Acesso em: 10 jul. 2019

MACULAN, B. C. M. DOS S. **ESTUDO E APLICAÇÃO DE METODOLOGIA PARA REENGENHARIA DE TESAURO: REMODELAGEM DO THESAGRO**. Tese—Belo Horizonte: Universidade Federal de Minas Gerais, jun. 2015.

MANOLA, F.; MILLER, E. **RDF Primer**. Disponível em: <https://www.w3.org/TR/rdf-primer/>. Acesso em: 8 maio. 2019.

MARCONDES, C. H. "Linked data" – dados interligados - e interoperabilidade entre arquivos, bibliotecas e museus na web. **Encontros Bibli: revista eletrônica de biblioteconomia e ciência da informação**, v. 17, n. 34, p. 22, 9 ago. 2012.

MCCRAE, J. P. **The Linked Open Data Cloud**. Disponível em: <https://lod-cloud.net/>. Acesso em: 21 jul. 2019.

MENDES, P. R.; REIS, R. M.; MACULAN, B. C. M. DOS S. TESAUROS NO ACESSO À INFORMAÇÃO: UMA RETROSPECÇÃO. **Revista ACB: Biblioteconomia em Santa Catarina**, v. 20, n. 1, p. 49–66, abr. 2015.

MILES, A.; BECHHOFER, S. **SKOS Simple Knowledge Organization System Namespace Document - HTML Variant, 18 August 2009 Recommendation Edition**. Disponível em: <https://www.w3.org/2009/08/skos-reference/skos.html>. Acesso em: 21 jul. 2019.

MONTEQUIN, V. R. et al. Analysis of the Success Factors and Failure Causes in Information & Communication Technology (ICT) Projects in Spain. **Procedia Technology**, v. 16, p. 992–999, 2014.

MOTTA, R. W. et al. Elicitação de requisitos orientada por ontologias de domínio. **Revista Gestão do Conhecimento e Tencologia da Informação**, v. 1, n. 2, p. 20, 2017.

NARDI, J. C.; FALBO, R. DE A. **Uma Ontologia de Requisitos de Software**. 2006

NATIONAL INFORMATION STANDARDS ORGANIZATION. **ANSI/NISO Z39.19-2005: guidelines for the construction, format, and management of monolingual thesauri**, 2005. Disponível em: <https://groups.niso.org/apps/group_public/download.php/12591/z39-19-2005r2010.pdf>. Acesso em: 20 jul. 2019

NETO, G. G. DA C. **Estudos qualitativos para elicitação de requisitos: uma abordagem que integra análise sócio-cultural e modelagem organizacional**. Tese de Doutorado em Ciência da Computação—Recife: Universidade Federal do Pernambuco (UFPE), mar. 2008.

NINAUS, G. et al. **Content-based recommendation techniques for requirements engineering**. 2014 IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering (AIRE). **Anais**... In: 2014 IEEE 1ST INTERNATIONAL WORKSHOP ON ARTIFICIAL INTELLIGENCE FOR REQUIREMENTS ENGINEERING (AIRE). Karlskrona, Sweden: IEEE, ago. 2014. Disponível em: <http://ieeexplore.ieee.org/document/6894853/>. Acesso em: 21 jul. 2019

OLARONKE, I.; RHODA, I.; ISHAYA, G. An Appraisal of Software Requirement Prioritization Techniques. **Asian Journal of Research in Computer Science**, v. 1, n. 1, p. 1–16, 19 abr. 2018.

PACHECO, C. et al. Reusing functional software requirements in small-sized software enterprises: a model oriented to the catalog of requirements. **Requirements Engineering**, v. 22, n. 2, p. 275–287, jun. 2017.

PALOMARES, C.; QUER, C.; FRANCH, X. Requirements reuse and requirement patterns: a state of the practice survey. **Empirical Software Engineering**, v. 22, n. 6, p. 2719–2762, 1 dez. 2017.

PETROV, I.; BUCHMANN, A. **Architecture of OMG MOF-based repository systems**. Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services - iiWAS '08. **Anais**... In: THE 10TH INTERNATIONAL CONFERENCE. Linz, Austria: ACM Press, 2008. Disponível em: <http://portal.acm.org/citation.cfm?doid=1497308.1497346>. Acesso em: 2 ago. 2019

PMI. **Um Guia Do Conhecimento Em Gerenciamento De Projetos**. 5ª ed. Newtown Square, Pennsylvania: Project Management Institute, 2013.

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

PRUD'HOMMEAUX, E.; SEABORNE, A. **SPARQL Query Language for RDF**. Disponível em: <https://www.w3.org/TR/rdf-sparql-query/>. Acesso em: 8 maio. 2019.

RAMALHO, R. A. S. Análise do Modelo de Dados SKOS: Sistema de Organização do Conhecimento Simples para a Web. **Informação & Tecnologia**, v. 2, n. 1, p. 66–79, jul. 2015.

RAPOSO JUNIOR, R. R. **Brechó-ABC: An Integrated Mechanism for Evaluation, Search and Categorization of Software Components.** Rio de Janeiro: Universidade Federal do Rio de Janeiro, 2007.

REMPEL, P.; MADER, P. **A quality model for the systematic assessment of requirements traceability**. 2015 IEEE 23rd International Requirements Engineering Conference (RE). **Anais**... In: 2015 IEEE 23RD INTERNATIONAL REQUIREMENTS ENGINEERING CONFERENCE (RE). Ottawa, ON, Canada: IEEE, ago. 2015. Disponível em: <http://ieeexplore.ieee.org/document/7320420/>. Acesso em: 9 mar. 2019

SALES, R. DE; CAFÉ, L. Diferenças entre Tesauros e Ontologias. **Perspectivas em Ciência da Informação**, v. 14, n. 1, p. 99–116, abr. 2009.

SANTOS, R. P. DOS; TOSTES, L. R.; WERNER, C. M. L. **A Brechó-EcoSys extension to support negotiation in the software ecosystems context**. 2013 IEEE 14th International Conference on Information Reuse Integration (IRI). **Anais**... In: 2013 IEEE 14TH INTERNATIONAL CONFERENCE ON INFORMATION REUSE INTEGRATION (IRI). ago. 2013

SANTOS, R. et al. **Brechó 2.0: Uma Ferramenta para Apoiar a Gerência de Reutilização.** . In: VI WORKSHOP ANUAL DO MPS. Campinas - SP: 2010

SAYÃO, M.; LEITE, J. C. S. DO P. Rastreabilidade de Requisitos. **Monografias em Ciência da Computação**, v. 20, n. 5, p. 26, maio 2005.

SEGUNDO, J. E. S.; CONEGLIAN, C. S. Tecnologias da web semântica aplicadas a organização do conhecimento: padrão SKOS para construção e uso de vocabulários controlados descentralizados. **Organização do Conhecimento e Diversidade Cultural**, p. 224–233, 2015.

SHIVA, S. G.; SHALA, L. A. **Software Reuse: Research and Practice**. Fourth International Conference on Information Technology (ITNG'07). **Anais**... In: FOURTH INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY (ITNG'07). Las Vegas, NV, USA: IEEE, abr. 2007Disponível em: <http://ieeexplore.ieee.org/document/4151749/>. Acesso em: 2 ago. 2019

SILVA, A. et al. Evaluation of an approach to define elicitation guides of non-functional requirements. **IET Software**, v. 11, n. 5, p. 221–228, 1 out. 2017.

SILVA, A. R. DA. **Uma Abordagem para Definição de Guias de Elicitação de Requisitos Não-Funcionais**. Tese de Doutorado em Informática Aplicada—Fortaleza: Universidade de Fortaleza, 2016.

SOMMERVILLE, I. **Software engineering**. 9ª ed. Boston: Pearson, 2011.

SPANOUDAKIS, G.; ZISMAN, A. SOFTWARE TRACEABILITY: A ROADMAP. In: CHANG, S. K. (Ed.). **Handbook Of Software Engineering And Knowledge Engineering**. UK: WORLD SCIENTIFIC, 2005. p. 395–428.

TAMAE, R. Y.; LIMA, P. R. WEB SERVICES: UMA NOVA VISÃO DA ARQUITETURA DE APLICAÇÕES DISTRIBUÍDAS NA INTERNET. **REVISTA CIENTÍFICA ELETRÔNICA DE SISTEMAS DE INFORMAÇÃO**, v. Ano I, n. 2, p. 3, fev. 2005.

THE APACHE SOFTWARE FOUNDATION. **Apache Jena**. Disponível em: <https://jena.apache.org/>. Acesso em: 26 jan. 2020.

VANDERLEI, T. A. et al. **Folksonomy in a Software Component Search Engine – Cooperative Classification through Shared Metadata**. XX Simpósio Brasileiro de Engenharia de Software (SBES). **Anais**... In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE. Florianópolis: 2006

W3C. **W3C Brasil - World Wide Web Consortium Escritório Brasil**. Disponível em: <http://www.w3c.br/Sobre/>. Acesso em: 8 maio. 2019.

WANG, Y. et al. **Hunter: next-generation code reuse for Java**. Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. **Anais**...: FSE 2016.Seattle, WA, USA: Association for Computing Machinery, 1 nov. 2016. Disponível em: <https://doi.org/10.1145/2950290.2983934>. Acesso em: 28 jan. 2020

WIEGERS, K. E.; BEATTY, J. **Software requirements**. 3ª ed. Redmond, Washington: Microsoft Press, 2013.

XAVIER, J. R. **Criação e Instanciação de Arquiteturas de Software Específicas de Domínio no contexto de uma Infra-estrutura de Reutilização**. Tese—Rio de Janeiro: Universidade Federal do Rio de Janeiro, jun. 2001.

APÊNDICE A – Seleção dos Trabalhos Relacionados

O procedimento aplicado para busca e seleção dos trabalhos científicos tem como base a metodologia apresentada por Bitencourt *et al.* (2016), já validada por um especialista na área de ER. Inicialmente, são definidos o objetivo de pesquisa, a estratégia de busca, os critérios de inclusão e exclusão.

- Objetivo da Pesquisa: identificar e analisar trabalhos e estudos científicos que abordem, preferencialmente de forma conjunta, os temas de Engenharia de Reguisitos, Tesauro, Tecnologias da Web Semântica e Reúso de Software, a fim de verificar a existência ou não de trabalhos similares ao objetivo deste trabalho.

- Questões de Pesquisa: buscam efetuar um levatamento sobre como os trabalhos abordam cada um dos temas ou podem oferecer algum embasamento para o desenvolvimento da abordagem proposta, considerando o objetivo da pesquisa. As questões de pesquisa são apresentadas a seguir:

- Q1: Há aplicação de tecnologia da web semântica na ER com o objetivo de reúso?
- Q2: Como são desenvolvidos os repositórios que têm finalidade de reúso de componentes de software?
- Q3: Que tipos de relação existem entre tesauros e ER?

- Estratégia de busca: nesta etapa é definida a *string* de busca padrão que será aplicada nas bases de conhecimento. A mesma pode ser adaptada ao formato de entrada de cada base para facilitar a busca. O Quadro 15 apresenta a *string* padrão que foi elaborada a partir das palavras-chave que representam os temas de interesse deste trabalho. O operador “OR” tem a finalidade de indicar os termos semelhantes que são considerados para a busca e o operador “*” indica que qualquer variação desinencial é aceita.

Quadro 15. *String* de busca padrão para a seleção dos trabalhos.

Palavras-chave	***String* de busca padrão**
Requirements Engineering Thesaurus Semantic Web Software Reuse	("requirements engineering" OR "requirement*") AND ("thesaurus" OR "thesauri") AND ("semantic web" or “ontolog*”) AND ("software reus*")

Fonte: Elaboração Própria.

O Quadro 16 apresenta as bases de conhecimento elencadas para esta pesquisa e comumente utilizadas em pesquisas em ER, conforme Bitencourt *et al.* (2016).

Quadro 16. Bases de busca para a seleção dos trabalhos.

Bases de Busca
ACM Digital Library
EEEXplore Digital Library
Science Direct
Scopus
Web of Science
Biblioteca Digital Brasileira de Teses e Dissertações - (BDTD)
Workshop de Engenharia de Requisitos - (WER)

Fonte: Elaboração Própria.

- Critérios de Inclusão e Exclusão: foram definidos com intuito de encontrar os trabalhos mais relevantes para o objeto de pesquisa, bem como aqueles que possam responder as questões de pesquisas. No Quadro 17 são apresentados os critérios definidos.

Quadro 17. Critérios de inclusão e exclusão para a seleção dos trabalhos.

Tipo	**ID**	**Descrição**
Inclusão	CI1	O trabalho reúne os quatros temas deste trabalho: ER, Tesauro, Tecnologias da Web Semântica e Reúso de Software.
Inclusão	CI2	O trabalho apresenta a utilização de tesauros para documentação ou especificação de requisitos.
Inclusão	CI3	O trabalho apresenta a aplicação de uma tecnologia da web semântica para mapeamento ontológico na ER.
Inclusão	CI4	O trabalho descreve um tipo de repositório para reúso de componentes de software.
Exclusão	CE1	O trabalho não apresenta algum tipo de interseção entre os temas pesquisados.
Exclusão	CE2	O trabalho está duplicado ou é um resumo de outro já selecionado.
Exclusão	CE3	O trabalho é uma revisão sistemática ou estudo bibliográfico.
Exclusão	CE4	A tecnologia da web semântica apresentada no trabalho não possui ligação direta com área de requisitos ou com reúso de software.

Fonte: Elaboração Própria.

As pesquisas nas bases científicas foram realizadas durante a segunda quinzena de julho de 2019. Ao utilizar a *string* padrão definida, foi percebido que não houve retorno de trabalhos em nenhuma das bases, o que pode indicar a não existência de trabalhos científicos que envolvam os quatro temas até o momento.

A partir desse resultado, foi adotada uma estratégia para localizar trabalhos que pudessem oferecer algum embasamento para o desenvolvimento da abordagem proposta. A pesquisa foi divida por área temáticas, adaptando a *string* de busca conforme as palavras-chave de cada tema, conforme Quadro 18.

Quadro 18. *Strings* de busca para cada área temática.

Área Temática	***String* de busca**
A1: Tesauros na Engenharia de Requisitos	("requirements engineering" OR "requirement*") AND ("thesaurus" OR "thesauri") AND (“mapping”)
A2: Tecnologias da Web Semântica na Engenharia de Requisitos	("requirements engineering" OR "requirement*") AND ("semantic web" or “ontolog*”) AND (“mapping”)
A3: Repositórios para Reúso de Componentes de Software	("software reus*") AND (“repository”)

Fonte: Elaboração Própria.

Foi adicionado o termo mapeamento (*mapping*) às *strings* de buscas de A1 e A2 com finalidade de buscar técnicas de mapeamento com uso de tesauros ou ontologias que possam ser base para o mapeamento que será proposto neste trabalho. Na A3, a finalidade é investigar como os repositórios são organizados e lidam com a disponibilidade dos componentes para reúso. A Tabela 2 apresenta o número de trabalhos encontrados após a utilização das *strings* de busca por área temática, somando o retorno de todas as bases.

Tabela 2. Quantitativo bruto de trabalhos retornados após a busca por área temática.

Área Temática	**Quantitativo de trabalhos**
A1	215
A2	411
A3	1.219

Fonte: Elaboração Própria.

Em seguida, os trabalhos passaram pelas seguintes etapas de verificação de acordo com a área temática:

1- Exclusão dos documentos duplicados.
2- Verificação da adequação do trabalho ao escopo de pesquisa por meio leitura do título e do resumo/*abstratc*. Caso negativo, o trabalho é excluído da pesquisa;

3- Aplicação dos critérios de inclusão e exclusão definidos no Quadro 17.

Nas seções seguintes deste capítulo são apresentados os trabalhos selecionados após a execução dos procedimentos mencionados.

Área Temática: Tesauros na Engenharia de Requisitos

Neste tópico são apresentados alguns trabalhos utilizaram tesauros como ferramentas para a melhoria de algum processo ou atividade da ER. O primeiro trabalho atende aos crítérios CI2 e CI3 e o segundo trabalho ao CI3.

Uso de tesauro para mapeamento ontológico

Kaiya e Saeki (2005) propõem um método para análise de requisitos de software baseado em uma ontologia de domínio. O diferencial é que esse sistema ontológico consiste de um tesauro e regras de inferência. Uma parte do tesauro é responsável por conter os conceitos do domínio e relacionamentos específicos para o processamento adequado da ontologia. A Figura 40 apresenta como ocorre o mapeamento dos requisitos para a ontologia de domínio.

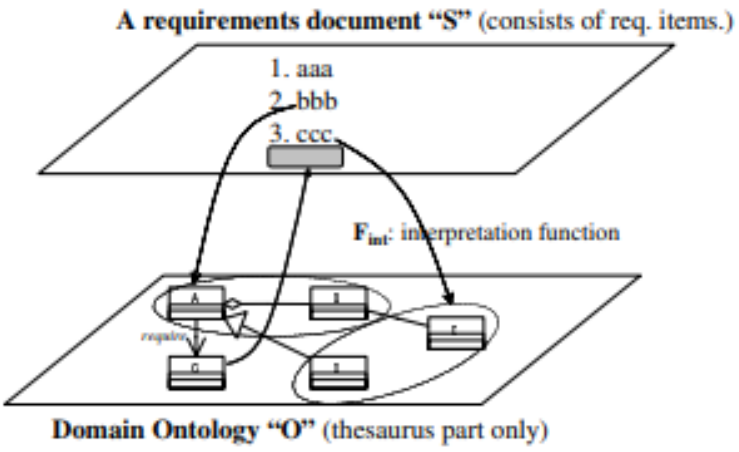


Figura 40. Mapeamento dos requisitos para a ontologia.
Fonte: Kaiya e Saeki (2005).

Esse mapeamento é feito com o auxílio de um tesauro, onde devem ser informados os dados de um documento de requisitos conforme os conceitos apresentados na Figura 41. Nesse caso, o analista deve mapear dados sobre cada requisito acerca da funcionalidade, atores, objetos, restrições, aspectos de qualidade, entre outros; além de estabelecer relacionamentos entre os mesmos, tais como generalização, agregação, associação, etc; para que seja estabelecida a semântica. É apresentado a seguir um exemplo considerando um requisito "Tocar Música":

- Função (Tocar) e Objeto (Música)

- Relacionamento de Generalização (Operação, Tocar)

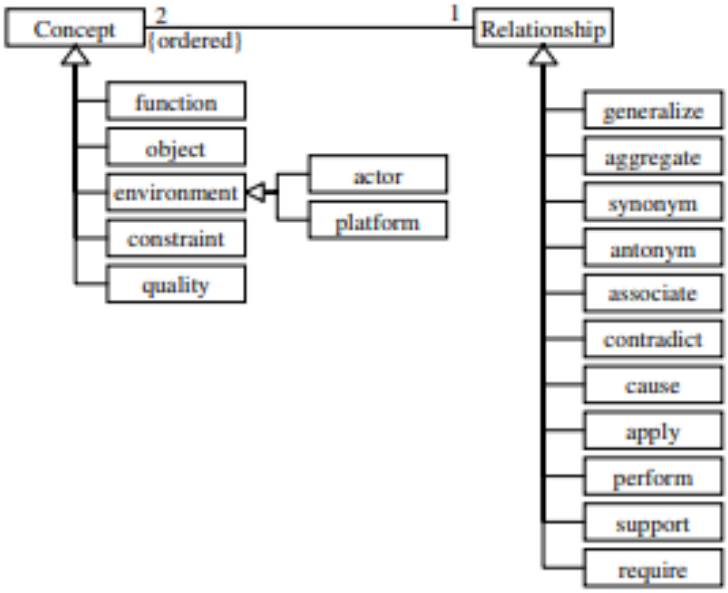


Figura 41. Conceitos e relacionamentos do Tesauro.
Fonte: Kaiya e Saeki (2005).

Após o mapeamento, os analistas podem obter inferências por meio da ontologia em relação à especificação dos requisitos e à semântica do domínio da aplicação, por meio de três processamentos semânticos: (1) detecção de incompletude e inconsistência incluídas em uma especificação de requisitos, (2) medição da qualidade de uma especificação em relação ao seu significado e (3) previsão de mudanças nos requisitos com base na análise semântica de um histórico de alterações.

Por meio de um estudo de caso, os autores puderam comprovar que o tesauro contribuiu positivamente para a construção da ontologia de domínio, tornando mais precisos os processamentos semânticos.

***IntelliReq* – Ferramenta para reuso de requisitos baseada em Tesauros**

Ninaus *et al.* (2014) apresentam a *IntelliReq*, uma ferramenta baseada em técnicas de inteligência artifical para apoiar o reuso de requisitos. Para isso, a *IntelliReq* auxilia na aferição da qualidade dos requistos por meio de processamento de linguagem natural, onde os principais termos dos requisitos são comparados com os termos de um tesauro de mesmo domínio do conjunto de requisitos, buscando similaridade entre os termos. Com isso, é possível detectar eventuais ambiguidades ou incompletudes no significado dos requisitos.

O texto que contém os requisitos passa por um pré-processamento onde são removidas as *stopwords* (palavras que não acrescentam significado relevante, como artigos ou

conjunções). Por funcionar em plataforma web, a *IntelliReq* pode se conectar a tesauros já existentes, como o *OpenThesaurus*. Por meio de processos de comparação, a ferramenta verifica se o termo utilizado possui significado adequado para aquele domínio dos requisitos. Caso não possua, uma palavra-chave é sugerida para substituir o termo anterior. Através disso, a *IntelliReq* permite verificar e melhorar a qualidade de um conjunto de requisitos, incentivando o reuso posterior. Outro ponto é que a ferramenta auxilia na redução do número de termos diferentes que expressam o mesmo significado, objetivando maior clareza para o reuso.

Para testar a eficácia da ferramenta, foi realizado um estudo de caso com dois grupos de estudantes, onde ambos deveriam escolher dentre um conjunto de requisitos, quais poderiam ser reutilizados, só que somente um trabalharia com a ferramenta. Como resultado, observou-se que o grupo que utilizou a ferramenta conseguiu identificar cerca de 20% a mais de possíveis requisitos reutilizáveis.

Área Temática: Tecnologias da Web Semântica na Engenharia de Requisitos

Neste tópico são apresentadas algumas aplicações de tecnologias da Web Semântica na ER, exemplificando como elas podem apoiar as atividades de elicitação e de especificação de requisitos por meio da organização de conhecimento e de mecanismos para recuperação da informação para reúso. Todos os trabalhos selecionados atendem ao critério CI3.

Uso de *boilerplates* para elicitação de requisitos

Farfeleder *et al.* (2011) apresentam a implementação de um protótipo de sistema orientado por ontologias para auxiliar profissionais a obter requisitos usando uma representação semi-formal. O protótipo tem como base o conceito de *boilerplate* ou clichê, um *template* para requisito em que o usuário substitui certas palavras-chave pelos elementos que identificam o sistema que será construído. Por exemplo, estes dois requisitos:

O usuário deve ser capaz de visualizar o estado de suas operações pendentes.

O administrador deve ser capaz de visualizar o estado de todas as operações pendentes.

correspondem ao *boilerplate*:

O <stakeholder> deve ser capaz de <capacidade>.

que representa um requisito de capacidade com atributos *stakeholder* e *capacidade*. O próprio atributo *capacidade* é um atributo composto: Uma ação e um objeto visado pela ação. Isso permite a estruturação dos requisitos num conjunto de frases que podem ser depois processadas por ferramentas de processamento de linguagens.

Assim, o sistema infere e sugere requisitos estruturados de um determinado domínio de conhecimento com o auxílio de uma ontologia de domínio, cabendo somente ao Engenheiro de

Requisitos os procedimentos de validação dos mesmos. A Figura 42 apresenta o fluxo para a elicitação dos *boilerplates* de requisitos, a partir das informações da ontologia de domínio.

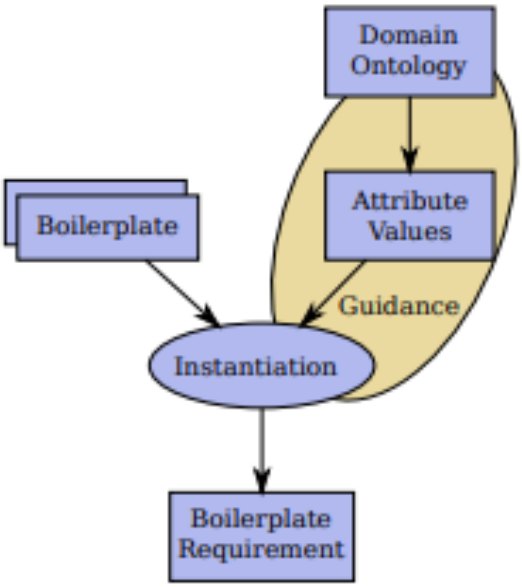


Figura 42. Fluxo para elicitação de *boilerplates* de requisitos.
Fonte: Farfeleder *et al.* (2011).

Para a instanciação, a ferramenta explora as relações e os axiomas da ontologia de domínio. Essas representam o conhecimento acordado das partes interessadas. Com isso, os autores esperam melhorar a precisão de requisitos. A Figura 43 apresenta os requisitos sendo sugeridos a partir dos *boilerplates* instaciados no fluxo. É possível perceber que o requisito descrito segue a estrutura do *template* obtido.

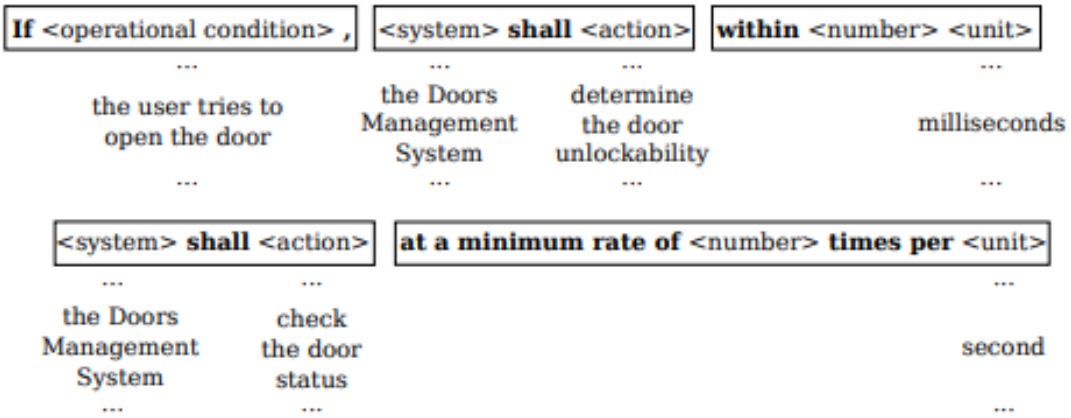


Figura 43. Sugestões de requisitos a partir dos *boilerplates*.
Fonte: Farfeleder *et al.* (2011).

A ferramenta conseguiu fornecer sugestões úteis na maioria dos casos, cerca de 85%. Como trabalhos futuros, pretende-se adicionar à ferramenta a conversão semiautomática de requisitos de linguagem natural em requisitos padrões.

Elicitação de requisitos orientada por ontologias de domínio

Motta *et al.* (2017) propõem um modelar para a redução do impacto de problemas existentes na comunicação entre *stakeholders* durante a elicitação de requisitos, gerando um impacto nos atributos de qualidade relacionados. Por meio de uma estrutura semântica, é possível verificar aspectos referentes à completude, consistência e não ambiguidade do documento de requisitos. É ressaltado que muitas vezes os atributos de qualidade são aplicados aos projetos de maneira informal, apesar do rígido controle recomendado pela ER, em decorrência da ausência de métodos e ferramentas que os suportem.

Este modelo proposto de ERC (Elicitação de Requisitos em Ciclos) aplica o método MAIA (Escutar, Pensar, Construir e Habitar) para o desenvolvimento de uma ontologia de domínio específica para cada projeto, que pode ser conectada a uma ontologia de domínio da organização a fim de construir um vocabulário comum e consensual para a especificação de requisitos, reduzindo os impactos da comunicação entre os envolvidos.

Os ciclos do método MAIA criam uma hierarquia de conceitos que auxiliam na análise da consistência e completude da informação da arquitetura de informação dos requisitos, permintindo a rastreabilidade. Os requisitos, elicitados por meio de técnicas tradicionais, irão compor tanto o documento de requisitos, quanto a ontologia de domínio específica, que permitirá a atividade de reuso. A avaliação do modelo foi documentada por meio de uma métrica baseada em *checklist*.A Figura 44 apresenta o Modelo ERC, que se baseia na elicitação de requisitos tradicional.

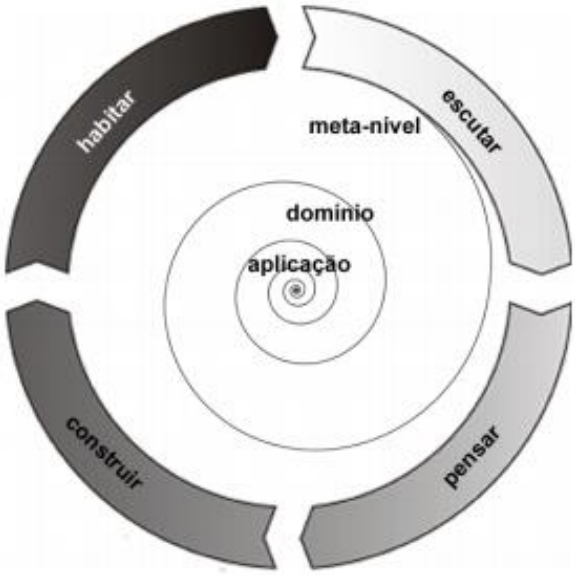


Figura 44. Modelo ERC.
Fonte: Motta *et al.* (2017).

A cada ciclo, há maior aprofundamento dos conceitos obtidos no ciclo anterior, partindo

o meta-nível até a camada de aplicação. Além disso, os momentos MAIA permitindo a aquisição de informação que será utilizada para a criação da ontologia de domíno. O requisito elicitado é mapeado ontologicamente, seguindo modelo de estrutura em triplas (sujeito, objeto e predicado), o que permite claramente identificar as partes que compõem o requisito, assemelhando-se ao uso de *boilerplates* proposto por Farfeleder *et al.* (2011). A Figura 45 apresenta algumas formas de estruturação do requisito na ontologia de domínio. A mesma pode ser aplicada tanto para RFs quanto para RNFs.

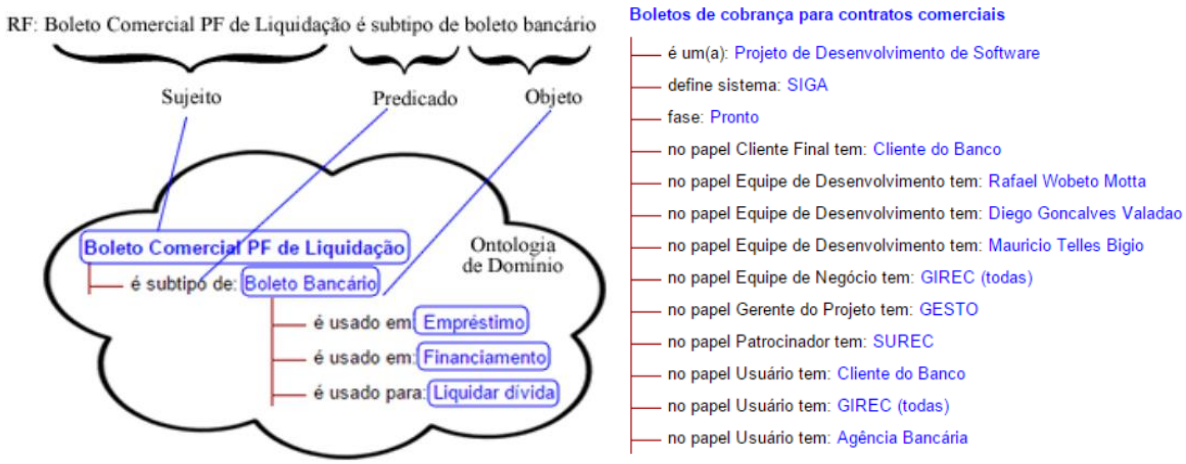


Figura 45. Estrutura de requisito na ontologia de domínio.
Fonte: Motta *et al.* (2017).

Os resultados avaliados da aplicação do modelo mostraram-se bastante promissores, com a criação de uma ontologia de domínio e de um ambiente de controle de qualidade dos requisitos, reduzindo os impactos dos problemas de comunicação. A proposta de compartilhamento de informações estruturadas em triplas ontológicas se mostrou viável e útil. O uso do MAIA auxiliou na reutilização e melhoria dos requisitos para outros projetos. Há a intenção de realizar novos testes em larga escala para avaliar a eficiência do método.

Uma Ontologia de Requisitos de Software

Nardi e Falbo (2006) apresentam o desenvolvimento de uma Ontologia de Requisitos de Software para a formalizar o conhecimento referente a um domínio de requisitos e, dessa forma, apoiar o desenvolvimento de ferramentas no ambiente ODE (*Ontology-based software Development Environment*). Os critérios para essa ontologia foram definidos com intuito de propiciar a reutilização desse conhecimento em diferentes ambientes relacionado à ER, bem como a integração com às demais ontologias do projeto ODE.

A Ontologia de Requisitos de Software foi elaborada com a utilização do método SABiO, que abrange as seguintes atividades: (i) identificação do propósito e especificação de

requisitos (questões de competência), (ii) captura da ontologia, que tem por objetivo capturar os conceitos, relações, propriedades e restrições relevantes sobre um determinado domínio; e (iii) formalização, que busca escrever os axiomas da ontologia em uma linguagem formal (optou-se pela lógica de primeira ordem).

Na primeira etapa, são elencadas as questões de competências para a atividade de especificação de requisitos, conforme Quadro 19.

Quadro 19. Conjunto de questões de competências para atividade de especificação de requisitos.

QC1. O que é um requisito?
QC2. Qual a natureza de um requisito?
QC3. Quais são os requisitos de um projeto de software?
QC4. Para que módulos do sistema um requisito é alocado?
QC5. Quais os responsáveis por um requisito?
QC6. Quais os interessados (stakeholders) em um requisito?
QC7. Qual a origem de um requisito?
QC8. Qual o estado de um requisito?
QC9. Como um determinado requisito é decomposto em outros requisitos?
QC10.Que requisitos são dependentes de um determinado requisito?
QC11.Que requisitos são conflitantes entre si?
QC12.O que é uma especificação de requisitos de software (ERS)?
QC13.Que artefatos descrevem, modelam ou implementam um requisito?
QC14.Como gerenciar mudanças nos requisitos e nos artefatos a eles relacionados?
QC15.Como avaliar a qualidade em requisitos?

Fonte: Nardi e Falbo (2006)

Para a segunda e terceira etapas do método SABiO, são identificados alguns aspectos relevantes com base nas questões de competência:

- Definição e Taxonomia de Requisitos (questões 1 a 4 e 12);
- Aprovação e Interesse em Requisitos (questões 5 e 6);
- Gerência de Requisitos (questões 7 a 11, 13 e 14);
- Qualidade em Requisitos (questão 15).

Para a definição e a taxonomia de requisitos, são definidos os axiomas que estabelecem as relações. Um dicionário foi elaborado para as definições dos conceitos que compõem a taxonomia. O Quadro 20 apresenta uma parte desse dicionário. Os termos em negritos são outros conceitos contidos no dicionário completo.

A aprovação e o interesse em requisitos é um processo que envolve várias pessoas na ER. A identificação dos interessados em cada requisito facilicita os procedimentos de negociação, esclarecimentos, discussão sobre impactos e validação. Nessa ontologia são criados axiomas que estabelecem relações de interesse e responsabilidade.

A gerência de requisitos é proporcionada por meio de axiomas que garantem a implementação de relações que mantenham a rastreabilidade entre os requisitos e permitam o controle de mudanças. Por fim, a qualidade é aferida com a definição de características de qualidade e com aplicação de métricas, podendo ser diretamente ou indiretamente mensuráveis.

Quadro 20. Parte do dicionário de termos da Ontologia de Requisitos.

Requisito	Representa especificações de serviços que o sistema deve prover, restrições do sistema e do processo de desenvolvimento e conhecimento necessário para a construção do sistema.
Documento de Especificação de Requisitos de Software	É um **documento**, portanto um **artefato** de software, formal que é utilizado para comunicação dos requisitos aos clientes, aos usuários etc, servindo de base para diversas **atividades** do **processo** de software. Sendo um documento, é normalmente associado a padrões organizacionais (**roteiros**) que definem a forma como deve ser produzido.
Tipo de Requisito	Conceito utilizado na caracterização de um requisito para retratar a dimensão que o requisito procura abordar, tal como funcional, não-funcional etc.
Módulo	É uma porção de sistema utilizada para organizar e agrupar, por exemplo, **artefatos**, **requisitos** e outros módulos que possuem certa afinidade.
Escopo	Determina os limites de um **projeto**, englobando o que faz parte do **projeto** e o que não faz. Um escopo pode ser decomposto em **módulos**.
Contexto	Caracteriza qual a situação na qual um **requisito** surgiu, procurando descrever quais as **pessoas** envolvidas, quais os **artefatos** analisados e em qual **atividade** do **processo** de software o requisito surgiu.

Fonte: Nardi e Falbo (2006).

Como trabalho futuro, é relatada a construção de uma ferramenta de apoio à ER baseada na ontologia proposta para ser integrada ao ambiente do projeto ODE.

Área Temática: Repositórios para Reúso de Componentes de Software

Nesta seção são apresentados alguns trabalhos que descrevem repositórios ou bibliotecas de reúso de componentes de software. Todos os trabalhos atendem ao critério CI4.

RCR Editor

Biblioteca de componentes reutilizáveis (RCR Editor) baseada em XML (*eXtensible Markup Language)* para *software* incorporado. Os componentes considerados para reúso no trabalho foram: controle de gestão, diagrama UML, diagrama de estrutura, código e linguagem de programação, além de restrições do sistema operacional. A biblioteca possui uma interface amigável, na qual os usuários podem incluir, editar e utilizar os componentes, como apresentado na Figura 46, que mostra um fluxograma operacional do trabalho da ferramenta. Na etapa final

é gerado um arquivo XML que contém os componentes reutilizáveis da biblioteca que podem ser acessados diretamente por várias *Application Programming Interface* (APIs) (CHANG *et al.*, 2011).

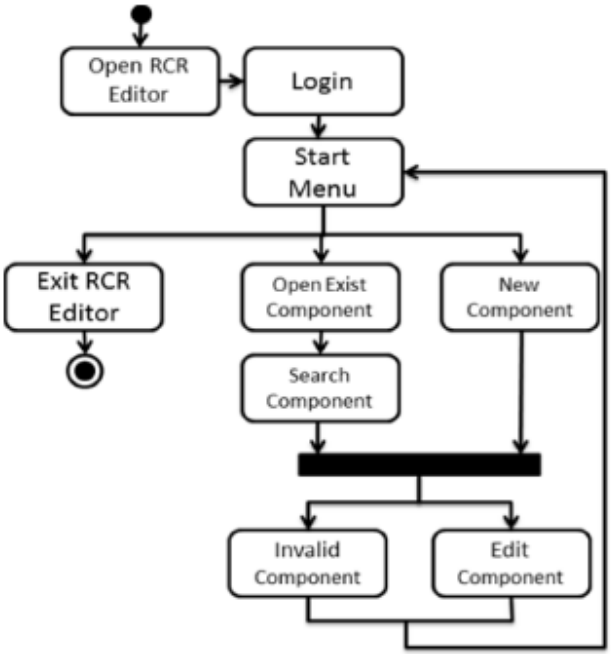


Figura 46. Fluxograma Operacional da RCR Editor.
Fonte: Change *et al.* (2011)

O mecanismo de busca utilizado na RCR Editor é executado por meio de palavras-chave, onde o usuário faz a consulta em um BD nativo do XML. Além disso, os próprios usuários podem adicionar componentes à biblioteca, para que o mesmo seja acessado posteriormente por outro usuário. No trabalho, os autores não citam como funciona a averiguação de qualidade na inserção desses novos componentes.

Brechó

A Biblioteca Brechó permite o desenvolvimento de extensões. De acordo com Santos *et al.* (2013), a Brechó é um sistema de informação web com uma base de aplicações, componentes, serviços, produtores/consumidores, com mecanismos de armazenamento, documentação, publicação, busca e recuperação.

A Brechó considera como componente os artefatos produzidos no desenvolvimento (processo, modelos, manuais, código, binário, testes etc.) e, assim, permite a aquisição de diferentes conjuntos de artefatos vinculados a licenças personalizadas e configuráveis. Ademais, a ferramenta também gerencia um mapa de reutilização dos componentes, que é estabelecido entre produtores e consumidores, corroboram Santos *et al.* (2013). A Figura 47, apresenta o fluxograma operacional da Brechó.

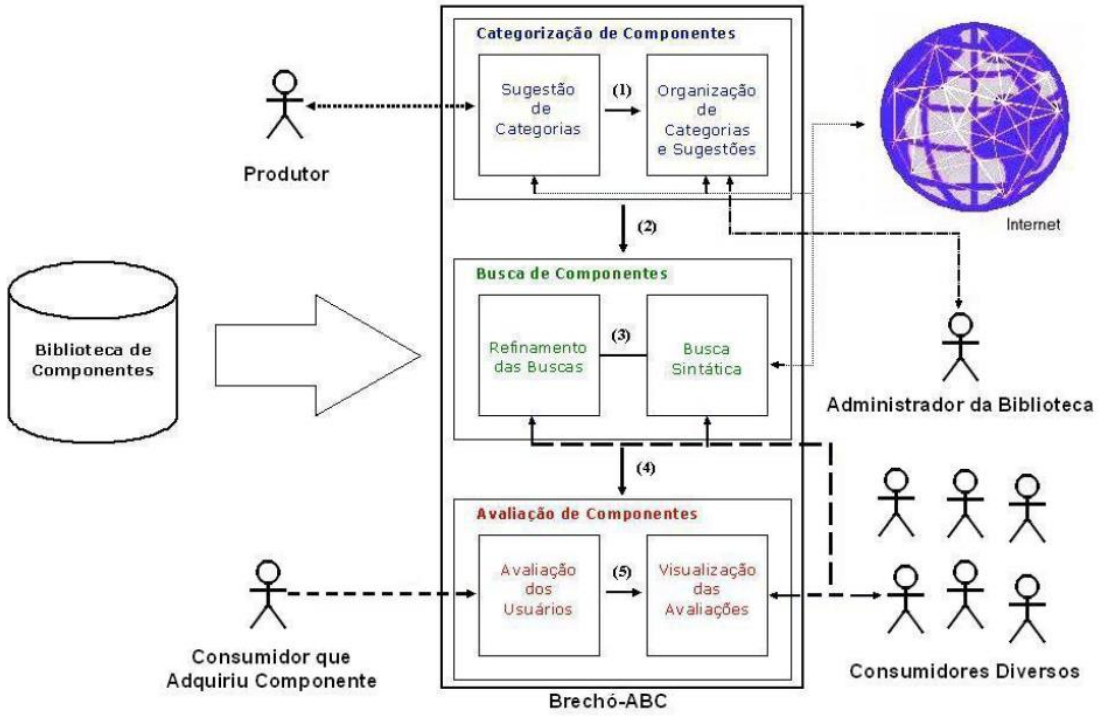


Figura 47. Fluxograma Operacional da Brechó.
Fonte: Raposo Junior (2007)

O mecanismo de busca utilizado na Brechó é baseado nos requisitos elencados abaixo, com o objetivo de oferecer opções de busca mais flexíveis e automatizadas, e assim, facilitar a pesquisa de componentes na biblioteca, aumentando a precisão dos resultados na busca, como salientam Santos *et al.* (2010).

- Limitação do espaço de Busca durante a pesquisa pelo componente;
- Permitir o uso de técnicas de pesquisa para potencializar a busca;
- Buscas textuais devem tratar possíveis casos de erros de digitação cometidos pelo usuário (Busca Sintática);
- Todas as fontes de documentação associadas ao componente, como versão e plataforma do componente, entre outros, devem ser utilizadas como fontes de pesquisa.

O mecanismo de avaliação dos componentes da Biblioteca é composto por dois módulos. O primeiro é relacionado com a avaliação dos usuários, ou seja, quem utiliza um componente reutilizável pode emitir a opinião a respeito do mesmo. Já o segundo módulo refere-se à disponibilização, na forma gráfica, o percentual de avaliações conforme cada uma das qualificações dadas pelos usuários sobre o componente. Além disso, também estão presente na biblioteca mecanismos relacionados à categorização dos componentes.

Hunter

A Biblioteca Hunter foi proposta pelos autores Wang *et al.* (2016). Esta ferramenta tem como objetivo a reutilização do código fonte da linguagem JAVA. Ademais, a biblioteca conta com a funcionalidade do alinhamento de interface e síntese do código reutilizável.

Como mecanismo de busca do código fonte, os autores utilizaram um hibridismo de métodos, no qual integraram o mecanismo do código Pliny[14], além da utilização de um BD com aproximadamente 12 milhões de métodos Java coletados de repositórios de código aberto, como GitHub[15] e Bitbucket[16].

Depois de executar a pesquisa de código, Hunter calcula uma matriz de semelhança de entre os resultados obtidos e a busca do usuário. Após essas etapas, a ferramenta Hunter tenta gerar uma implementação do método desejado, utilizando uma adaptação entre os resultados de pesquisa de código. Assim, visto que fora encontrado o alinhamento ideal entre a pesquisa do usuário e o código reutilizável, o algoritmo de reutilização do código Hunter utiliza o método AdapterGen (implementado pelos autores) para gerar automaticamente o código que será reutilizado. A Figura 48 apresenta o fluxograma operacional da ferramenta Hunter.

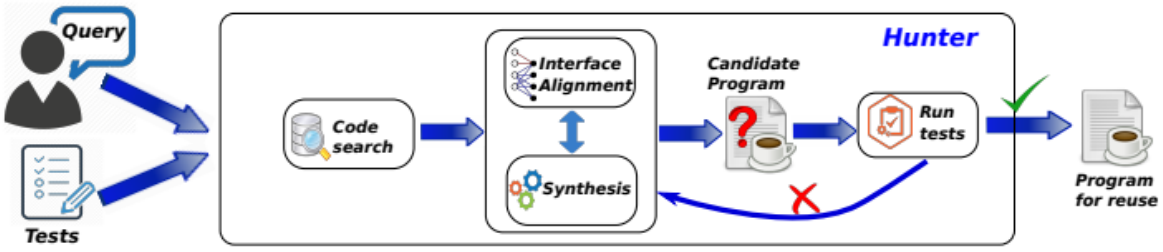


Figura 48. Fluxograma Operacional da Hunter.
Fonte: Wang *et al.* (2016)

No artigo da Hunter, não foi mencionado nenhum mecanismo de aferição de qualidade dos códigos reutilizáveis.

CodeEase

De acordo com Abid *et al.* (2017), CodeEase é uma ferramenta que tem como principal objetivo a reutilização de código fonte, e foi desenvolvida como um *plug in* para o Eclipse[17].

O mecanismo de busca está associado ao ambiente de desenvolvimento Eclipse, assim quando um usuário escreve ou edita linhas de código em um método, o CodeEase fornece recomendações de conclusão para o mesmo (*Reconmedation Engine)*, baseado na reutilização

[14] http://pliny.rice.edu/
[15] https://github.com/
[16] https://bitbucket.org/
[17] https://www.eclipse.org/

de código. Essa busca é realizada através da consulta em repositórios de código aberto, e um repositório local, o qual foi pré-populado com uma grande gama de códigos.

Para análise da igualdade dos métodos, são utilizadas duas abordagens: a primeira consiste na *Method Clone Structure (MCS)*, que é um grupo de métodos, que avaliam através de um algoritmo a possibilidade de projetos iguais ou diferentes em um repositório de código. A segunda abordagem refere-se ao *Clone Detection*, ou seja, um algoritmo que avalia através de parâmetros como tamanho do texto, palavras chaves, variáveis literais a possibilidade de o método ser o que o usuário procura.

A Figura 49 apresenta a arquitetura da CodeEase que representa a sequência das atividades mencionadas anteriormente.

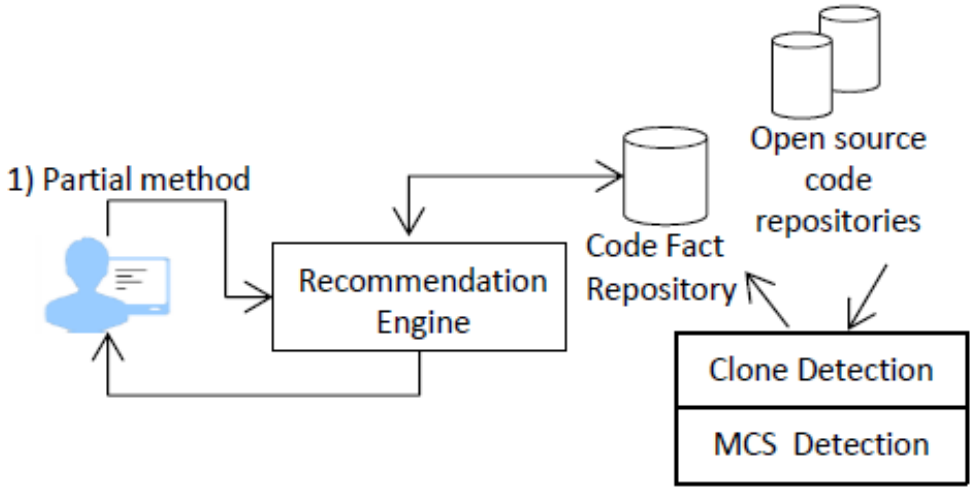


Figura 49. Arquitetura da CodeEase.
Fonte: Abid *et al.* (2017)

No artigo relacionado a ferramenta CodeEase, não foi mencionado nenhum mecanismo de aferição de qualidade dos códigos reutilizáveis.

APÊNDICE B – Grafos dos Tesauros Criados para o Protótipo

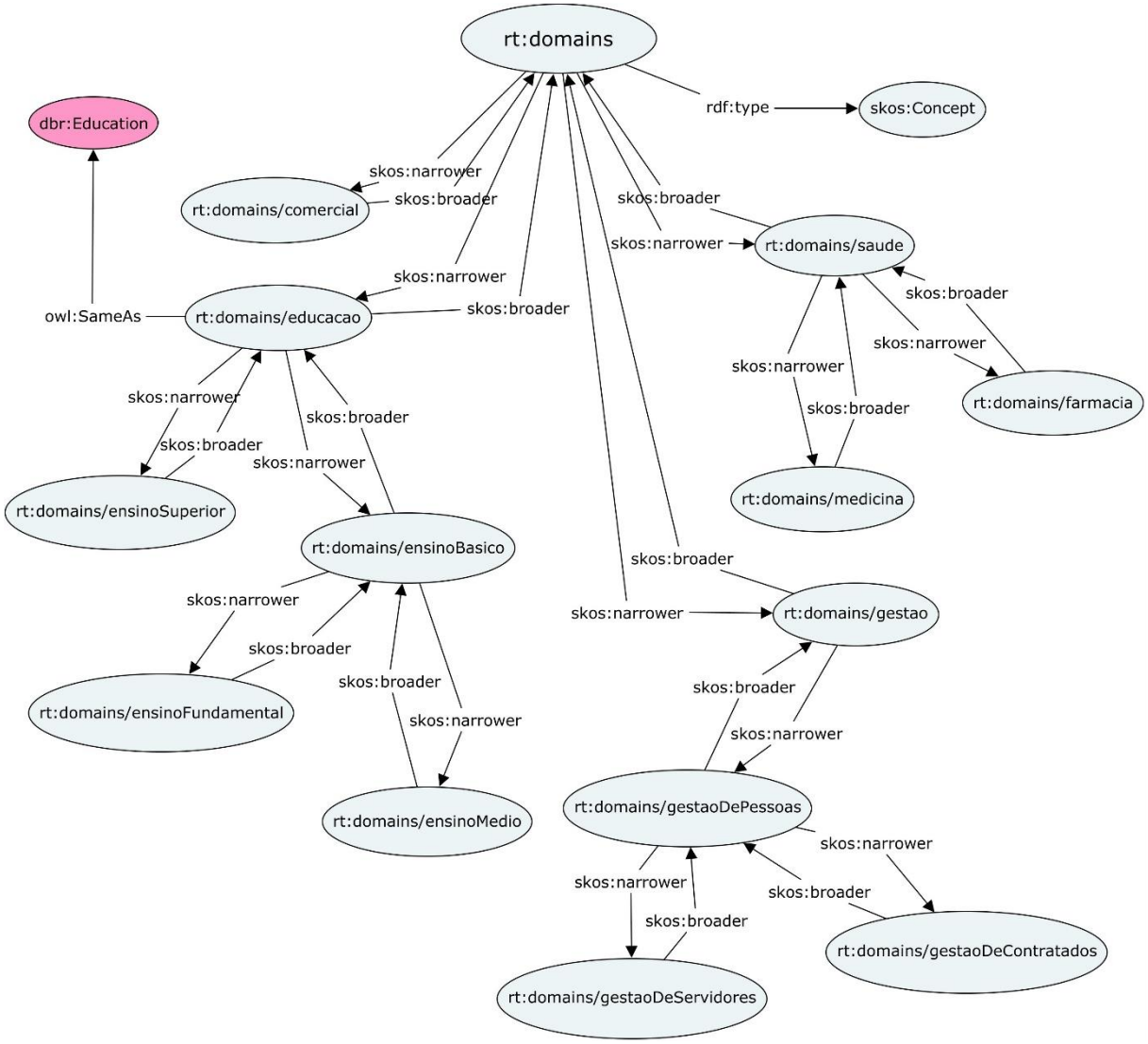


Figura 50. Grafo de conhecimento inicial do tesauro de Domínio.

Fonte: Elaboração Própria.

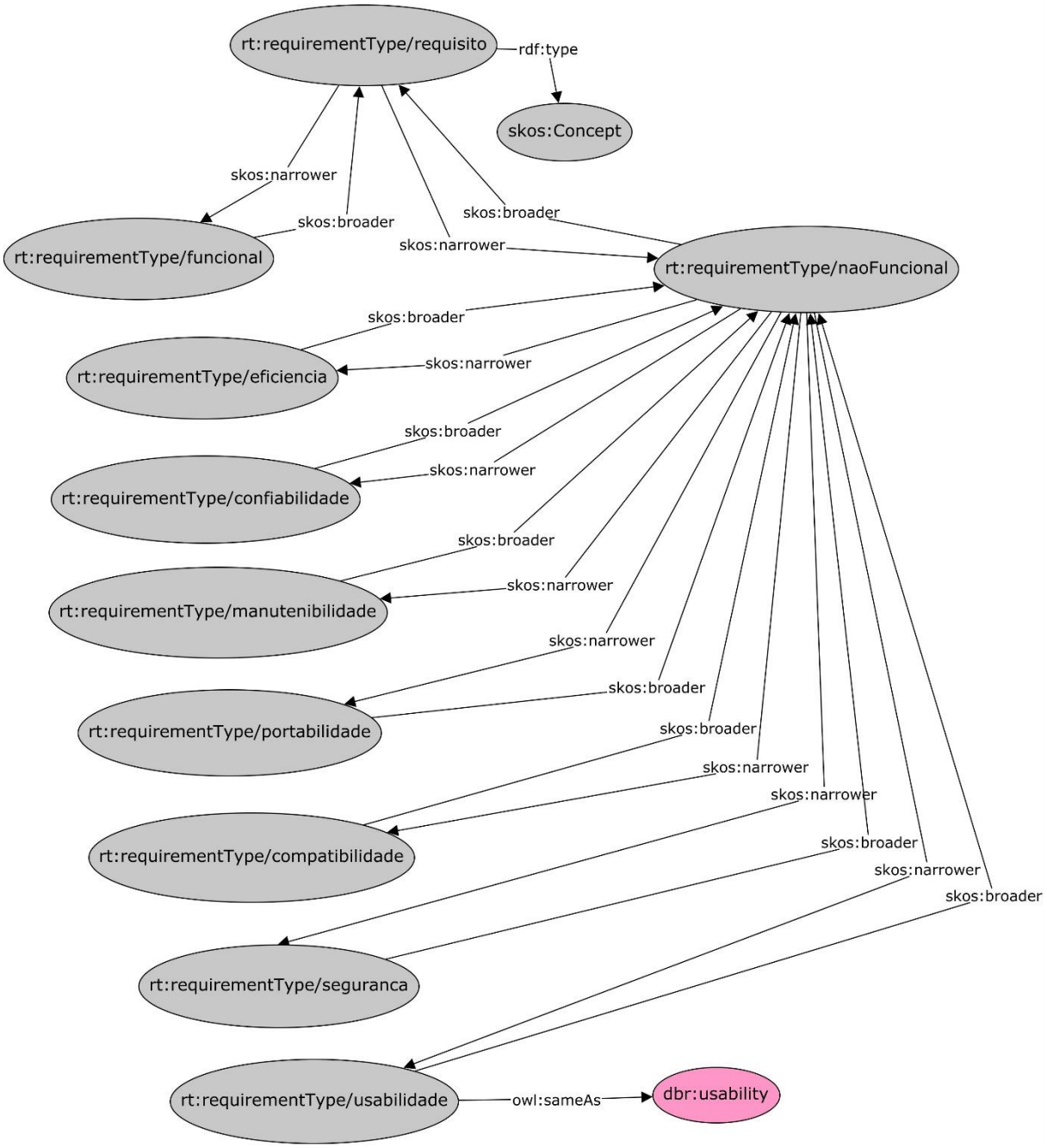


Figura 51. Grafo de conhecimento inicial do tesauro de Tipos de Requisito.

Fonte: Elaboração Própria.

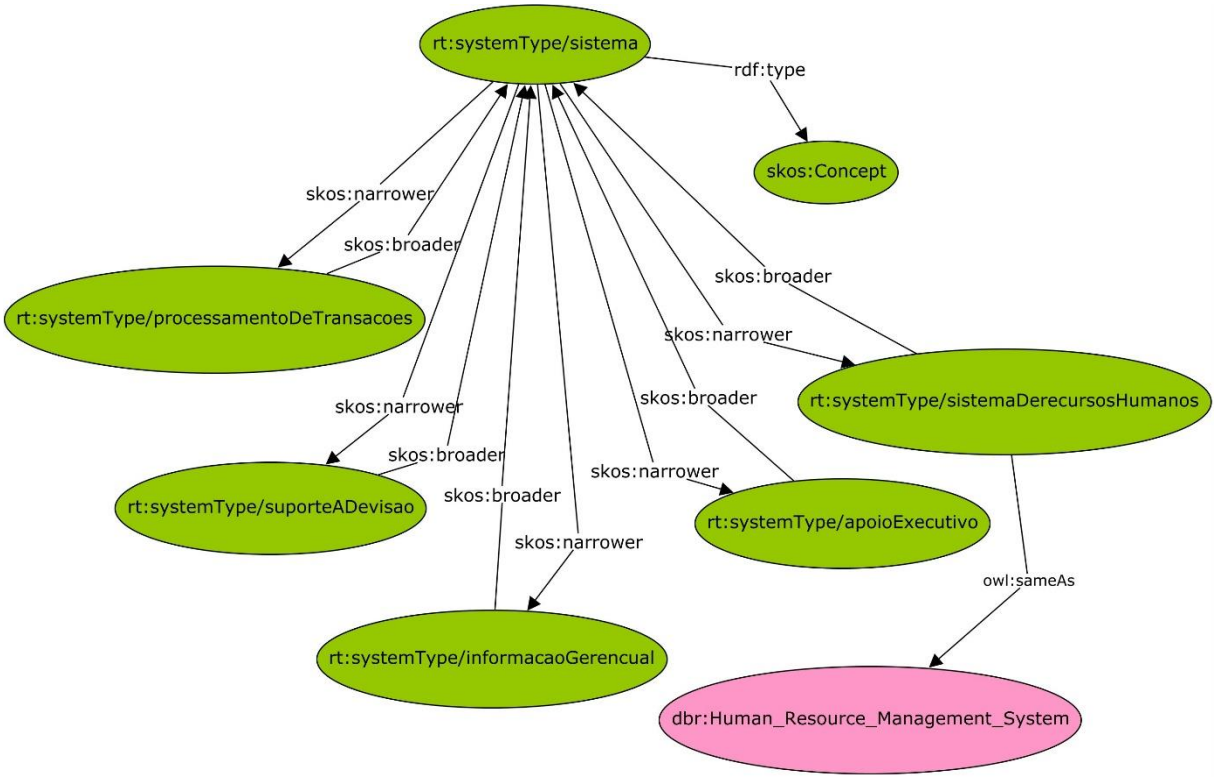


Figura 52. Grafo de conhecimento inicial do tesauro de Tipos de Sistemas.

Fonte: Elaboração Própria.

APÊNDICE C – Exemplo de Aplicação Cliente para o Web Service

Neste apêndice é apresentado um exemplo de aplicação cliente desenvolvida para consumir o web service construído e, com isso, possibilitar uma visão gráfica dos métodos para o estudo de viabilidade. Este exemplo representa como um dispostivo externo (computador, *tablet* ou *smartphone*) pode acessar os *endpoints* do web service para consumir os serviços disponibilizados por meio das requisitções HTTP, conforme apresentado na arquitetura do protótipo e ilustrado na Figura 53.

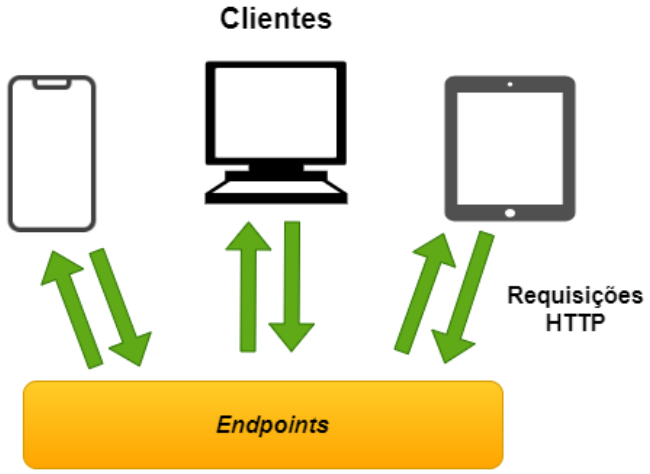


Figura 53. Alguns dispositvos que podem ter acesso aos serviços por meio de uma aplicação cliente.
Fonte: Elaboração própria.

Os requisitos são inseridos no web service por meio de uma tela de cadastro, cujo o *layout* foi construído com base no *template* proposto para especificação estruturada de requisitos, conforme a Figura 54. O usuário deve preencher os campos (atributos) com as informações do requisito e, no fim da tela, acionar o botão “Salvar”, que envia uma requisição HTTP do tipo POST ao servidor para armazenar o requisito.

Um detalhe nessa tela de cadastro é em relação à recuperação dos recursos para os atributos “Domínio”, “Tipo de Requisito”, “Tipo de Sistema”, “Requisito Generelista” e “Requisitos Específicos”, onde é utilizada *tag* <*select*> do HTML para exibir os recursos já existentes em cada categoria e, assim, estabelecer a conexão com outros tesauros, conforme apresentado na Figura 55.

A Figura 56 apresenta a tela que lista os requisitos cadastrados. A tabela traz o nome e o URI do requisito e permite, por meio de botões, detalhar o conteúdo do requisito (requisição do tipo GET) ou excluí-lo (requisição do tipo DELETE).

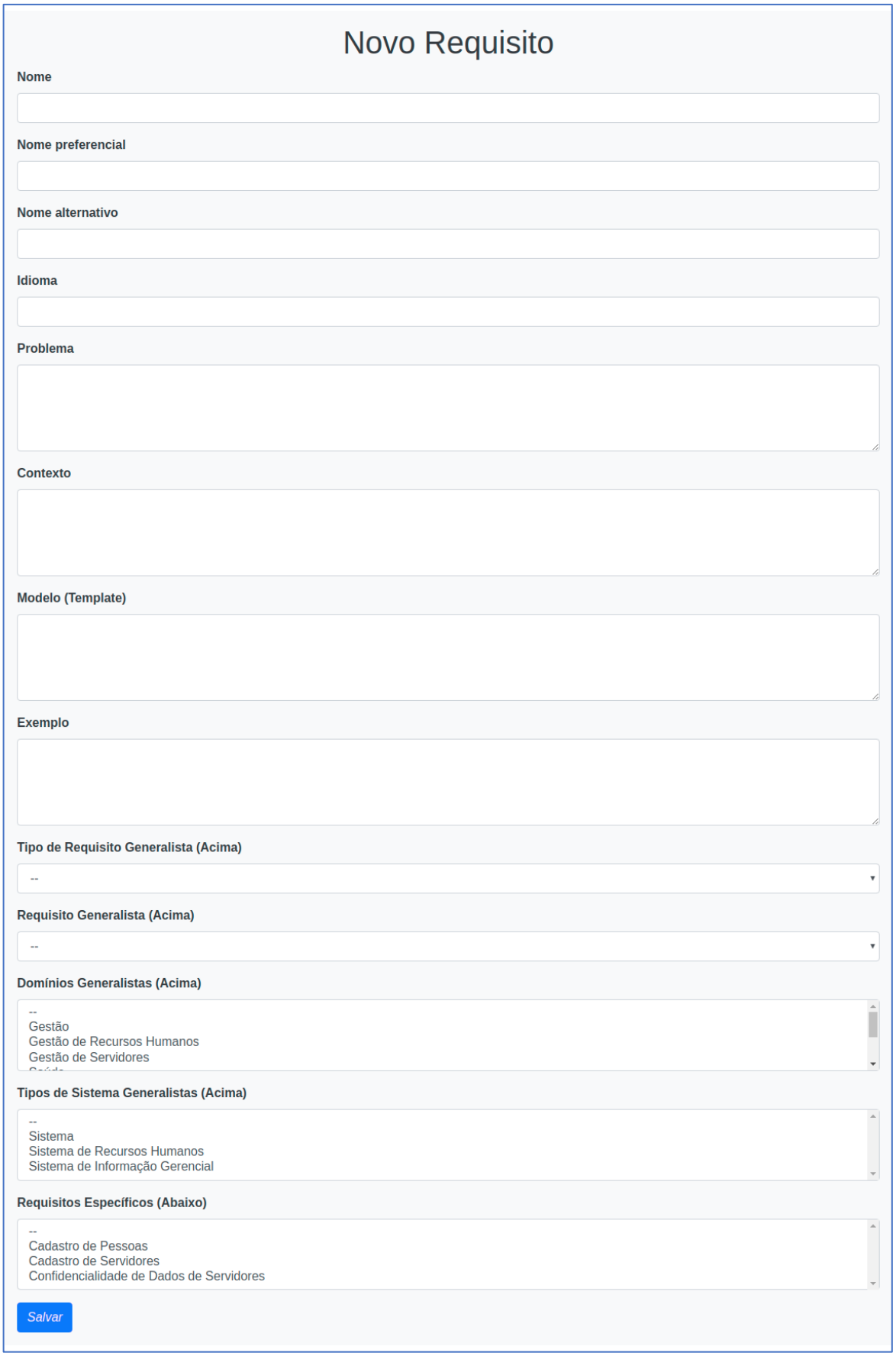


Figura 54. Tela para cadastro de requisitos no web service.
Fonte: Elaboração Própria.

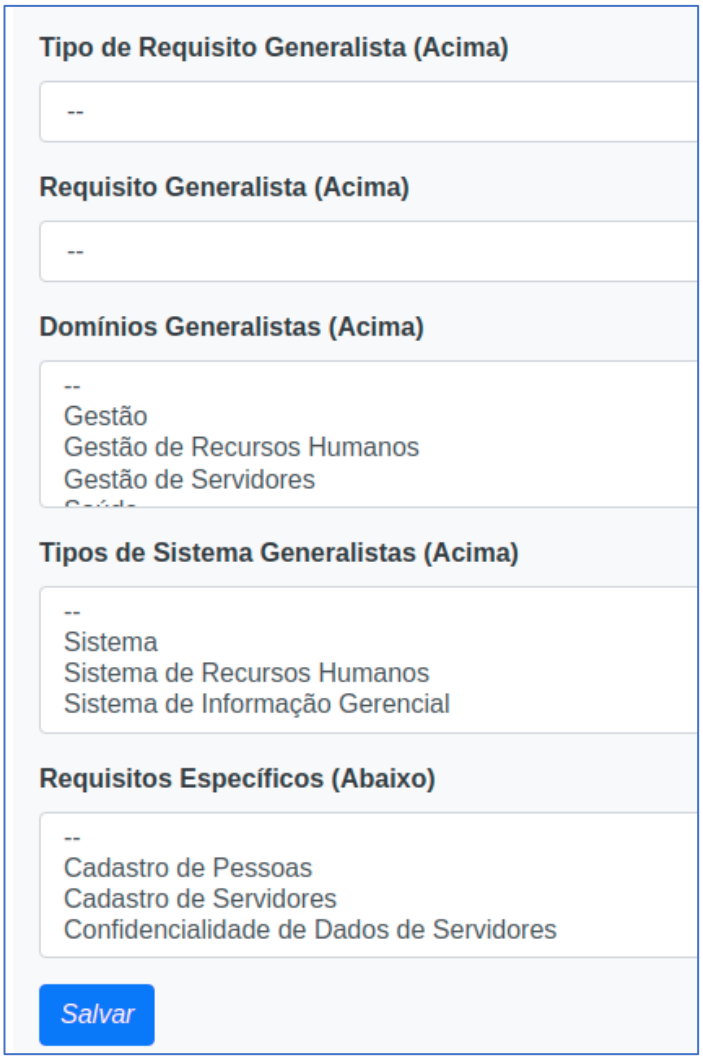


Figura 55. Uso da tag <select> do HMTL para recuperação e vinculação dos recursos cadastrados em outros tesauros.
Fonte: Elaboração Própria.

Figura 56. Tela de requisitos cadastrados no web service.
Fonte: Elaboração Própria.

O mesmo *layout* e as mesmas funcionalidades foram adotados para as telas de domínios (Figura 57), tipos de requisito (Figura 58) e tipos de sistema (Figura 59).

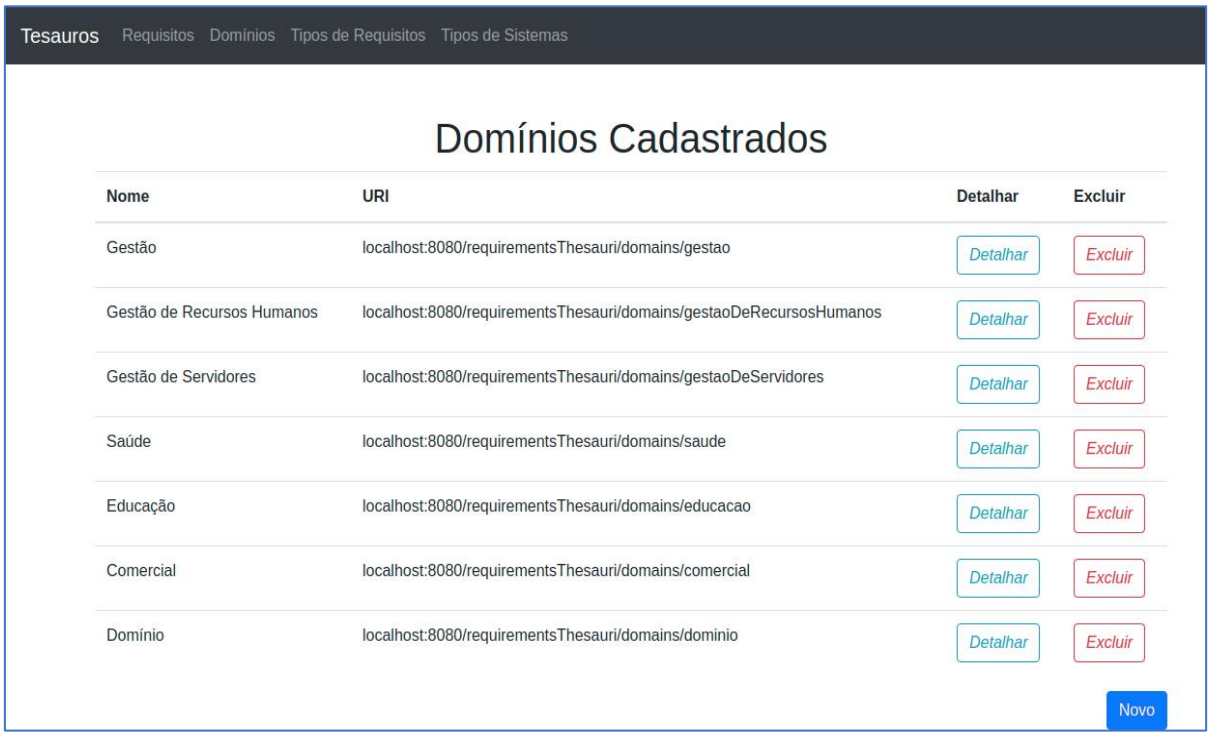


Figura 57. Tela de domínios cadastrados no web service.
Fonte: Elaboração Própria.

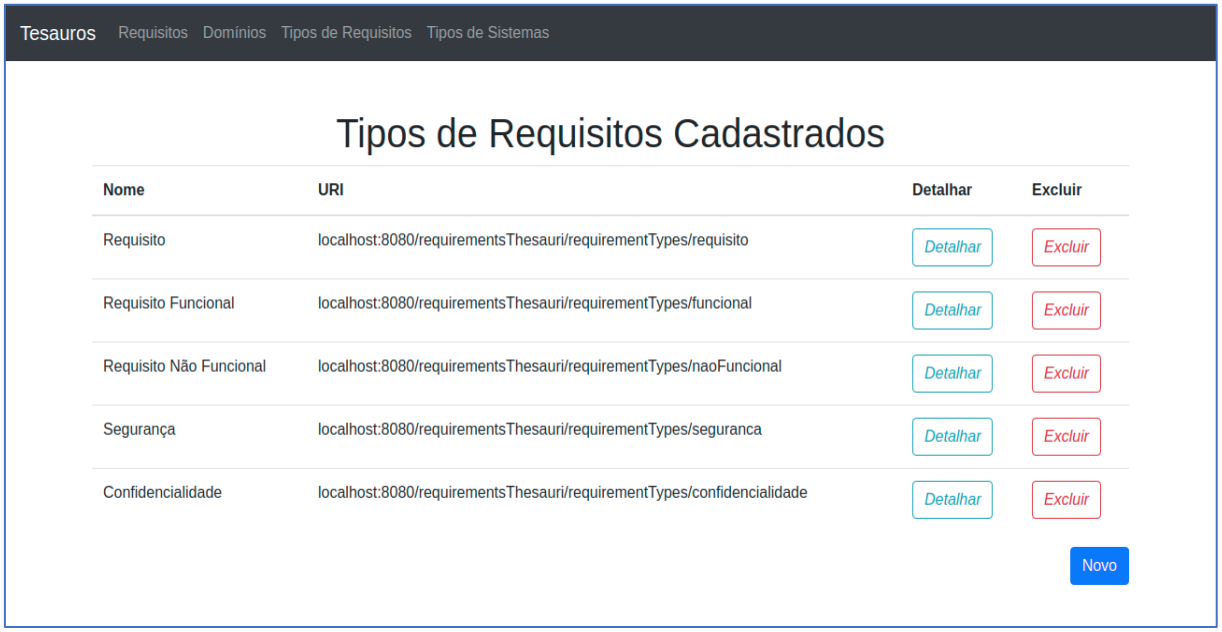


Figura 58. Tela de tipos de requisitos cadastrados no web service.
Fonte: Elaboração Própria.

Figura 59. Tela de tipos de sistema cadastrados no web service.
Fonte: Elaboração Própria.

Ao selecionar o requisito “Confidencialidade de Dados de Servidores” por meio do botão “Detalhar”, o usuário é redirecionado para uma nova tela, onde é possível visualizar os atributos do requisito, conforme Figura 60. O *layout* também é baseado no *template* de especificação de requisitos proposto.

Os atributos “Domínio”, “Tipo de Requisito” e “Tipo de Sistema” foram recuperados dos respectivos tesauros. O detalhe da inscrição “Acima” é para orientar o usuário sobre ponto de vista hierárquico desses recursos em relação ao requisito. É possível observar que não houve retorno de “Requisito Generalista” ou de “Requisitos Específicos”, o que significa que ainda não há nenhum tipo de vinculação entre o requisito “Confidencialidade de Dados de Servidores” e outros requisitos já armazenados.

Na frente de cada ainda há um botão “Acessar” que permite que o usuário acesse a respectiva tela de detalhes do recurso. Um exemplo é se o usuário escolher acessar o domínio “Gestão de Servidores”, o mesmo será direcionado para uma tela de detalhamento do domínio, conforme a Figura 61.

Nessa tela do domínio Gestão de Servidores, é possível perceber que tipos de recursos que foram recuperados. No caso “Domínio Generalista”, além de trazer o recurso principal “Gestão de Recursos Humanos”, a requisição retornou o URI pertencente à página da DBpedia que traz informações mais detalhadas sobre o domínio em questão, conforme destacado na Figura 62.

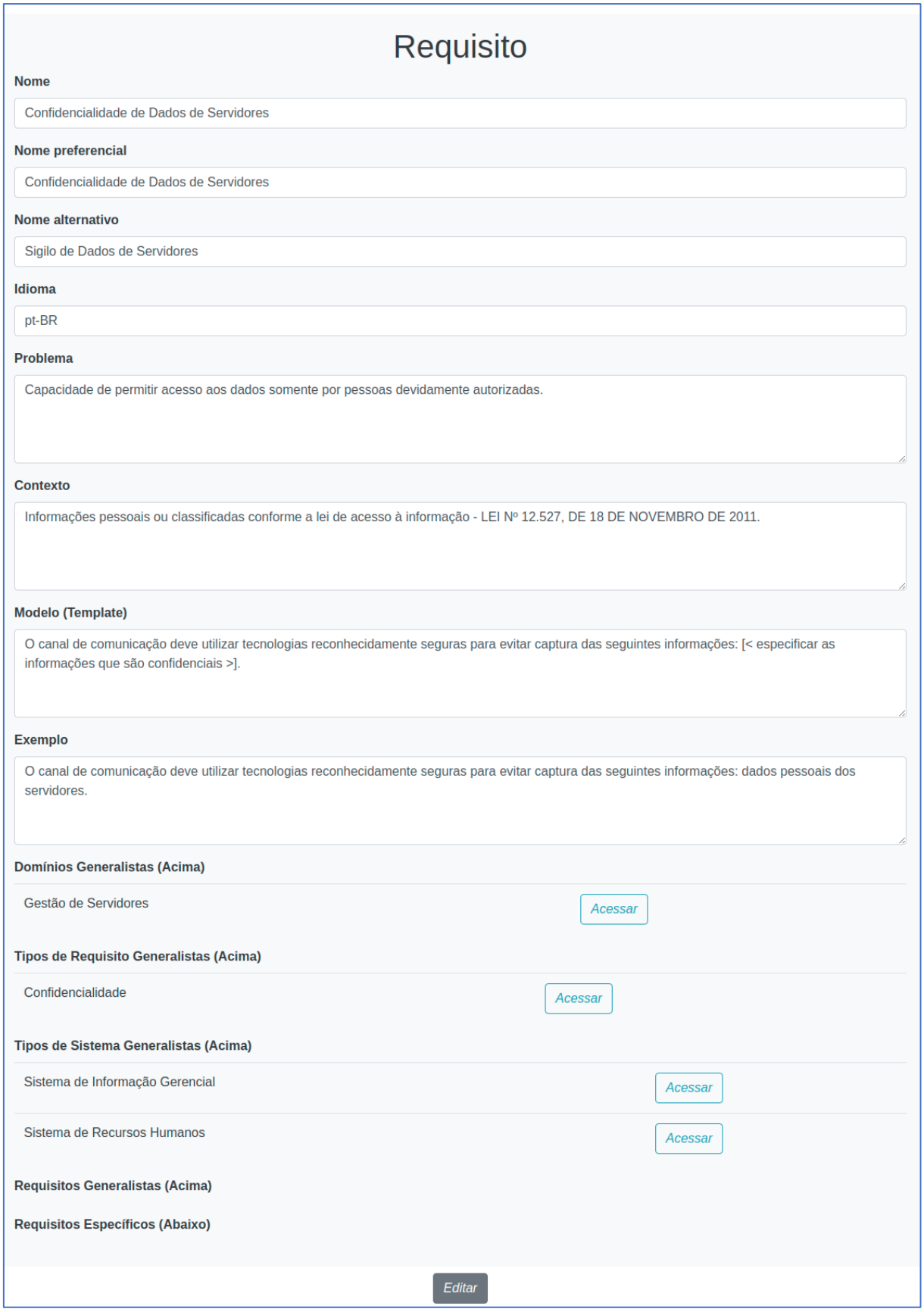


Figura 60. Tela de detalhamento do requisito “Confidencialidade de Dados de Servidores”.
Fonte: Elaboração Própria.

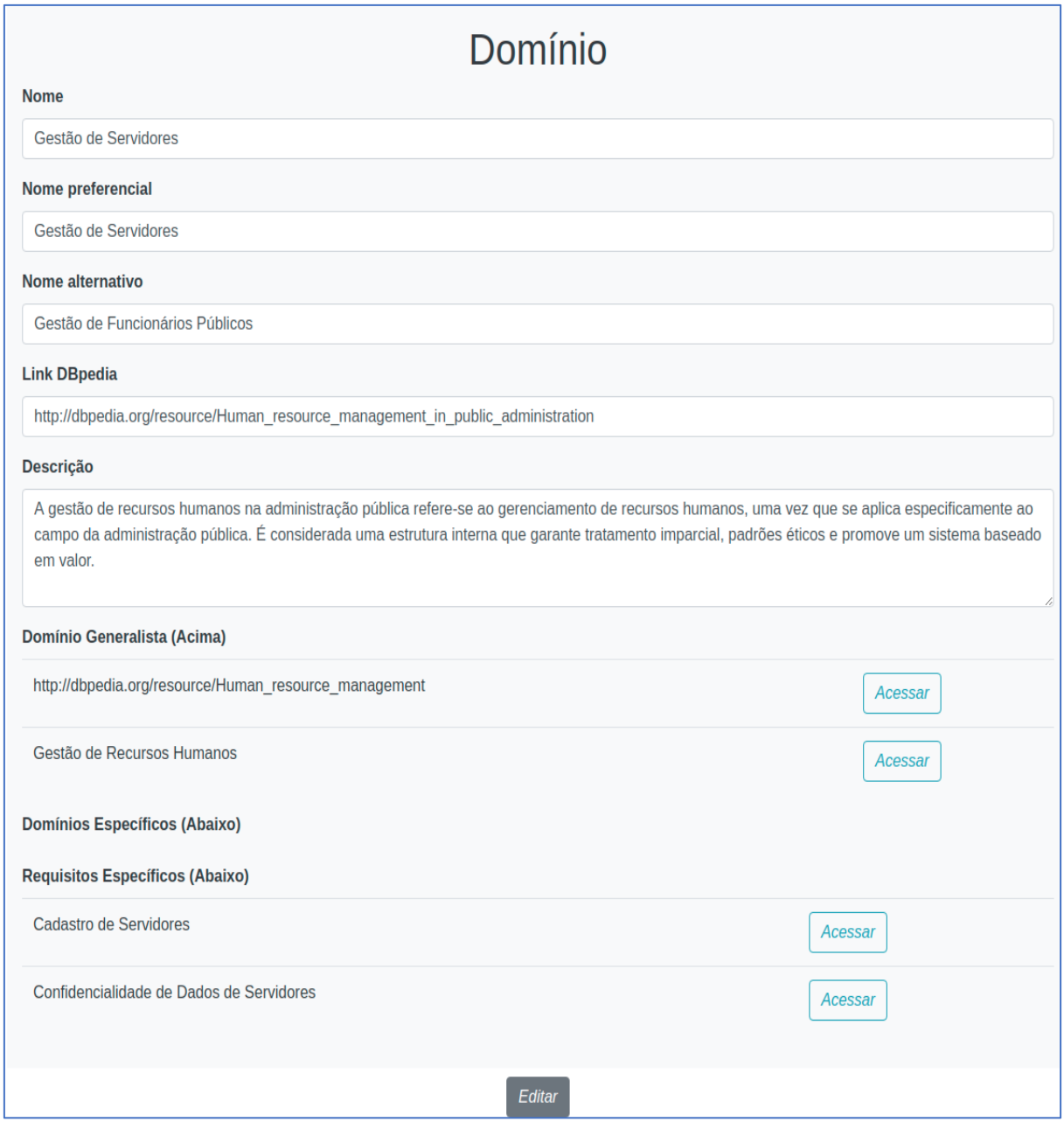


Figura 61. Tela de detalhamento do domínio “Gestão de Servidores”.
Fonte: Elaboração Própria.

Isso é possível por meio do raciocinador (*reasoner*) do *framework* Jena que realizou a inferência a partir da propriedade *owl:SameAs*, que considera que dois recursos diferentes possuem a mesma semântica, retornando, pois, os dois recursos como representantes do domínio “Gestão de Recursos Humanos”. Além disso, é mais uma evidência do quarto princípio *Linked Data*, ocorrendo o *manshup* com outras fontes de dados consagradas da web semântica.

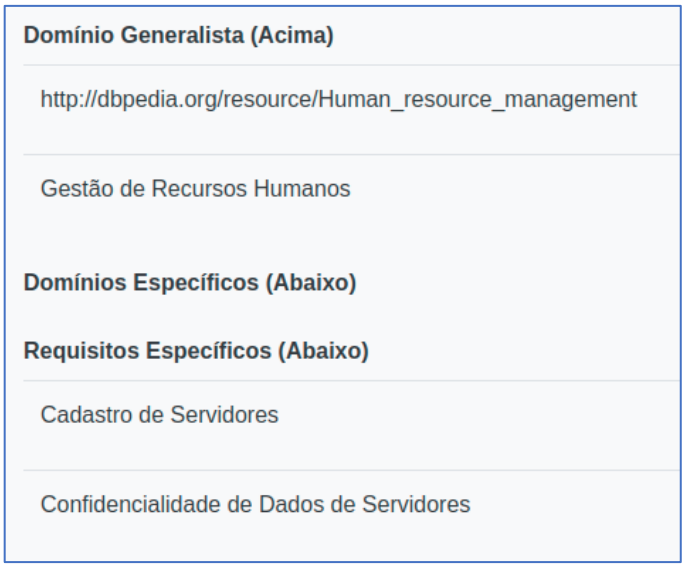


Figura 62. Recursos recuperados de outros tesauros vinculados ao domínios “Gestão de Servidores”. Fonte Elaboração Própria.

Ainda na Figura 62, é possível observar os “Requisitos Específicos”, sendo “Cadastro de Servidores” e “Confidencialidade de Dados de Servidores”. Nesse momento, o usuário pode acessar o novo requisito ou retonar ao anterior. Essas conexões auxiliam na navegabilidade entre os grafos dos recursos dos tesauros e atestam a rastreabilidade entre as informações, uma vez que é possivel ir no sentido de dados mais generalistas ou mais específicos de acordo com as necessidades do usuário. Além disso, ao conectar com diferentes recursos de outros tesauros, há um aumento no número de “caminhos para chegar” a um requisito, pois ele pode ser acesssado a partir de uma consulta a um domínio, tipo de requisito ou tipo de sistema.

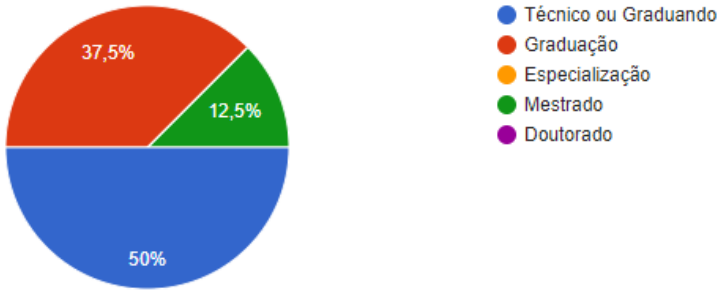
Por fim, cada tela de detalhamento possui um botão “Editar” que permite, além atualizar ou corrigir informações dos recursos, que novas relações possam ser estabelecidas. Com isso, novas inferências podem surgir a cada consulta, pois o racionador se torna mais preciso de acordo com o aumento no número de informações sobre as relações entre os recursos, à medida em que são cadastrados.

APÊNDICE D – Questionário do Estudo de Viabilidade

Perfil do profissional participante

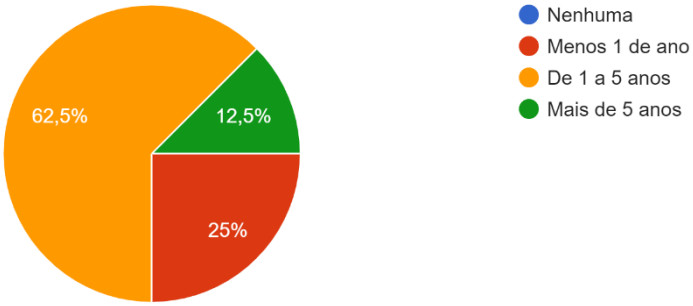
Formação acadêmica:

8 respostas



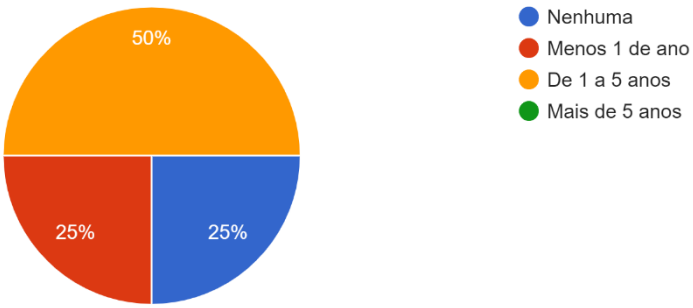
Experiência com projetos de desenvolvimento de software:

8 respostas



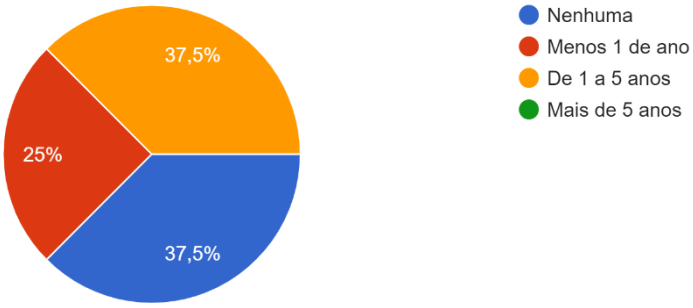
Experiência com levantamento e análise de requisitos:

8 respostas



Experiência com reúso de requisitos:

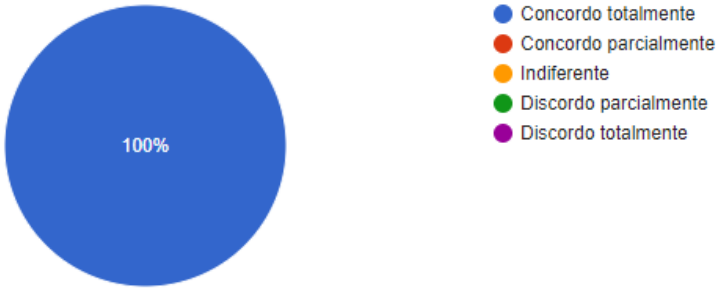
8 respostas



Avaliação do Protótipo

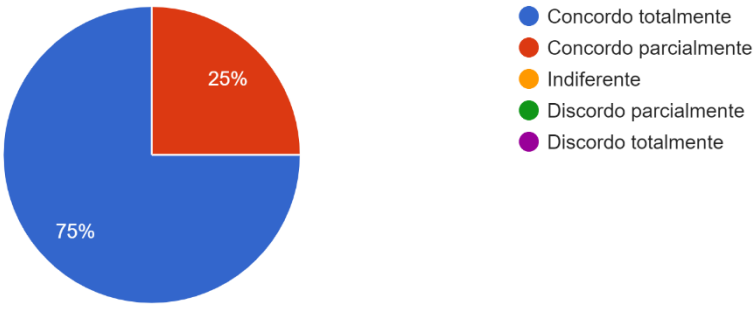
O levantamento e a análise de requisitos pode ser uma tarefa complexa para a elaboração de projetos de software em determinados domínios de conhecimento.

8 respostas



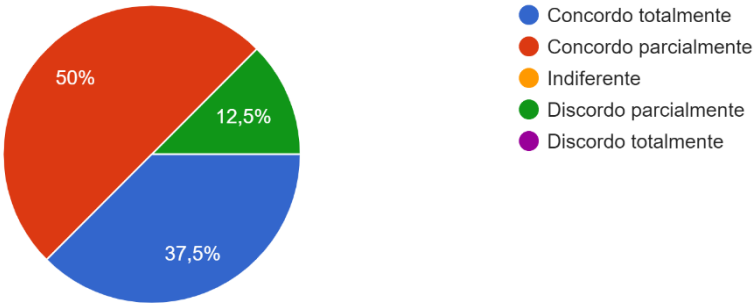
A especificação correta de requisitos é um dos fatores determinantes para o sucesso de um projeto de software.

8 respostas



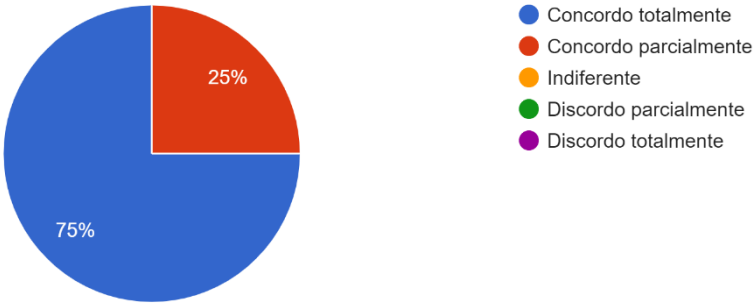
O reuso de requisitos auxilia no entendimento de um domínio de conhecimento.

8 respostas



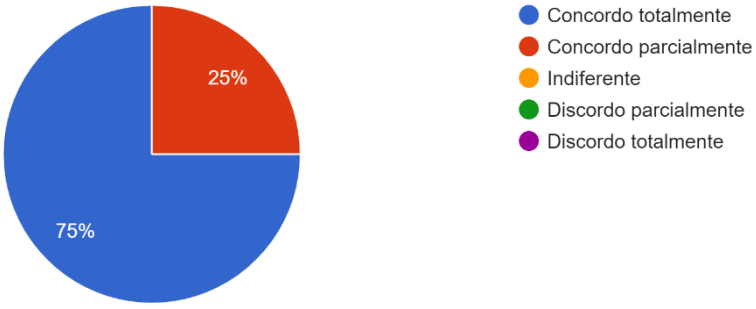
O reuso facilita o processo de levantamento e análise de requisitos em um projeto de software.

8 respostas



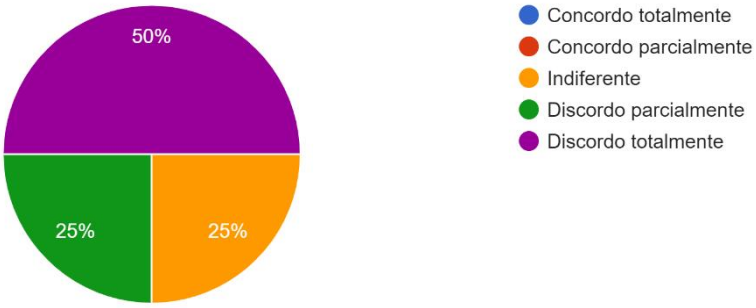
A especificação de requisitos de forma estruturada facilita a sua documentação para reuso.

8 respostas



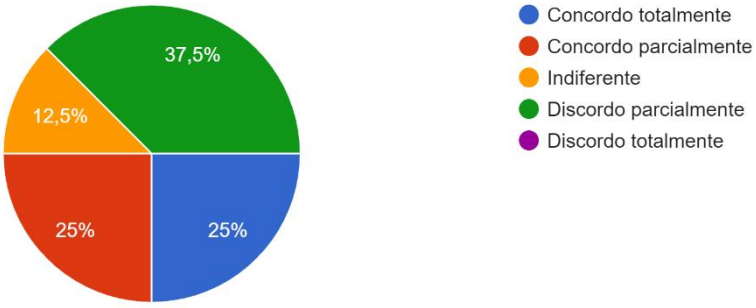
Não tenho dificuldades em organizar requisitos para reúso.

8 respostas



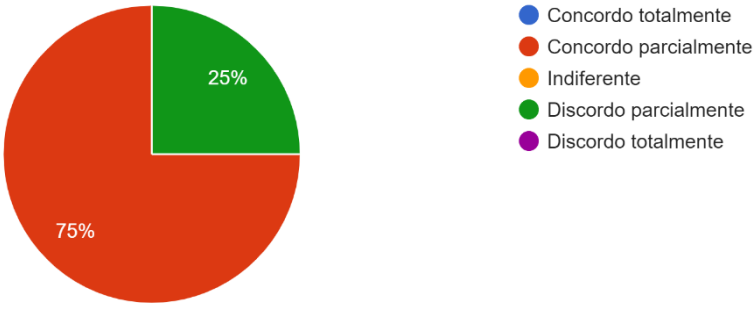
O conceito de tesauro é de fácil entendimento.

8 respostas



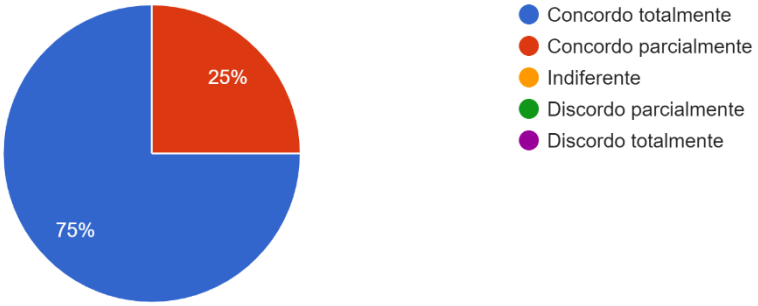
Organizar requisitos em forma de tesauros parece uma atividade complexa.

8 respostas



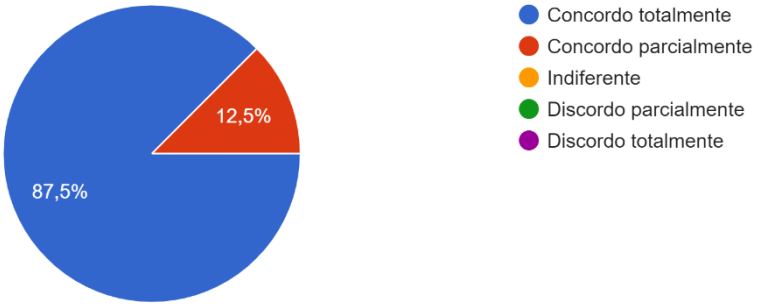
A hierarquia de requisitos (genérico e específico) obtida por meio de um tesauro facilita a sua localização para reuso em um domínio.

8 respostas



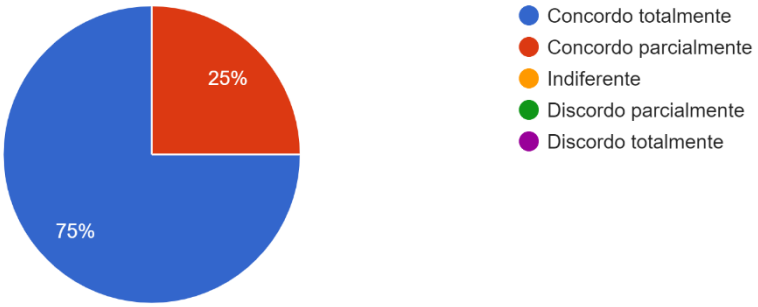
A utilização de recursos da web semântica pode potencializar e facilitar a atividade de reuso.

8 respostas



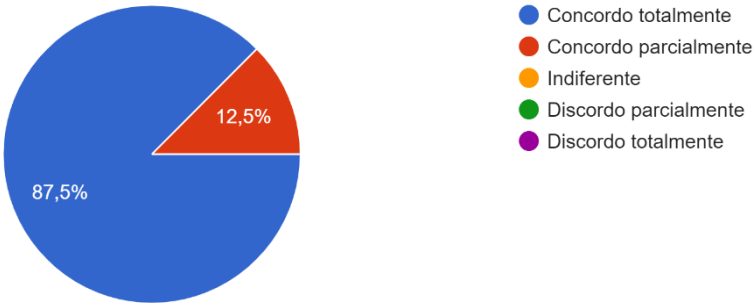
Permitir que uma máquina raciocine e realize inferências sobre os requisitos pode ser considerado um diferencial positivo para a Engenharia de Requisitos.

8 respostas



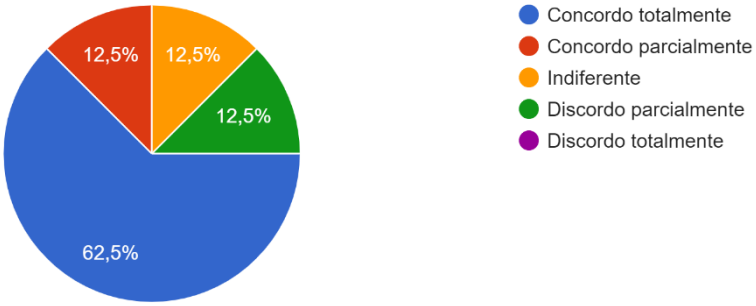
A estrutura de tesauro facilita a identificação da rastreabilidade de informações.

8 respostas



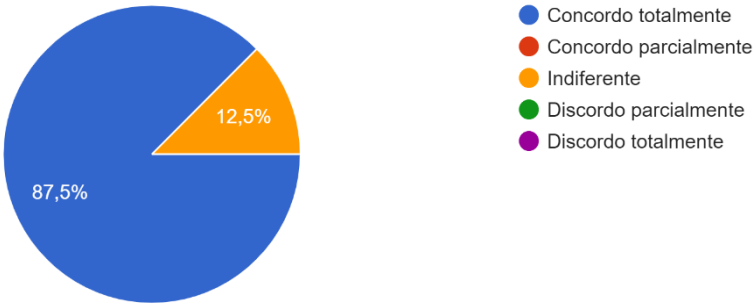
Considero o web service uma boa opção de repositório de conhecimento para reuso.

8 respostas

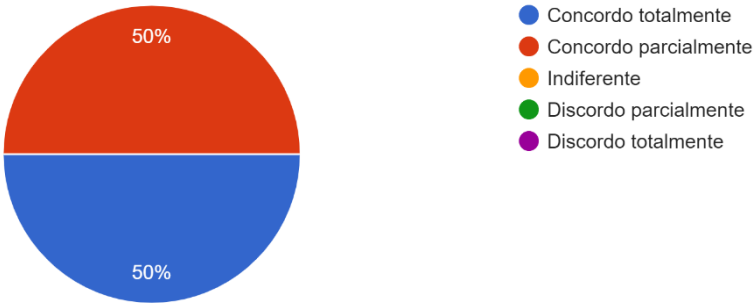


Os métodos REST inicialmente implementados no projetos são de fácil compreensão.

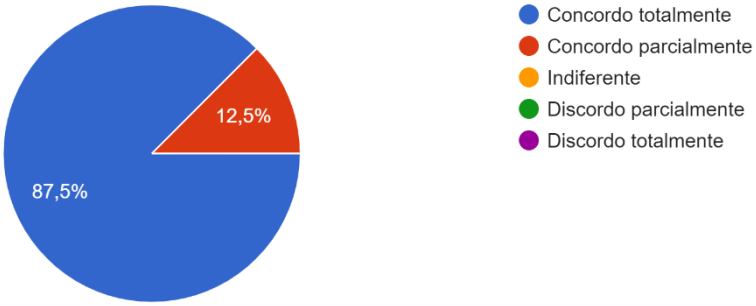
8 respostas



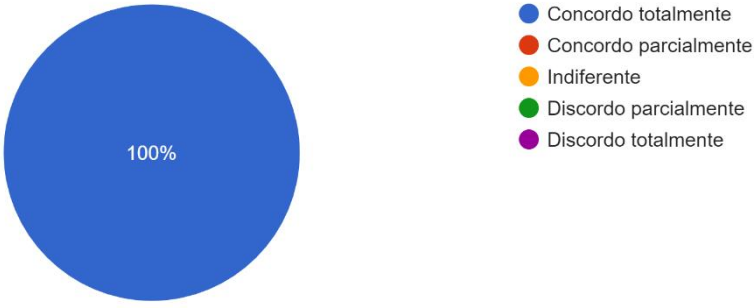
Os formatos de inserção e consulta aos requisitos são de simples compreensão e execução.
8 respostas



Por ser tratar de um web service, isso facilita a interoperabilidade e o acesso geograficamente distribuído, seja na Web ou numa Intranet, aos te...eúso do seu conteúdo de forma simples e eficiente
8 respostas



Acredito que há potencial de expansão do projeto apresentado.
8 respostas



Conhece algum outro mecanismo para reuso de requisitos de forma automatizada? Qual?

7 respostas

Não

Não

Quando é feito o levantamento de requisitos de sistemas, muitos possuem alguns requisitos em comum, como requisitos não funcionais relacionados a segurança dos mesmos. A maioria dos frameworks de desenvolvimento, já vem com esses requisitos implementados, se tratando então de um reaproveitamento de requisitos indireto. Esta é a única forma que eu conhecia até então.

Não.

não conheço

Não conheço.

Consegue assimilar algum ponto da proposta apresentada a algo já existente no mercado?

7 respostas

Não

A única consigo assimilar é em relação aos frameworks. Como na maioria dos casos desenvolvimento do software, acredito que se reutilize código, sem se considerar o ponto de se estar reutilizando requisitos.

Atualmente não vi nenhum projeto que tenha a proposta similar a esse.

não

Não sei exatamente o que já existe no mercado sobre esta área.

Consideraria a utilização do web service apresentado na sua rotina como fonte para reuso de requisitos? Se não, por quê?

8 respostas

Sim

A princípio não me parece interessante, pois eu iria precisar uma interface gráfica própria para poder operar os requisitos do tesauros. Um webservice só se torna interessante quando vou desenvolver um software que vai tirar algum proveito dos dados, nesse caso só imagino um software que atenderia aos próprios desenvolvedores de software, ou seja, algo que a própria ferramenta proposta no trabalho poderia fazer.

Sim.

sim

Não sei se reuso. Mas de obtenção das informações para integração com outras, por exemplo para BI, sim.

Sim, pois, apesar de não ter experiência com a área de estudo, me parece uma boa solução para reuso de requisitos.

Consegue identificar algum impedimento para a proposta?

7 respostas

Não

Não

O impedimento principal acredito que seja a familiaridade dos desenvolvedores de software com os conceitos de tesauros e websemântica. Ambos se tratam de áreas de predominância do ambiente acadêmico, e no âmbito comercial pode haver uma resistência no uso dos mesmos devido ao pensamento de alguns desenvolvedores acharem que isso irá "burocratizar" o processo todo.

A curva de aprendizado, seria mais um item para estudo num ramo que sempre há com que se atualizar e como é algo novo, se não for bem apresentado os profissionais optaram por darem foco em algo mais próximo do que algo novo.

não

Talvez um projeto tenha algum requisito que não possa divulgar (por tratar-se de segredo de uma empresa) para os usuários do web service.

Tem alguma sugestão de melhoria para o projeto?

8 respostas

Não

Fazer um manual ensinando como mexer e como implantá-lo, além de construir uma interface simples para uso visual. No futuro fazer com que a máquina trasse e interligue os requisitos, montando e indicando aos profissionais templates prontos com uma gama de requisitos que a máquina "ache" que o projeto apresentará.

Informar o tipo de projeto e ter um retorno dos requisitos relacionados a ele

Minha sugestão é uma ferramenta que possa funcionar de forma funcional por uma interface gráfica, mas ainda mantendo o web service como um possível potencializador do projeto (no qual desenvolvedores futuramente podem implementar suas próprias abordagens). Outra seria estudar formas de simplificar o processo, como poder identificar automaticamente o domínio dos requisitos de um determinado sistema através de uma breve descrição, e também identificar automaticamente requisitos relacionados.

* Estender usando ou criando ontologias de requisitos ligadas a diferentes metodologias de desenvolvimento de software;
* Verificar como seria a gerência de configuração dos requisitos e sua relação com as ontologias. Inclusive como seria o caso de requisitos que migram entre ontologias conforme o tempo.

Permitir que os utilizadores possam avaliar a qualidade dos requisitos exibidos pelo web service.

criação de uma biblioteca para poder, em certos projetos futuros, eliminar a necessidade de se acessar a plataforma via web