

UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

HENRIQUE VIGNANDO

**OntoExper-SPL: uma ontologia de apoio a experimentos de linha
de produto de software**

Maringá

2020

HENRIQUE VIGNANDO

**OntoExper-SPL: uma ontologia de apoio a experimentos de linha
de produto de software**

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Edson A. Oliveira
Junior

Maringá
2020

FOLHA DE APROVAÇÃO

HENRIQUE VIGNANDO

OntoExper-SPL: uma ontologia de apoio a experimentos de linha de produto de software

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação pela Comissão Julgadora composta pelos membros:

BANCA EXAMINADORA

Edson A. Oliveira Junior
Universidade Estadual de Maringá — DIN/UEM

Prof. Dr. Aline Maria Malachini Miotto Amaral
Universidade Estadual de Maringá — DIN/UEM

Prof. Dr. Katia Romero Felizardo Scannavino
Universidade Tecnológica Federal do Paraná — UTFPR

Aprovada em:

Local da defesa:

AGRADECIMENTOS

Agradeço a Deus por ter me abençoado nesta caminhada e as pessoas que contribuíram na realização deste trabalho. Em especial:

A minha família pelo carinho, apoio e incentivo.

Ao meu orientador professor Dr. xxxxxxxxxxxx pelo apoio, comentários e sugestões no desenvolvimento deste projeto.

Aos demais professores das disciplinas que cursei e aos colegas ...

E a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro concedido a este trabalho.

OntoExper-SPL: uma ontologia de apoio a experimentos de linha de produto de software

RESUMO

O processo de experimentação em Engenharia de Software (ES) tem se mostrado fundamental para o ciclo de vida de um software. Com ele é possível reduzir grandes esforços de desenvolvimento e principalmente de manutenção. A comunidade de ES vem discutindo e avaliando como melhorar a qualidade de experimentos de ES, visando aumentar a confiabilidade dos seus resultados. Por mais que se tem abordado a qualidade de experimentos, ainda há carência em contextos específicos, como é o caso de Linhas de Produto de Software (LPS). Assim, planejar, executar e analisar os resultados de um experimento de LPS torna-se crucial para apoiar a evolução de LPS e fornecer um corpo de conhecimento confiável e auditável. Nesse sentido este trabalho apresenta uma ontologia para apoiar experimentos de LPS, a OntoExper-SPL. A ontologia foi concebida com base em diretrizes pré-definidas, projetada usando a linguagem *Ontology Web Language* (OWL) e apoiada pelo ambiente Protégé. A OntoExper-SPL foi povoada com mais de 200 experimentos em LPS. A ontologia foi avaliada com base em um estudo de viabilidade com 17 especialistas em LPS e ontologia. Além disso, foi implementado um protótipo de um Sistema de Recomendação (SR) capaz de utilizar a ontologia para fazer inferências sobre os dados dos experimentos de LPS. Assim, acredita-se que a ontologia possa contribuir diretamente com uma melhor documentação dos experimentos de LPS, disseminar o conhecimento de experimentação em LPS, apoiar a cultura de experimentação na academia e na indústria e melhorar os projetos de software e execução de experimentos, aumentando a confiança do corpo de conhecimento visando a transferência de tecnologia para indústria.

Palavras-chave: Experimento de LPS. Linha de Produto de Software. Ontologia. Ontologia em Engenharia de Software. Sistema de Recomendação em Engenharia de Software. Sistemas de Recomendação.

OntoExper-SPL: an ontology for supporting software product line experiments

ABSTRACT

The process of experimentation in Software Engineering (ES) is fundamental to the life cycle of a software. It is possible to reduce major development efforts and mainly maintenance. The ES community has been discussing and evaluating how to improve the quality of the experiments, in order to increase the reliability of its results. As much as they have approached the quality of generally controlled experiments, there is as yet no evidence they are analyzing in specific contexts, such as Software Product Lines (LPS). In this case, there is still a lack of instrumentation and specific measurement of the quality of experiments in LPS. It is therefore necessary to provide a reliable and replicable body of knowledge in the context of LPS. Because of this importance, designing, executing and analyzing the results of an experiment in LPS becomes crucial to guarantee the quality of the experiments. In this sense we propose an ontology (SMartyOntology) for experiments in LPS, because it has hundreds of published experiments. The ontology is primarily designed based on defined guidelines and is designed using OWL language, supported by the Protégé environment for syntax checking and initial evaluation. The ontology was filled with more than 150 experiments in software product lines, assembled in a systematic mapping study. It is also the opportunity to investigate the elaboration of a recommendation system for experiments in LPS based on the information modeling structured by the proposed ontology. Therefore, this paper presents fundamental concepts for the elaboration of an ontology for experiments in LPS, and for the creation of a recommendation system for experiments in LPS. We believe that this ontology as well as the recommendation system can contribute to better document the essential elements of an experiment, thus promoting the replication, replication and reproducibility of the experiments. In which, it can bring quality to the experimental projects and results obtained through the recommended experiment. Ontologies and Recommendation Systems are well known in ES, it is believed to be possible to apply these theories to recommend experiments in LPS. We also present a feasibility evaluation of the proposed ontology. At the end we have a recommendation system that shows good results in recommendations for controlled experiments. It is also hoped with this project to contribute to the LPS community in order to improve the projects and execution of experiments, increasing the trust of the body of knowledge in order to transfer technology

to industry

Keywords: SPL Experiment. Software Product Line. Ontology. Ontology in Software Engineering. Software Engineering Recommendation System. Recommendation Systems.

LISTA DE FIGURAS

Figura - 1.1	Etapas da Metodologia de Desenvolvimento de Pesquisa	17
Figura - 2.1	Um modelo de Características	21
Figura - 2.2	<i>Framework</i> de Engenharia de LPS	22
Figura - 2.3	Conceitos Essenciais de um Experimento	23
Figura - 2.4	Visão Geral do Processo Experimental	25
Figura - 2.5	Estrutura de uma Ontologia	27
Figura - 2.6	Exemplo de uma Ontologia para o domínio: Destino de Viagem .	29
Figura - 2.7	Ontologias para formalizar o conhecimento em ES	30
Figura - 3.1	Grafo inicial da proposta de ontologia	40
Figura - 3.2	Modelo Conceitual Clusterizado	41
Figura - 3.3	Diagrama de classes para a modelagem da ontologia	43
Figura - 3.4	Definição da classe Experiment no Protégé	45
Figura - 3.5	Definição da propriedade de objeto documentation no Protégé .	46
Figura - 3.6	Definição da propriedade de dado nameSPLUsed no Protégé . .	47
Figura - 3.7	Grafo da modelagem da ontologia gerado pela ferramenta WbVOWL	62
Figura - 4.1	Frequência das Respostas dos Participantes	70
Figura - 4.2	Histograma da Somatória	72
Figura - 4.3	Correlação dos critérios de avaliação.	74
Figura - 5.1	Filtragem Colaborativa	86
Figura - 5.2	Passos de Construção para um RSSE	88
Figura - 5.3	Modelagem geral do SR como prova de conceito para OntoExer-LPS	90
Figura - 5.4	Árvore de diretórios da aplicação SR com Django Framework . .	92
Figura - 5.5	Tela do SR	94
Figura - 5.6	<i>Rating</i> Explícito no SR	96
Figura - 4.1	Instrumento de Avaliação pagina 1	142
Figura - 4.2	Instrumento de Avaliação pagina 2	143
Figura - 4.3	Instrumento de Avaliação pagina 3	144
Figura - 4.4	Instrumento de Avaliação pagina 4	145
Figura - 4.5	Instrumento de Avaliação pagina 5	146
Figura - 4.6	Instrumento de Avaliação pagina 6	147
Figura - 4.7	Instrumento de Avaliação pagina 7	148

Figura - 4.8 Instrumento de Avaliação pagina 8 149

LISTA DE TABELAS

Tabela - 2.1	Tipos de Ontologia	31
Tabela - 3.1	Resumo dos resultado da avaliação executada por OOPS!	59
Tabela - 4.1	Perfil dos Participantes do Estudo	69
Tabela - 4.2	Porcentagem de Frequência Amostral das Respostas dos participantes	70
Tabela - 4.3	Somatório das Resposta dos Especialistas	71
Tabela - 4.4	<i>Ranking</i> de correlação dos critérios de avaliação	75
Tabela - 4.5	Total de respostas positivas, neutras e negativas	77
Tabela - 5.1	Sumário de técnicas de recomendação em cada domínio	89
Tabela - 5.2	Cálculo para previsão de nota	98
Tabela - 5.3	Entidade <i>rating_rating</i> do Banco de Dados	99

LISTA DE SIGLAS E ABREVIATURAS

API: *Application Programming Interface*

AQ: Avaliação de Qualidade

CBF: *Content-based Filtering*

ES: Engenharia de Software

ESE: Experimentação em Engenharia de Software

ETL: *Extract, Transform, Load*

ID: Identificador Único

LPS: Linha de Produto de Software

MTV: *Model, Template, View*

MVC: *Model, View, Controller*

POO: Programação Orientado a Objetos

OWL: *Ontology Web Language*

RSSE: *Recommendation System in Software Engineering*

SGBD: Sistemas Gerenciador de Banco de Dados

SPARQL: *Protocol and RDF Query Language*

SUMÁRIO

1	Introdução	14
1.1	Contextualização	14
1.2	Motivação e Justificativa	15
1.3	Objetivos	16
1.4	Metodologia de Desenvolvimento	16
1.5	Organização do texto	17
2	Fundamentação Teórica	19
2.1	Considerações Iniciais	19
2.2	Linha de Produto de Software	19
2.3	Experimentação em Engenharia de Software	22
2.4	Experimentos de LPS	26
2.5	Ontologias	26
2.5.1	Ontologias para ES	29
2.5.2	Tipos de Ontologias	30
2.5.3	Metodologias para Construção de Ontologias	32
2.6	Trabalhos Relacionados	32
2.7	Considerações Finais	33
3	OntoExper-SPL: uma Ontologia para Experimentos de LPS	35
3.1	Considerações Iniciais	35
3.2	Concepção	36
3.3	Projeto	44
3.3.1	Modelagem com Protégé	44
3.3.2	Povoamento com Python	46
3.4	Exemplo de Aplicação	55
3.5	Avaliação Preliminar	56
3.6	Considerações Finais	60
4	Avaliação Empírica da OntoExper-SPL	63
4.1	Considerações Iniciais	63
4.2	Definição do Estudo	63
4.3	Planejamento do Estudo	64
4.4	Execução do Estudo	67
4.5	Análise dos Resultados	68

4.5.1	Perfil dos Especialistas	68
4.5.2	Frequência Likert	68
4.5.3	Teste de Normalidade	71
4.5.4	Teste de Hipótese	72
4.5.5	Correlação dos Critérios	73
4.6	Discussão dos Resultados	76
4.7	Ameaças à Validade	77
4.7.1	Ameaças à Validade Interna	77
4.7.2	Ameaças à Validade Externa	78
4.7.3	Ameaças à Validade de <i>Constructo</i>	79
4.7.4	Ameaças à Validade de Conclusão	79
4.8	Empacotamento e Compartilhamento	79
4.9	Propostas de Melhorias com Base na Discussão dos Resultados	79
4.9.1	Melhorias Apontada pelos Especialistas	79
4.9.2	Melhorias com Base em Armadilhas Identificadas na Ferramenta do OOPS!	82
4.10	Considerações Finais	82
5	Protótipo de um Sistema de Recomendação para OntoExper-SPL	84
5.1	Considerações Iniciais	84
5.2	Sistema de Recomendação	84
5.2.1	<i>Collaborative Filtering</i>	86
5.2.2	Sistemas de Recomendação em Engenharia de Software	86
5.3	Concepção	89
5.4	Projeto	90
5.4.1	Fase 1: O Projeto de SR Usando Django Framework	91
5.4.2	Fase 2: Carregamento da OntoExper-SPL no SR	93
5.4.3	Fase 3: <i>Ratings</i> dos Usuários no SR	95
5.4.4	Fase 4: Modelagem de Recomendação, Um Algoritmo para Filtragem Colaborativa	95
5.5	Banco de Dados	98
5.6	Ambiente de Desenvolvimento	99
5.7	Empacotamento e Compartilhamento	100
5.8	Considerações Finais	100

6 Conclusão	101
6.1 Contribuições	102
6.2 Limitações	102
6.3 Trabalhos Futuros	103
REFERÊNCIAS	104
A Apêndice A: Ontologias - Tipologias e Metodologias de Construção	110
A.1 Metodologias de Construção	110
A.1.1 Metodologia TOVE (Grüninger Fox, 1995)	110
A.1.2 Metodologia de Uschold e King - ENTERPRISE (USCHOLD e KING, 1995)	111
A.1.3 Methontology (GÓMEZ-PEREZ, FERNANDEZ-LOPES e VICENTE, 1996)	111
A.1.4 Método Kactus (BERNARAS, LARESGOTTI, CORERA, 1996) . .	112
A.1.5 Método Sensus (SWARTOUT et al., 1996)	113
A.1.6 Método 101 (NOY e GUINNESS, 2001)	113
A.1.7 Método CYC (REED, LENAT, 2002)	114
A.1.8 On-to-Knowledge Methodology (OTKM) (SURE, STAAB e STU- BER, 2003)	114
A.1.9 Metodologia UP for ONtology (UPON) (DE NICOLA, MISSI- KOFF e NAVIGLI, 2009)	115
A.1.10 Metodologia NeON (SUARÉZ - FIGUEROA, 2010)	117
A.1.11 Metodologia MFPFO (LIM, LIU e LEE, 2011)	118
A.1.12 Ciclo de Vida de Schiessl e Bräscher (2011)	118
B Apêndice B: Código Fonte da OntoExper-SPL	120
C Apêndice C: Código Fonte do Povoamento de Indíviduos na OntoExper-SPL	133
D Apêndice D: Questionário de Avaliação e Respostas Aplicado à OntoExper-SPL	141
E Apêndice E: Código Fonte dos Exemplos de Aplicação de Recomendação Usando OntoExper-SPL	150
E.1 Arquivo <code>settings.py</code>	150
E.2 Arquivo <code>views.py</code>	152
E.3 Arquivo <code>collaborativeFilter.py</code>	154

Introdução

1.1 Contextualização

Experimentação em Engenharia de Software (ES) tem se tornado fundamental para desenvolver e melhorar métodos e ferramentas, bem como melhorar os processos de manutenção de software (Kitchenham et al., 2007). Essa discussão permite que o conhecimento seja gerado de forma sistemática, disciplinada, quantificável e controlada (Wohlin et al., 2012a). Dessa forma, melhorando-se a qualidade dos experimentos¹ pode ser obtido um corpo de conhecimento confiável e referente em um dado tópico de pesquisa.

Atualmente, não há diretrizes específicas para avaliar a qualidade de experimentos em ES especialmente. Para áreas emergentes e em processo de consolidação como é o caso de Linha de Produto de Software (LPS), em que aspectos específicos do domínio como, por exemplo, os artefatos utilizados como objetos experimentais, a complexidade do treinamento, a dificuldade de seleção de participantes qualificados e a falta de repositórios, podem influenciar os experimentos. Além disso, tem-se percebido uma constante carência de documentação adequada dos experimentos que acabam por inviabilizar a repetição e auditoria dos estudos em LPS (Furtado, 2018).

Para permitir a repetição, reprodução e a replicação de experimento é necessária uma formalização dos principais conceitos. Para isso, pode-se obter vantagem do uso de ontologias.

Ontologias estão entre os métodos mais utilizados para formalizar informações. Ontologias são representações formais de uma abstração contendo definições formais de

¹Neste trabalho usaremos o termo "experimento" para denotar ambos os conceitos de "experimento" e "quasi-experimento".

nomenclatura, conceitos, propriedades e relações entre os conceitos, a fim de definir um vocabulário controlado de termos e relações de conceitos (Gruber, 1993). Assim, o uso de ontologias para representar informações sobre experimentos permite padronizar dados facilitando a interoperabilidade, a troca de informações e a replicação de experimentos.

Por mais atraente que seja, uma solução geral de ontologia para experimentos de ES é muito esparsa. Dadas as particularidades de cada tópico de pesquisa em ES, seria complexo projetar uma ontologia capaz de representar todas as características particulares de cada tópico. Dessa forma este trabalho, concentra esforços no campo de LPS, por causa da ampla gama de dados de experimentos neste assunto e experiência do grupo de pesquisa em experimentação de LPS da UEM (Furtado, 2018).

Experimentos no campo de LPS requerem considerável experiência em ES, pois durante o planejamento e execução dos experimentos vários artefatos específicos de ES e LPS podem ser gerados. Assim, a curva de aprendizado é longa o suficiente para extrair e apresentar os resultados de experimentos de LPS satisfatoriamente (Furtado, 2018) e (Furtado e OliveiraJr, 2019).

Portanto, a definição de uma ontologia para experimentos de LPS se destina a apoiar pesquisadores e profissionais no desenvolvimento e replicação de experimentos, além de auxiliar na auditoria e validação de experimentos baseados em diretrizes determinadas pela ontologia.

Além disso, dados formalmente representados permitem o desenvolvimento de sistemas especialistas, como sistemas de recomendação. Um sistema de recomendação para experimentação ES poderia ser útil de duas maneiras: (i) didaticamente, para ajudar alunos, professores e profissionais a pesquisar experimentos relacionados; e, (ii) ajudar os profissionais da Engenharia de Software Experimental (ESE) a planejar e executar um experimento seguindo a experiência de outros, baseando-se em consultas e recomendações de experimentos correlacionadas à sua área de atuação.

1.2 Motivação e Justificativa

Realizar um experimento em LPS exige alguns pontos de atenção específicos para garantir a qualidade do experimento. Esses pontos foram investigados no trabalho de Furtado (2018). Nesse trabalho foram elaboradas diretrizes para determinar a qualidade de experimentos em LPS. Essa tarefa possui um árduo trabalho para garantir que, aspectos específicos do domínio como, por exemplo, os artefatos utilizados, que são, os objetos experimentais, a complexidade do treinamento, a dificuldade de seleção de participantes qualificados em LPS e a falta de repositórios de LPS, não influenciam nos experimentos

ao ponto de invalidá-los. A falta de experimentos com qualidade afeta diretamente a possibilidade de repetição dos estudos em LPS.

Sabendo que para realizar um experimento em LPS com qualidade exige-se seguir alguns modelos e diretrizes, construir uma modelagem formal por meio de uma ontologia para representação desse conhecimento adquirido pode proporcionar maior facilidade ao desenvolvimento dos mesmos, incentivando a cultura de desenvolvimento de experimentos na academia e indústria, seguindo um modelo formal de estrutura do conhecimento sobre experimento em LPS. Sendo assim, surge como uma oportunidade de pesquisa responder a seguinte questão: **Como formalizar o conhecimento de experimentação em LPS?**

1.3 Objetivos

Esta pesquisa tem como objetivo geral especificar uma ontologia que representa formalmente o conhecimento adquirido do estado da arte e do prático sobre experimentos de LPS. Nomeada **OntoExper-SPL**, uma ontologia para apoiar experimentos em LPS.

Os objetivos específicos desta dissertação são:

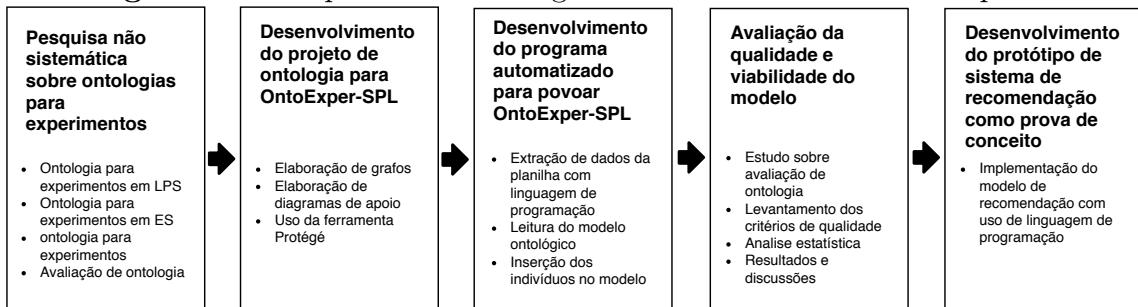
- gerar e representar um conjunto de metadados a partir de informações sobre experimentos de LPS;
- definir um modelo de ontologia para experimentos de LPS; e
- avaliar empiricamente a ontologia proposta.

1.4 Metodologia de Desenvolvimento

Para o desenvolvimento deste trabalho foi necessária uma pesquisa exploratória para definir um modelo de ontologia para experimentos de LPS. Para tal: foi realizada um pesquisa não sistemática sobre ontologias para experimentos, foi desenvolvido um projeto de ontologia (OntoExper-SPL), foi criado um programa automatizado para povoar a mesma, foi realizada uma avaliação da qualidade e viabilidade do modelo, e em seguida, foi elaborado um protótipo de sistema de recomendação como prova de conceito. A Figura - 1.1 apresenta as etapas da metodologia segundo neste trabalho, descritas a seguir:

- **Pesquisa não sistemática sobre ontologias para experimentos:** foram realizado estudos não sistemáticos para encontrar trabalhos relacionados descrevendo ontologias para experimentos de ES e avaliação de ontologia. Alguns estudos encontrados serviram de apoio à essa pesquisa, estão descritos na Seção 2.6;

Figura 1.1: Etapas da Metodologia de Desenvolvimento de Pesquisa



Fonte: O Autor

- **Desenvolvimento do projeto de ontologia:** trata do desenvolvimento da OntoExper-SPL, que incluiu realizar a escolha das tecnologias e ferramentas de construção de ontologia, como por exemplo o Protégé, envolver os *stakeholders* no projeto com experiência no domínio de experimentação, selecionar abordagens e modelos de ontologias relacionados, prototipação e diagramação do projeto;
- **Desenvolvimento do programa automatizado para povoar OntoExper-SPL:** refere-se ao processo de desenvolvimento do programa automatizado de povoamento da ontologia. Para isso foi necessário o uso de ferramentas, bibliotecas e linguagens de programação. Foram mais de 200 experimentos encontrados na literatura. Em seguida foi executado o programa de extração de dados, povoando a ontologia por meio dos metadados mais o programa de extração;
- **Avaliação da qualidade e viabilidade do modelo:** refere-se ao estudo empírico, com objetivo de avaliar a qualidade da OntoExper-SPL. Para isso foi elaborado um questionário com oito critérios de qualidade, enviado para especialistas responderem, e posteriormente realizada uma análise estatística das respostas, bem como pontos de melhorias encontrados durante a avaliação; e
- **Desenvolvimento do protótipo de sistema de recomendação como prova de conceito:** após processar os indivíduos na OntoExper-SPL foi elaborado um protótipo de sistema de recomendação utilizando os modelos de recomendação *Collaborative Filtering*, além de *frameworks* e linguagens de programação.

1.5 Organização do texto

Este capítulo apresentou a contextualização desta dissertação, a motivação e a justificativa, objetivos e metodologia. O restante da dissertação está estruturado da seguinte

forma: o Capítulo 2 apresenta a fundamentação teórica sobre LPS, experimentos em ES, qualidade de experimentos em ES e ontologias; o Capítulo 3 apresenta uma ontologia para experimentos de LPS - OntoExper-SPL, destacando desde a concepção ao povoamento da ontologia proposta e pré-análise de pontos de falha da ontologia; o Capítulo 4 apresenta uma avaliação empírica sobre a qualidade da OntoExper-SPL por especialistas da área; o Capítulo 5 apresenta um sistema de recomendação para experimentos de LPS, usando como modelo de dados a OntoExper-SPL; e o Capítulo 6 apresenta a conclusão acerca deste trabalho e as suas contribuições, limitações e trabalhos futuros.

Fundamentação Teórica

2.1 Considerações Iniciais

Este capítulo apresenta conceitos importantes, necessários para a compreensão desta pesquisa como LPS, experimentos de ES, ontologias e trabalhos relacionados. O conceito de LPS é fundamental para entendimento dos termos elaboradas na OntoExper-SPL, pois sua construção se baseia nas características que determina uma LPS. Entender experimentação em ES é imprescindível, pois é o cerne deste trabalho e toda modelagem da ontologia proposta foi criada a partir dos elementos experimentais descritos neste capítulo. O conceito de ontologia dirige solução apresentada, de modo que a base fundamental apresentada neste capítulo serviu diretamente para elaboração da OntoExper-SPL.

2.2 Linha de Produto de Software

Uma Linha de Produto de Software (LPS) é um conjunto de produtos que endereçam a um determinado segmento de mercado ou missão particular (Clements e Northrop, 2002). Esse conjunto de produtos também é denominado família de produtos, no qual os membros desta família são produtos específicos gerados a partir da reutilização de uma infraestrutura comum, denominada núcleo de artefatos (*Core assets*).

O núcleo de artefatos é composto de conjunto de características comuns chamadas de similaridades, e características variáveis chamadas de variabilidades (Van der Linden et al., 2007). Tal núcleo forma a base de uma LPS que determina a Arquitetura de uma LPS, formado por, componentes reusáveis, modelos de domínios, requisitos da LPS, planos de testes e modelos de características de variabilidades (Capilla et al., 2013).

O modelo de características contém todas as características de uma LPS e os seus inter-relacionamentos. De acordo com Apel (2013), "uma característica é um comportamento característico ou visível ao usuário final de um sistema de software". Uma característica pode ser obrigatória, opcional ou alternativa. O modelo de características representa as variabilidades e as variantes de uma LPS:

- Variabilidades são descritas por: ponto de variação que permite a resolução de variabilidades em artefatos genéricos de uma LPS, e;
- Variantes são representadas pelos: possíveis elementos que podem ser escolhidos para resolver um ponto de variação.

Restrições entre variantes, estabelecem os relacionamentos entre uma ou mais variantes, com o objetivo de resolver seus respectivos pontos de variação ou variabilidade em um dado tempo de resolução (OliveiraJr et al., 2010).

A *MobileMedia* é um exemplo didático de LPS, o produto final é um software que gerencia as mídias de um aparelho celular. O núcleo de artefatos deve conter as seguintes características: opções de criar, visualizar, remover e editar a legenda da imagem. As características opcionais podem ser, a capturar uma nova imagem, ordenar e favoritar imagens. As características variantes alternativas podem ser os diversos tamanhos de tela. O produto final deve possuir ao menos uma variante alternativa.

A Figura - 2.1 apresenta um modelo de características da LPS *MobileMedia*. As arestas com círculos preenchidos representam as características obrigatórias. As arestas com círculos vazios representam características opcionais. As arestas ligadas por um triângulo, como as que saem do vértice Seleção de Mídia, representam características alternativas, por exemplo, uma instância desta LPS deve possuir ao menos um tipo de seleção de mídia, seja ele, por Foto, Música ou Vídeo.

Uma instância deste exemplo teria as seguintes características no seu produto de software final:

- Gerenciamento de Álbum:

Criar Álbum;

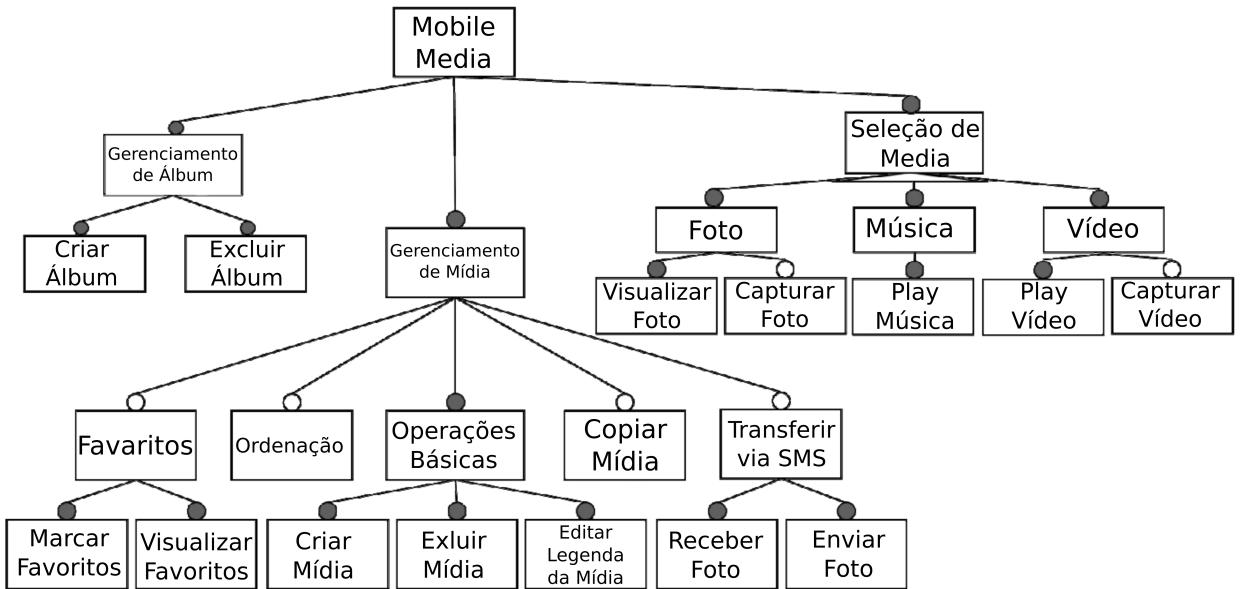
Excluir Album.

- Gerenciamento de Mídia:

– Operações Básicas:

Criar Mídia;

Figura 2.1: Um modelo de Características



Fonte: Traduzido de Sommerville (2011)

Excluir Mídia; e
Editar Legenda Mídia.

- Favoritos:
 - Marcar Favoritos;
 - Visualizar Favoritos.

- Seleção de Mídia:

Foto;
Visualizar Foto;
Capturar Foto.

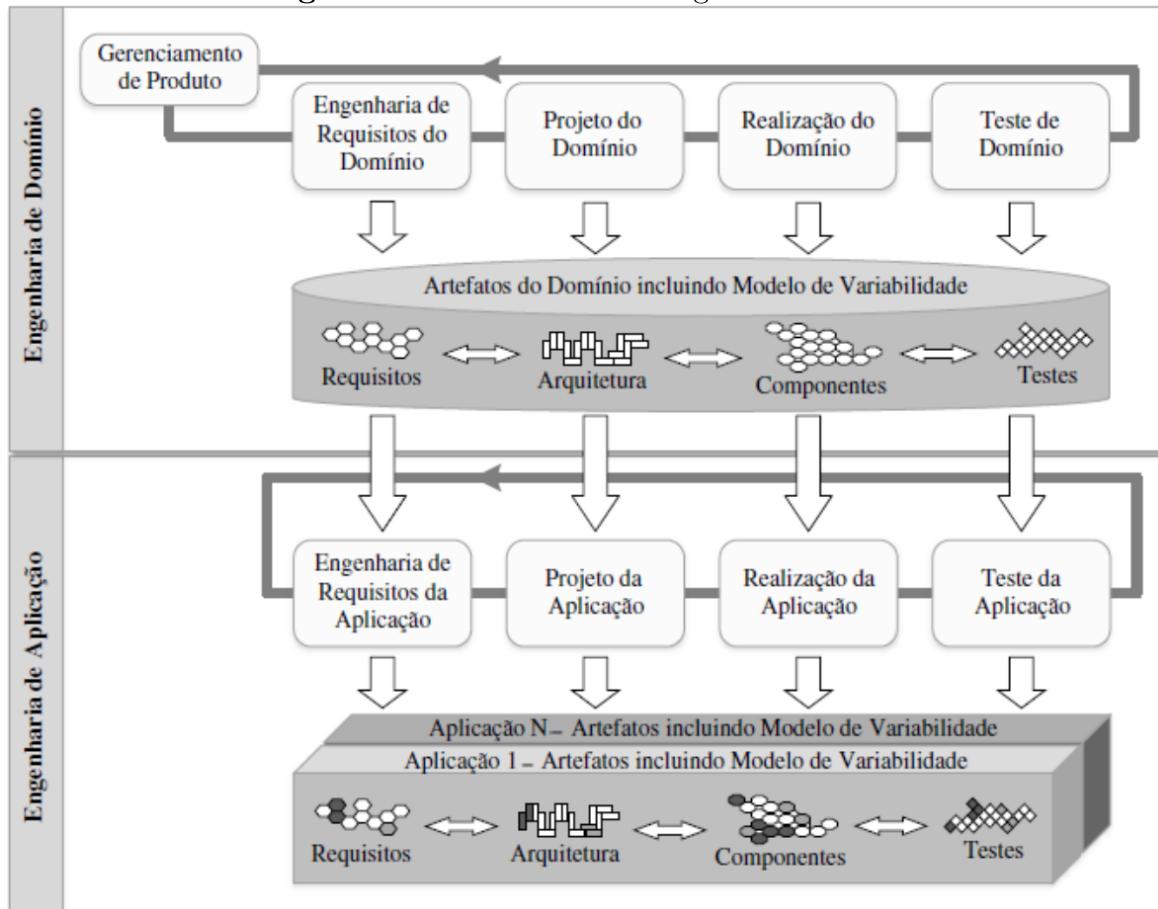
Pohl, Böckle e van Der Linden (2005) desenvolveram o *framework* para engenharia de LPS. O objetivo deste de *framework* é incorporar os conceitos centrais da engenharia de linha de produto tradicional, proporcionando a reutilização de artefatos e a customização em massa por meio de variabilidades. O *framework* está dividido em dois processos, o de Engenharia de Domínio e o de Engenharia de Aplicação, conforme apresentado na Figura - 2.2:

- **Engenharia de Domínio:** processo em que as similaridades e as variabilidades das LPSs são identificadas e realizadas, composto de cinco subprocessos principais:

Gerenciamento de Produto, Engenharia de Requisitos do Domínio, Projeto do Domínio, Realização do Domínio e Teste de Domínio;

- **Engenharia de Aplicação:** processo em que as aplicações de uma LPS são construídas por meio da reutilização de artefatos de domínio, explorando as variabilidades de uma LPS, composta dos subprocessos: Engenharia de Requisitos da Aplicação, Projeto da Aplicação, Realização da Aplicação e Teste da Aplicação.

Figura 2.2: Framework de Engenharia de LPS



Fonte: Pohl et al. (2005). Traduzido por Geraldi (2015)

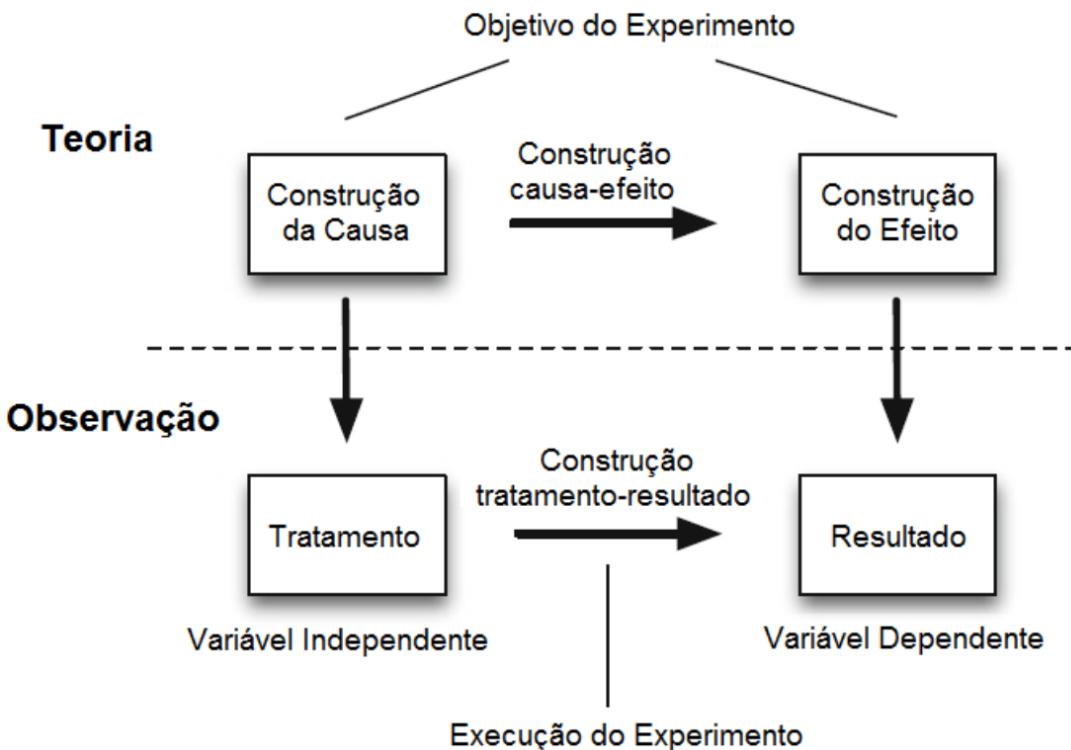
2.3 Experimentação em Engenharia de Software

Existe uma diferença relevante entre experimento e *quasi-experimento*, esta diferença está relacionada a amostra do experimento. Quando se trata de um experimento a amostra é uma representação aleatória e válida de uma determinada população, ou seja, a amostra é

uma representação da população. Quando se trata de *quasi*-experimento a amostra não é aleatória e não representa sua população. É difícil realizar experimentos em LPS, devido a dificuldade de determinar uma amostra representativa e aleatória da população, pois normalmente estas amostras são pessoas (Wohlin et al., 2012a).

Por meio de um modelo teórico entre dois ou mais fenômenos relacionados a fim de determinar se este modelo proposto pode ser considerado correto, se desenvolve o experimento onde relacionamos a causa e o efeito deste modelo. Assim, utiliza-se o modelo para criar uma hipótese em relação às mudanças particulares nos fenômenos (a causa) que levarão a mudanças no outro (o efeito). Logo, o papel do experimento é testar a hipótese para decidir se é verdadeira ou falsa (Kitchenham et al., 2015). A Figura - 2.3 apresenta à ideia de uma relação causa e efeito em teoria, na qual a parte superior à linha tracejada se encontra a teoria e, na parte inferior a observação (Wohlin et al., 2012a).

Figura 2.3: Conceitos Essenciais de um Experimento



Fonte: Traduzido de Wohlin et al. (2012a)

Um dos principais elementos de um experimento são as variáveis dependentes e independentes:

- **Variáveis independentes:** estão associadas à causa e controladas como resultado das atividades do experimentador, também são chamadas de fatores que podem assumir valores denominados tratamentos;
- **Variáveis dependentes:** estão associadas ao efeito e resultam nas mudanças que o experimentador realiza nas variáveis independentes (Kitchenham et al., 2015).

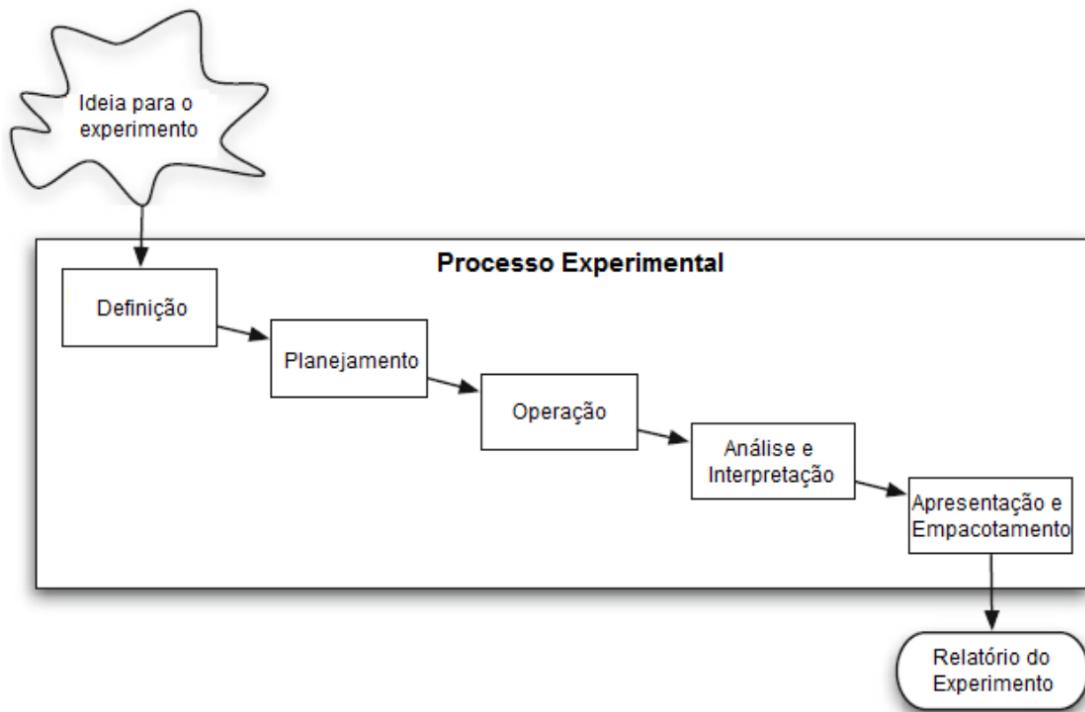
Segundo Kitchenham et al. (2015), existe uma característica dita fator de confusão em experimentos de ES que envolvem seres humanos. Esse fator pode ser representado pela presença de algum elemento indesejável no estudo que dificulta distinguir entre duas ou mais causas possíveis de um efeito que foi medido pela variável dependente como, por exemplo, os níveis de habilidade dos participantes e a extensão de suas experiências anteriores com o objeto experimental.

Em Engenharia de Software, especialmente em LPS, é difícil de se executar experimentos dado que estes devem possuir aleatoriedade completa em suas variáveis. Isto se deve à dificuldade de alocar os participantes e/ou objetos a diferentes tratamentos de maneira aleatória, bem como, à falta de representatividade do número de participantes em uma amostra da população. Portanto, os experimentos realizados nesta área são, frequentemente, *quasi-experimentos*, nos quais não há aleatoriedade dos participantes e/ou dos objetos experimentais, em ES chamados de artefatos de software, podendo ser processos ou ferramentas (Wohlin et al., 2012a).

Segundo Wohlin et al. (2012a) a realização de um experimento pode ser dividido em um processo contendo cinco atividades, conforme apresentadas na Figura - 2.4 e descritas a seguir:

- **Definição:** é a primeira atividade, onde define-se o problema, objetivo e metas do experimento. Caso não seja devidamente estabelecida, pode ocorrer retrabalho ou o experimento não pode ser utilizado para se estudar o que era almejado;
- **Planejamento:** é uma preparação de como o experimento será conduzido, em que ocorre a determinação do contexto do experimento, a formulação das hipóteses, sendo **hipótese nula** que o experimentador espera rejeitar com a maior confiança possível e a **hipótese alternativa** que se espera aceitar, a seleção de variáveis (dependentes e independentes), a seleção dos participantes, o projeto do experimento, a instrumentação e a avaliação da validade, dividida em quatro tipos, sendo **validade interna** refere-se ao relacionamento tratamento-resultado; **validade externa** apresenta a generalização dos resultados a uma população maior; **validade de constructo** demonstra a relação entre a teoria e observação;

Figura 2.4: Visão Geral do Processo Experimental



Fonte: Traduzido de Wohlin et al. (2012a)

e **validade de conclusão** refere-se a como os experimentadores foram aptos de analisar os resultados de um estudo e se a forma como foi feita é apropriada (Kitchenham et al., 2015).

- **Operação:** essa atividade é composta da **preparação** dos participantes e dos materiais necessários (instrumentação); **execução** das tarefas pelos participantes de acordo com diferentes tratamentos e coleta dos dados; e **validação dos dados** pelo experimentador, verificando os dados informados pelos participantes, de forma que os resultados do experimento sejam válidos;
- **Análise e Interpretação:** os dados coletados na atividade anterior são analisados utilizando a estatística descritiva. Após isso, é verificada a necessidade de redução do conjunto de dados, de forma a garantir que os dados representam uma informação correta e/ou esperada. Por fim, realiza-se o teste de hipóteses para avaliar estatisticamente se hipótese nula pode ser rejeitada.
- **Apresentação e Empacotamento:** nessa atividade, os resultados são reportados, por exemplo, como artigos em conferência e/ou periódico, relatórios de tomada de

decisão e empacotados para permitir a replicação do experimento, como material educativo, entre outros.

2.4 Experimentos de LPS

....apresentar um resumo do MS da Viviane, os trabalhos que ela leu, como foi construído o modelo conceitual para o leitor poderá entender como surgiu a sua ontologia...

2.5 Ontologias

A palavra ontologia é formada por meio dos termos gregos ontos (ser) e logos (estudo, discurso), que engloba algumas questões abstratas como a existência de determinadas entidades, o que se pode dizer que existe, qual o significado do ser, etc. Segundo Wolff e École (1962), ontologia é um ramo da filosofia que estuda a realidade e existência, ou o ser enquanto ser. Em outras palavras, é o estudo da descrição de coisas do mundo real. Outro ponto de vista proposto por Gruber (1993), diz que ontologias são uma especificação formal de uma contextualização e uma contextualização é uma visão abstrata e simplificada do mundo.

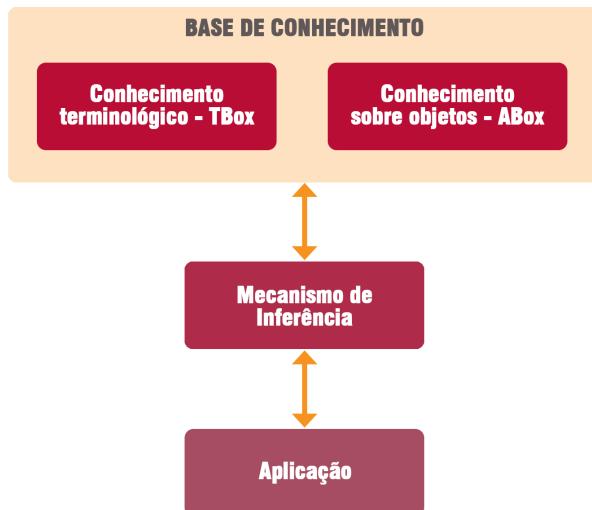
Ontologia em Computação, Sistemas de Informação e Ciência da Informação, é definida como um modelo de dados que representa um conjunto de conceitos dentro de um domínio e os relacionamentos entre estes. Uma ontologia é utilizada para realizar inferência sobre os objetos do domínio. No cenário atual, as ontologia em ciências da informação são utilizadas como uma forma de representação de conhecimento lógico, possibilitando a inferência de novos fatos com base nos dados armazenados na ontologia.

Uma ontologia define primitivas/diretrizes de um domínio de conhecimento, estas primitivas/diretrizes podem ser definidas como classes, atributos, propriedades e restrições. Essas definições seguem o padrão de representação conhecido como lógica descritiva. A lógica descritiva representa os conceitos de um domínio (chamado de **TBox - Terminological Box**) separadamente dos indivíduos (chamado de **ABox - Assertion Box**) (Calvanese et al., 2005). A lógica descritiva é mais representativa e eficiente que a lógica proposicional e a lógica de predicados (usados em linguagens de programação lógica, como em Prolog).

Portanto uma ontologia para a representação de um conhecimento possui a seguinte estrutura: Uma base de conhecimento onde estão os dois conjuntos de conhecimento terminológico (**TBox**) e o conjunto de conhecimento sobre objetos (**ABox**), seguido de

um mecanismo de inferência e uma aplicação para atuar na manipulação de informações extraídas do mecanismo de inferência, A Figura - 2.5 apresenta essa estrutura.

Figura 2.5: Estrutura de uma Ontologia



Fonte: o Autor

Na prática uma ontologia têm o propósito permitir o compartilhamento de conteúdos conceitualizados formalizando o conhecimento de um determinado domínio. Abordando com um conjunto de termos que serão usados para a elaboração de consultas de conteúdos. Estas consultas são chamadas de inferência.

Segundo Morais e Ambrósio (2007), A ontologia define as regras de combinação entre os termos e seus relacionamentos, estes relacionamentos são criados por especialistas, e os usuários formulam consultas usando os conceitos especificados. Dessa forma o usuário que está interagindo com uma aplicação que realiza inferência na ontologia, pode navegar pela ontologia em busca de conceitos adequados para a representação do objeto em estudo.

O benefícios de se usar ontologias pode ser descritos em 3 áreas (GUIZZARDI, 2000):

- **Comunicação:** As ontologias possibilitam a comunicação entre pessoas acerca de determinado conhecimento, pois permitem raciocínio e entendimento sobre um domínio;
- **Formalização:** A formalização está relacionada à especificação da ontologia, que permite eliminar contradições e inconsistências na representação de conhecimento, além de não ser ambígua. Além disso, essa especificação pode ser testada, validada e verificada;

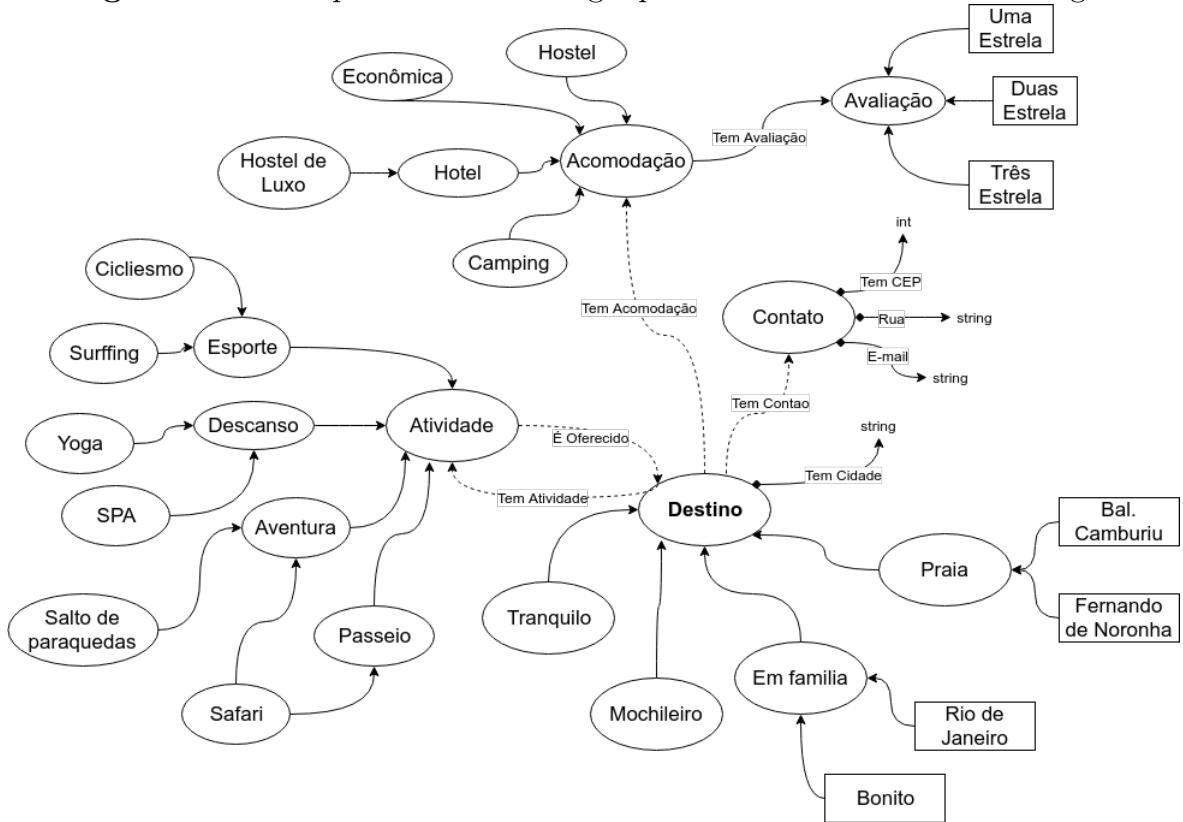
- **Representação de Conhecimento:** As ontologias formam um vocabulário de consenso que permite representar conhecimento de um domínio em seu nível mais alto de abstração, desta forma, aumentando o potencial de reutilização.

Assim GUIZZARDI (2000) classificou as ontologias em cinco categorias, uma para cada foco em específico:

- **Ontologias genéricas:** descrevem conceitos bastante gerais, tais como, espaço, tempo, matéria, objeto, evento, ação, etc., que são independentes de um problema ou domínio particular. Geralmente buscam por meio deste tipo de ontologia relacionar ao sentido filosófico de categorização e lingüística, tal como sobre as teorias básicas do mundo, de modo bem abrangente aplicando ao conhecimento de senso comum;
- **Ontologias de domínio:** expressam conceituações de domínios particulares, descrevendo o vocabulário relacionado a um domínio genérico, tal como Medicina, Direito ou Computação. Este tipo de ontologia é o mais comum, sendo aplicado para representar um micro-mundo ao qual o domínio está vinculado;
- **Ontologias de tarefas:** expressam conceituações sobre a resolução de problemas, independentemente do domínio em que ocorram, isto é, descrevem o vocabulário relacionado a uma atividade ou tarefa genérica, tal como, diagnose ou vendas. Tendo como destaque a facilidade em interligar conhecimentos de tarefas e domínio, buscando esta interligação de modo uniforme e consistente;
- **Ontologias de aplicação:** descrevem conceitos dependentes do domínio e da tarefa particulares. Estes conceitos frequentemente correspondem a papéis desempenhados por entidades do domínio quando da realização de uma certa atividade;
- **Ontologias de representação:** explicam as conceituações que fundamentam os formalismos de representação de conhecimento, tendo como foco deixar claras as informações contidas nesse formalismo.

A Figura - 2.6 apresenta um exemplo de ontologia, por meio de um grafo, para o domínio: “destino de viagem”. Os vértices ovais representam as classes, e os vértices retangulares representam os indivíduos (instâncias da classe). As arestas comuns representa um relacionamento de classe e subclasse, as arestas tracejadas representam um relacionamento de propriedade, já as arestas que começam com um losango indica a definição de uma propriedade, especificando sua tipagem.

Figura 2.6: Exemplo de uma Ontologia para o domínio: Destino de Viagem



Fonte: o Autor

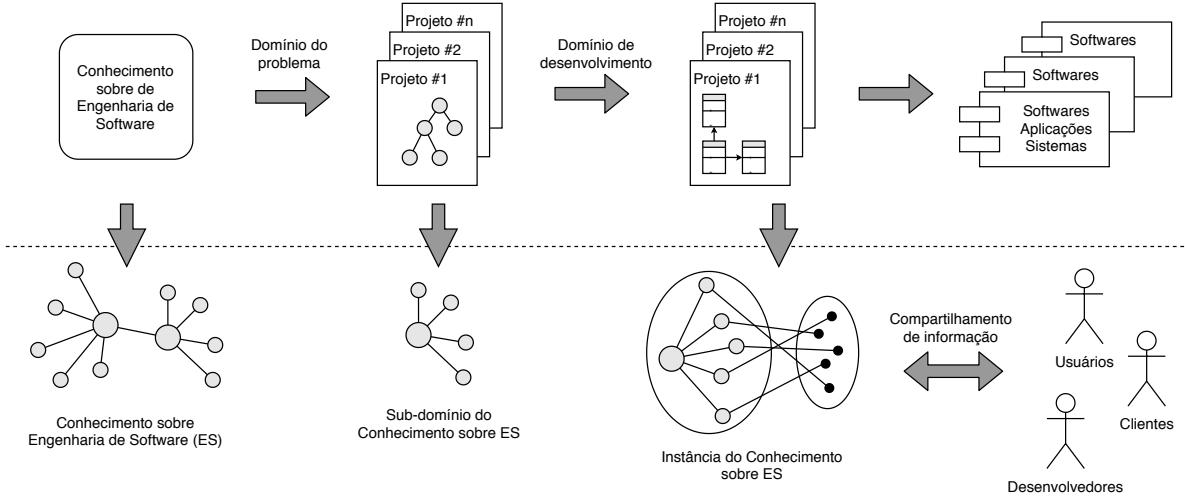
2.5.1 Ontologias para ES

As ontologia pode estar associada à Ciência da Computação em diferentes visões, sendo elas: (i) Recuperação de informações na Internet, onde se utiliza de sistemas on-line que possibilitam a pesquisa sobre a contextualização de fatos, como também permite o desenvolvimento de sites baseados em ontologia na web-semântica, (ii) Gestão do Conhecimento, visto que pode-se criar um sistema de armazenamento e acesso a informações, educacionais ou empresariais, através do uso de ontologias, (iii) Web-Semântica, esta área da Ciência da Computação dita como a visão para o futuro da internet, onde temos informações associadas a um significado explícito, deixando mais fácil o processamento e interpretação de informações para máquinas, (iv) Educação, onde se permite utilizar uma ontologia em um ambiente de aprendizagem sobre diversos conceitos associados à Ciência da Computação, possibilitando a interpretação de domínios, e no auxílio de pesquisas dinâmicas (Morais e Ambrósio, 2007).

O trabalho de Isotani et al. (2015), apresenta a *Ontology-Driven Software Engineering* a ES baseada em Ontologias. Esse tema é tratado como uma subárea da ES e tem como

objetivo estudar as diversas formas de como as ontologias, engenharia de ontologias e outras tecnologias procedentes podem contribuir para o processo de desenvolvimento de software.

Figura 2.7: Ontologias para formalizar o conhecimento em ES



Fonte: Adaptado de Isotani et al. (2015)

A Figura - 2.7 apresenta a formalização do conhecimento de ES via ontologias de domínio, dessa forma podem ser explorados os conhecimentos dos diversos processos de análise, desenvolvimento, artefatos, entre outros conhecimentos que podem ser representados por uma ontologia. A formalização do sub-domínio do conhecimento de ES contendo informações específicas sobre conceitos relacionados ao domínio, por exemplo o sub-domínio de LPS. Por fim a instância de conhecimento sobre ES apresentando um espaço que consta a consolidação e relações de conhecimento contido na ontologia para explorar os conceitos associados ao domínio de ES.

As ontologias garantem que a mesma linguagem, métodos, processos e entendimento sobre a área de ES sejam utilizados e compartilhados formalmente, mesmo que os envolvidos estejam espalhados em diferentes localidades Isotani et al. (2015).

2.5.2 Tipos de Ontologias

Na literatura existem diferentes aplicações, conteúdos e funções para as ontologias. Tal diversidade dificulta sua tipificação ao mesmo tempo em que aumenta a quantidade de abordagens para elaboração de ontologias. Almeida e Bax (2003) diferenciar alguns tipos de ontologias classificando-as em abordagens: (i) quanto à função, (ii) ao grau de formalismo, (iii) à aplicação, (iv) à estrutura e (v) ao conteúdo. a Tabela - 2.1 apresenta um breve resumo dessa tipologia.

Tabela 2.1: Tipos de Ontologia

Abordagem	Classificação	Descrição
Quanto à função Mizoguchi; Vanwelkenbuyse n; Ikeda (1995)	Ontologias de domínio	Reutilizáveis no domínio, fornecem vocabulário sobre conceitos e seus relacionamentos, sobre as atividades e regras que os governam.
	Ontologias de tarefa	Fornecem um vocabulário sistematizado de termos, por meio da especificação de tarefas que podem ou não estar no mesmo domínio.
	Ontologias gerais	Incluem um vocabulário relacionado a coisas, eventos, tempo, espaço, causalidade, comportamento, funções etc.
Quanto ao grau de formalismo Uschold; Gruninger (1996)	Ontologias altamente informais	Expressa livremente em linguagem natural.
Quanto à aplicação Jasper; Uschold (1999)	Ontologias semi informais	Expressa em linguagem natural de forma restrita e estruturada.
	Ontologias semi formais	Expressa em uma linguagem artificial definida formalmente
	Ontologias rigorosamente formais	Os termos são definidos com semântica formal, teoremas e provas.
Quanto à estrutura Haav; Lubi (2001)	Ontologias de autoria neutra	Uma ontologia escrita em uma única língua e depois convertida para uso em diversos sistemas, utilizando-se as informações.
	Ontologias como especificação	Cria-se uma ontologia para um domínio, a qual é usada para documentação e manutenção no desenvolvimento de softwares.
	Ontologias de acesso comum à informação	Quando o vocabulário é inacessível, a ontologia torna inteligível a informação, proporcionando conhecimento compartilhado dos termos
Quanto ao conteúdo Van-Heijst; Schreiber; Wielinga (2002)	Ontologias de alto nível	Descrevem conceitos gerais relacionados a todos os elementos da ontologia (espaço, tempo, matéria, objeto, evento etc.), os quais são independentes do problema ou domínio.
	Ontologias de domínio	Descrevem o vocabulário relacionado a um domínio, como, por exemplo, medicina ou automóveis
	Ontologias de tarefa	Descrevem uma tarefa ou atividade, como por exemplo, diagnósticos ou compras, mediante inserção de termos especializados na ontologia.
	Ontologias de informação	Especificam a estrutura de registros de bancos de dados (por exemplo, os esquemas de bancos de dados).
	Ontologias terminológicas	Especificam termos que serão usados para representar o conhecimento em um domínio (por exemplo, os léxicos).
	Ontologias de modelagem do conhecimento	Especificam conceituações do conhecimento, têm uma estrutura interna semanticamente rica e são refinadas para uso no domínio do conhecimento que descrevem
	Ontologias de aplicação	Contêm as definições necessárias para modelar o conhecimento em uma aplicação.
	Ontologias de domínio	Expressam conceituações que são específicas para um determinado domínio do conhecimento.
	Ontologias genéricas	Similares às ontologias de domínio, mas os conceitos que as definem são considerados genéricos e comuns a vários campos
	Ontologias de representação	Explicam as conceituações que estão por trás dos formalismos de representação do conhecimento.

Fonte: Almeida e Bax (2003)

2.5.3 Metodologias para Construção de Ontologias

As metodologias para construção de ontologias têm como objetivo sistematizar o projeto de desenvolvimento das mesmas.

A tese de Mendonca (2015) relaciona as principais metodologias / métodos e etapas de cada metodologia/método para construção de ontologias, esses métodos estão descritos no Apêndice A.1. Segundo ele o processo de construção de uma ontologia é complexo, pois envolve a criação de modelos semânticos ou descrições simplificadas da realidade de um dado domínio e também exige dos desenvolvedores conhecimentos técnicos em modelagem conceitual, em lógica formal e em alguns aspectos filosóficos.

A escolha da metodologia para a construção da ontologia impacta diretamente o modelo final, sendo necessário um estudo sobre suas etapas e processos a serem executados.

2.6 Trabalhos Relacionados

Durante uma revisão não sistemática (*ad hoc*) da literatura foram encontrados alguns estudos que propõem abordagens para representar formalmente dados sobre experimentos em ES. A revisão da literatura mostrou que a maioria dos estudos focou em representar todo o domínio do ES, ou experimentos em geral, o que é um fator agravante para elaboração dos modelos, por causa da quantidade de detalhes que pode variar significativamente. Estes trabalhos estão descritos a seguir. A contribuição deles é discutida ao final desta seção.

Os trabalhos de Garcia et al. (2008), Scatalon et al. (2011) e da Cruz et al. (2012) se destacam em nosso contexto por propor e modelar ontologias específicas para experimentos em engenharia de software. O trabalho de Garcia et al. (2008) propõe, por meio de diagramas de classes UML, uma ontologia para experimentos controlados em engenharia de software denominada EXPEROntology. Com o objetivo de ser uma ferramenta de transferência de conhecimento para pesquisadores e revisores, além de propor meta-análises, conduzir e avaliar experimentos controlados. O trabalho de Scatalon et al. (2011) é uma evolução do trabalho de Garcia et al. (2008), mas focado na evolução da ontologia proposta. O trabalho de da Cruz et al. (2012) apresenta uma ontologia chamada OVO (*Open on Vence Ontology*) na qual é inspirada por três teorias: (i) O ciclo de vida de experimentos científicos, (ii) *Open Provent (OPM)* e (iii) *Unified Foundational Ontology (OVNI)*. Este modelo OVO pretende ser uma referência para modelos conceituais que podem ser usados por pesquisadores para explorar a semântica de metadados.

Por outro lado, os trabalhos de Blondet et al. (2016) e Soldatova e King (2006) tratam ontologias no contexto geral de experimentos. O trabalho de Blondet et al. (2016) traz uma proposta de ontologia para DoE (*Designs of Experiments*) para apoiar as decisões de processo de experimentação. O trabalho de Soldatova e King (2006) propõe a ontologia EXPO que é uma cerne da ontologia SUMO (*Suggested Upper Merged Ontology*). Esta ontologia visa especificar os experimentos formalizando e generalizando os conceitos de *design*, metodologias e representação de resultados. Este trabalho é o único que usa o modelo OWL-DL para representar a ontologia.

O trabalho de Gelernter e Jha (2016) fornece uma visão geral sobre os desafios de avaliar uma ontologia, mas não trata da ontologia para experimentos.

Finalmente, o trabalho de Cruzes et al. (2007) trata de uma técnica para extrair meta-information de experimentos em ES, o que está assunto está relacionado a este trabalho, pois a OntoExper-SPL toma como base metadados sobre experimentos em SPL.

Apesar de não haver um padrão para modelar o conhecimento de experimentação em ES a literatura apresenta estratégias diferentes neste sentido, como pode ser notado nos trabalhos de Garcia et al. (2008) e Scatalon et al. (2011). Entretanto as modelagens propostas por estes trabalho serviram como *insights* para fases de criação da OntoExper-SPL. Desse modo, a modelagem de ontologia para este trabalho será aplicada para experimento em LPS, de tal modo que, será possível fazer inferência de LPS na ontologia e extrair informações. Assim, a OntoExper-SPL se diferencia das outras propostas por ser puramente construída usando o padrão OWL.

2.7 Considerações Finais

Este capítulo apresentou os conceitos essenciais sobre a abordagem de LPS como variabilidades e variantes, apresentando o *MobileMedia* como exemplo para melhor entendimento dos conceitos, bem como o framework de engenharia de LPS proposto por Pohl et al. (2005).

Em relação à experimentação em ES, foi apresentado os principais elementos como as variáveis dependentes e independentes, as atividades de um processo experimental como definição, planejamento, operação, análise e interpretação, apresentação e empacotamento. Sobre qualidade de experimentos em ES foi introduzido conceitos preliminares sobre a qualidade de experimentos como viés, validade interna e validade externa e também algumas abordagens para aplicar uma avaliação de qualidade sobre os experimentos em ES.

Quando foi tratado de ontologia, foi descrito a definição universal de ontologia e a definição voltada para computação, mais especificamente para ES. Nessa temos que a ontologia é uma modelagem formal de conhecimento para representação de um determinado domínio, para isso, existe a *TBox* para representar a lógica descritiva e a *ABox* para representar os indivíduos pertencentes a ontologia. Assim como em LPS foi apresentado um exemplo (Destino de Viagem) para facilitar o entendimento dos conceitos abordados.

Por fim, temos os trabalhos relacionados traçando um paralelo de modelagem para representação do domínio de LPS eom este trabalho.

No próximo capítulo, será apresentado uma ontologia para experimento em LPS, a OntoExper-SPL.

OntoExper-SPL: uma Ontologia para Experimentos de LPS

3.1 Considerações Iniciais

Este capítulo apresenta a elaboração da proposta de ontologia OntoExper-SPL, sua concepção, a construção do projeto, exemplos de aplicação e uma avaliação preliminar. Dessa forma, a Seção 3.2 apresenta o processo de construção da ontologia seguindo as etapas de Mendonca (2015) e a tipologia de Almeida e Bax (2003) bem como a elaboração de um grafo como modelagem inicial da ontologia, seguindo como base um modelo conceitual clusterizado sobre elementos experimentais de ES em LPS. A Seção 3.3 apresenta o projeto da OntoExper-SPL para a qual, foi utilizado como tecnologia base *Ontology Web Language* (OWL) juntamente com a Ferramenta Protégé. O resultado final foi um artefato do tipo OWL contendo a modelagem de classes, subclasses, propriedades de objeto e propriedades dos dados. Além da modelagem, foi criado um programa automatizado para povoar a ontologia utilizando os metadados de trabalhos anteriores. Este programa conta com a utilização da linguagem de programação Python e a biblioteca OwlReady2. Na Seção 3.4 foi elaborado um breve exemplo de aplicação da ontologia e uma breve explicação deste procedimento. Na Seção 3.5 foi realizado uma avaliação preliminar da ontologia, por meio da ferramenta OOPS!¹ que avalia pontos de falha da ontologia. Foram discutidos como esses pontos de falha impactam na ontologia proposta por este trabalho.

¹<http://oops.linkeddata.es>

3.2 Concepção

A metodologia aplicada na construção desta proposta, foi a **MFPFO**, descrita por Mendonca (2015). O propósito dessa metodologia é ser multifacetada, anotada semanticamente, para a modelagem de uma família de produtos. Tal metodologia é capaz de sugerir anotações semanticamente relacionadas, baseadas no design e no repositório de construção. O domínio de aplicação dessa metodologia é voltado para família de produtos. Todas as metodologias e métodos no podem ser encontradas no Apêndice A.1.

As etapas de construção dessa metodologia são:

1. Construção de uma taxonomia da família de produtos;
2. Extração de entidades;
3. Identificação do conceito e geração da unidade facetada;
4. Anotação semântica e modelagem faceta;
5. Construção de uma ontologia de família de produtos multifacetada e anotada semanticamente;
6. Avaliação e validação da ontologia.

Com base nas tipologias proposta por Almeida e Bax (2003) e descritos na Tabela - 2.1, definimos o tipo da ontologia proposta neste trabalho.

O tipo é caracterizado como; **quanto à função**: é uma ontologia de domínio, **quanto ao grau de formalismo**: é uma ontologia semi formal, **quanto à aplicação**: é uma ontologia de especificação, **quanto à estrutura**: é uma ontologia de domínio e **quanto ao conteúdo**: é uma ontologia para modelagem de conhecimento e para aplicação de domínio.

Também foi aplicado o seguinte processo de elaboração da ontologia, seguindo os passos proposto por Fernanda (2007):

- definição e estruturação dos termos por meio de classes;
- estabelecimento de propriedades (atributos) inerentes ao conceito representado por um termo;
- povoamento da estrutura que satisfaçam um conceito e as suas propriedades;
- estabelecimento de relações entre os conceitos; e

- elaboração de sentenças para restringir inferências de conhecimento baseadas na estrutura.

Esta modelagem inicial da ontologia foi baseada no trabalho de mapeamento sistemático de experimentos em LPS de Furtado (2018), por meio de uma análise exploratória dos dados levantados nesse mapeamento. Foi possível usá-lo como guia de informações, metadados de ESE em LPS e o modelo conceitual.

O mapeamento sistemático levou em consideração o temple experimental de Wohlin, que traz cinco fases para elaboração de ESE, são eles, Definição, Planejamento, Operação, Análise e Interpretação (Wohlin et al., 2012b).

O primeiro passo foi desenvolver um grafo como modelo inicial da ontologia OntoExper-SPL, objetivo principal foi externalizar idéias iniciais do modelo e visualizar hierarquias e relacionamentos entre os termos inicialmente propostos.

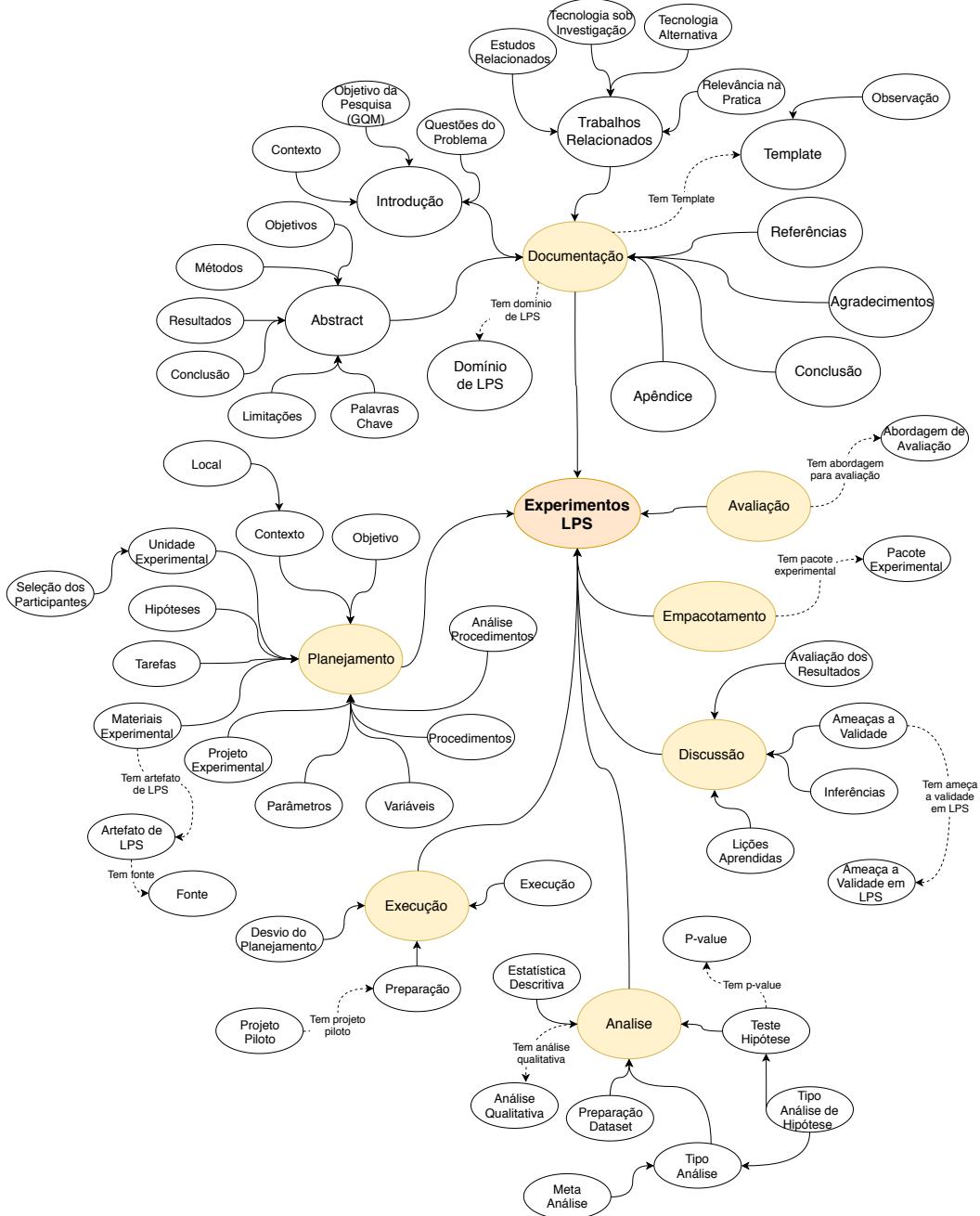
A Figura - 3.1 representa este modelo inicial contendo a definição principal dos termos para um experimento: Experimento LPS, Documentação, Template, Avaliação, Discussão, Análise, Execução e Planejamento. Em seguida o grafo foi refinado para termos, de menor granularidade, definidas a seguir:

- Documentação:
 - Domínio em LPS;
 - *Abstract*:
 - * Palavra Chave;
 - * Limitações;
 - * Conclusões;
 - * Resultados;
 - * Métodos; e
 - * Objetivos.
 - Introdução:
 - * Contexto;
 - * Objetivo da Pesquisa (GQM); e
 - * Questões do Problema.
 - Trabalhos Relacionados:
 - * Estudos Relacionados;
 - * Tecnologia sob Investigação;

- * Tecnologia Alternativa; e
- * Relevância na Prática.
- Template:
 - * Observação.
- Referências;
- Agradecimentos;
- Conclusão; e
- Apêndice.
- Planejamento:
 - Contexto:
 - * Local.
 - Unidade Experimental:
 - * Seleção dos Participantes.
 - Material Experimental:
 - * Artefatos de LPS:
 - Fonte.
 - Projeto Experimental;
 - Parâmetros;
 - Variáveis;
 - Procedimentos;
 - Objetivo;
 - Hipóteses;
 - Tarefas; e
 - Análises de procedimentos.
- Execução:
 - Preparação:
 - * Projeto Piloto.
 - Execução; e

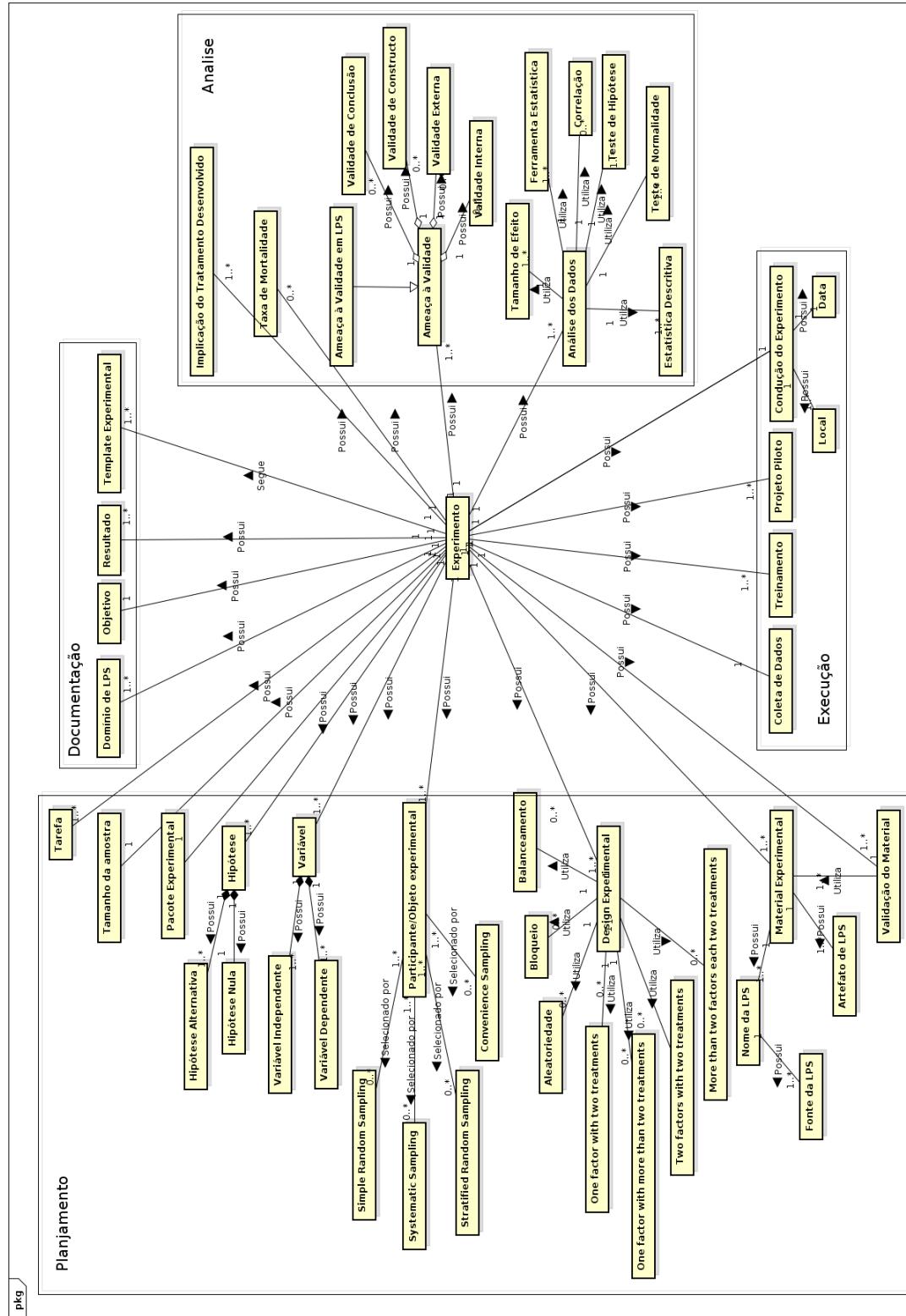
- Desvio do Planejamento;
- Análise:
 - Tipo de Análise:
 - * Meta Análise; e
 - * Tipo de Análise de Hipótese.
 - Teste de Hipótese:
 - * P-value.
 - Estatística Descritiva;
 - Análise qualitativa; e
 - Preparação do *Dataset*;
- Discussão:
 - Ameaças à Validade:
 - * Ameaça à Validade em LPS.
 - Avaliação dos Resultados;
 - Inferências; e
 - Lições Aprendidas;
- Empacotamento:
 - Pacote Experimental.
- Avaliação:
 - Abordagem de Avaliação.

Figura 3.1: Grafo inicial da proposta de ontologia



Fonte: o Autor

Figura 3.2: Modelo Conceitual Clusterizado



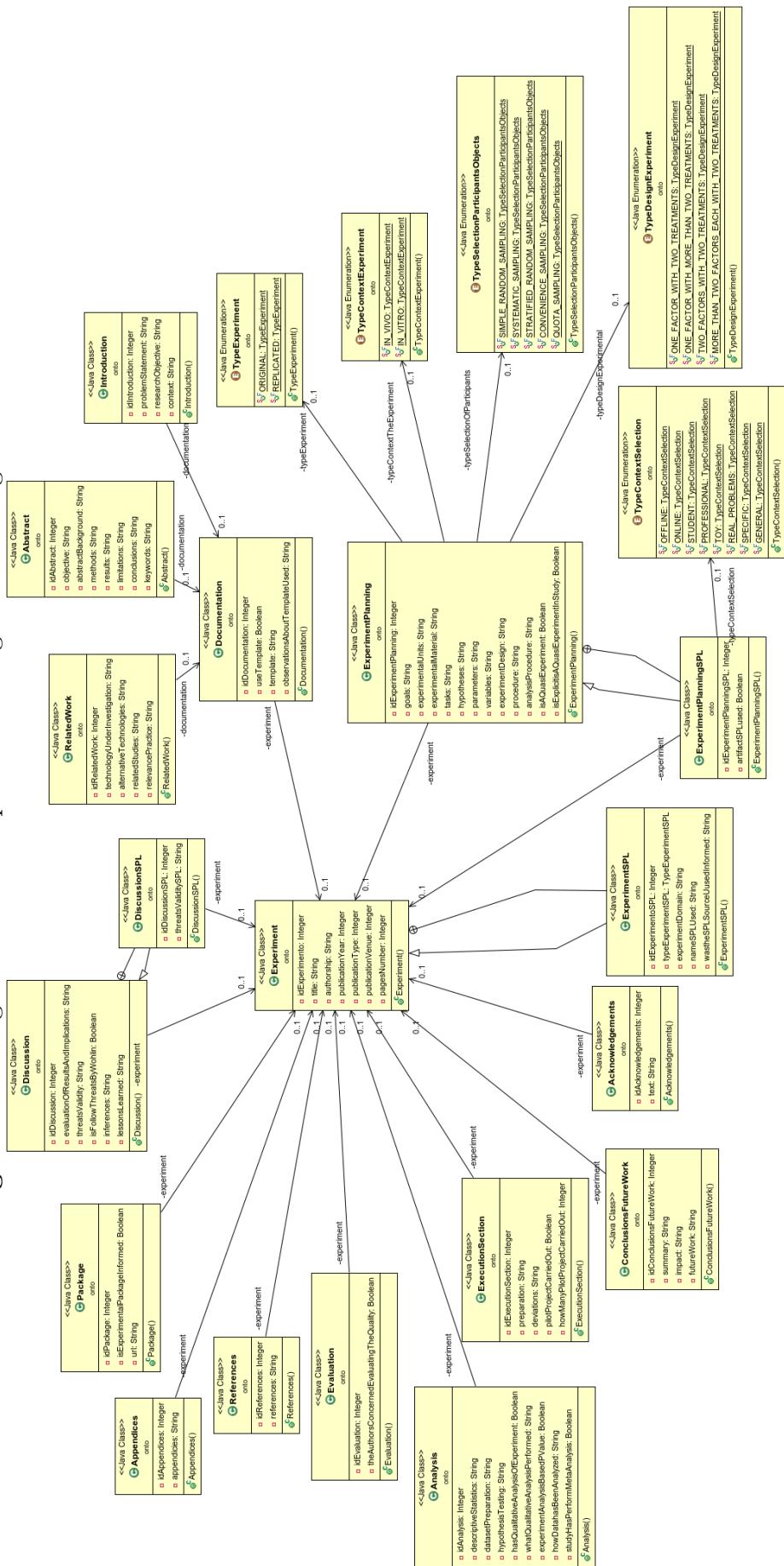
Fonte: o Autor

A Figura - 3.2, apresenta uma modificação do modelo conceitual original de Furtado (2018), em que foi aplicada a clusterização separando cada cluster para cada fase do template experimental de Wohlin. Essa clusterização foi necessária para a compreensão mais abstrata das relações entre os termos do domínio identificados no modelo conceitual original e as fases do template do Wholin. Dessa forma foi possível validar cada termo do grafo inicialmente proposto.

Em seguida, um diagrama de classes foi elaborado para representar de uma maneira mais intuitiva a modelagem, realizada com grafos. O objetivo é transformar cada termo do grafo em uma classe do conceito de Orientação a Objetos. Nessa representação a relação entre os termos (classes) e suas propriedades (atributos) é perceptível para o usuário. Essa forma de representação destacou a relação principal quando é definido a composição da classe **Experiment** e **ExperimentSPL** em quase todas as outras subclasses. Nessa representação também ficou explícita o tipo das propriedades, para posteriormente executar a inserção da *ABox* na modelagem.

A Figura - 3.3 apresenta o diagrama de classes gerado para a modelagem da ontologia. Esse diagrama representa todas as composições de classes que existe com **Experiment** e **ExperimentSPL**, bem como as relações hierárquicas como por exemplo, **ExperimentSPL** é uma subclasse de **Experiment**.

Figura 3.3: Diagrama de classes para a modelagem da ontologia



Fonte: O Autor

3.3 Projeto

A ontologia foi criada usando a tecnologia OWL. Para isso foram construídas as classes e subclasses para representar os elementos da ontologia, por meio da ferramenta Protégé.

3.3.1 Modelagem com Protégé

A ferramenta Protégé¹ é um ambiente de desenvolvimento voltado para criação de ontologias. Ela dispõe de uma interface gráfica para edição de ontologias e uma arquitetura para a criação de ferramentas baseadas em conhecimento. Pode ser usada por desenvolvedores de sistema e por especialistas em domínio para criar bases de conhecimento, permitindo representar o conhecimento de uma área. Esse ambiente é capaz de tratar classes, com suas definições e exemplos e simultaneamente propriedades de objetos e de dados (Musen, 2015).

Fazendo uma analogia ao diagrama de classes, no Protégé, as classes, os atributos e seus relacionamentos estão em um contexto de entidades. As classes e hierarquias são definidos na aba **Class**, os relacionamentos são definidos na aba **Object properties** e os atributos são definidos na aba **Data properties**.

As entidades foram definidas com a seguinte estrutura: (i) definição de classes, (ii) definição das propriedades de objetos e (iii) definição das propriedades dos dados.

Definição de classes

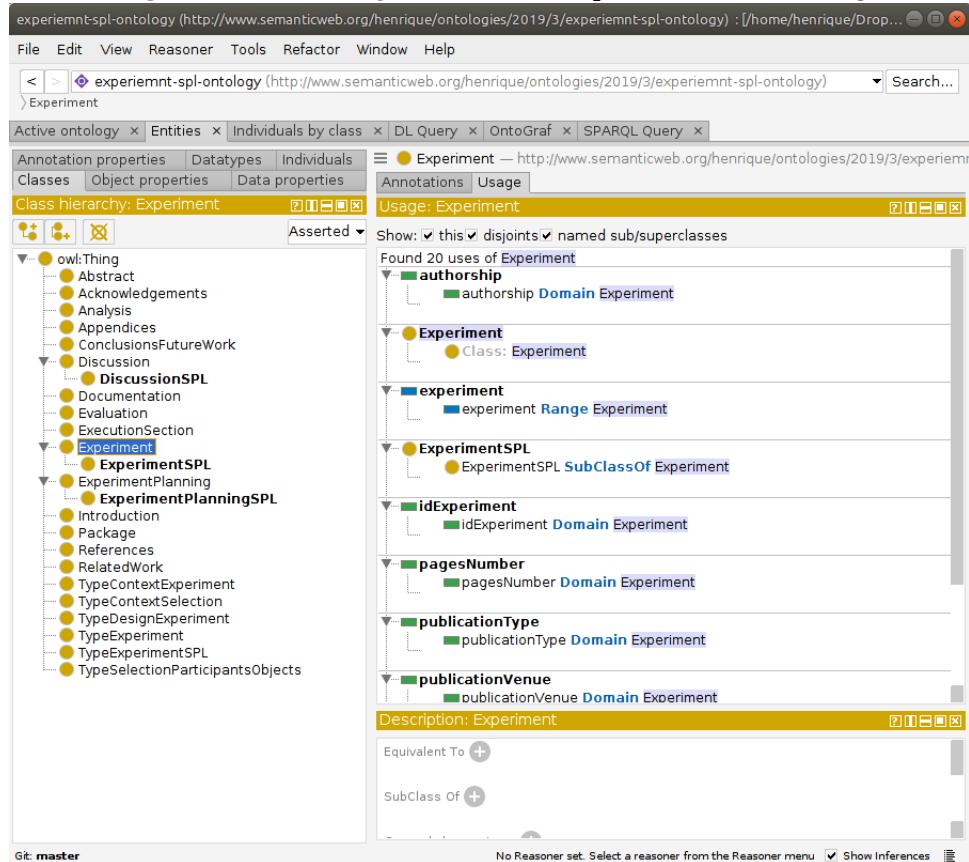
No Protégé, é definido uma classe raiz chamada **Thing** na qual todas as outras classes são subclasses. A Figura - 3.4 apresenta a definição da classe **Experiment** dentro do modelo, neste momento a definição é composta do nome da classe e seus principais relacionamentos (**Equivalent To** e **SubClass Of**, no painel direito inferior).

Definição das propriedades de objetos

Uma propriedade de objeto raiz chamada **topObjectProperty** é definido na qual todas as outras propriedades são sub-propriedades dela. A Figura - 3.5 apresenta a definição de propriedade de objeto para propriedade **documentation**, neste momento a definição é composta do nome da propriedade e seus relacionamentos com classes ((**Domains**) e (**Ranges**) no painel **Description**: **documentation** do lado direito inferior).

¹<https://protege.stanford.edu>

Figura 3.4: Definição da classe Experiment no Protégé

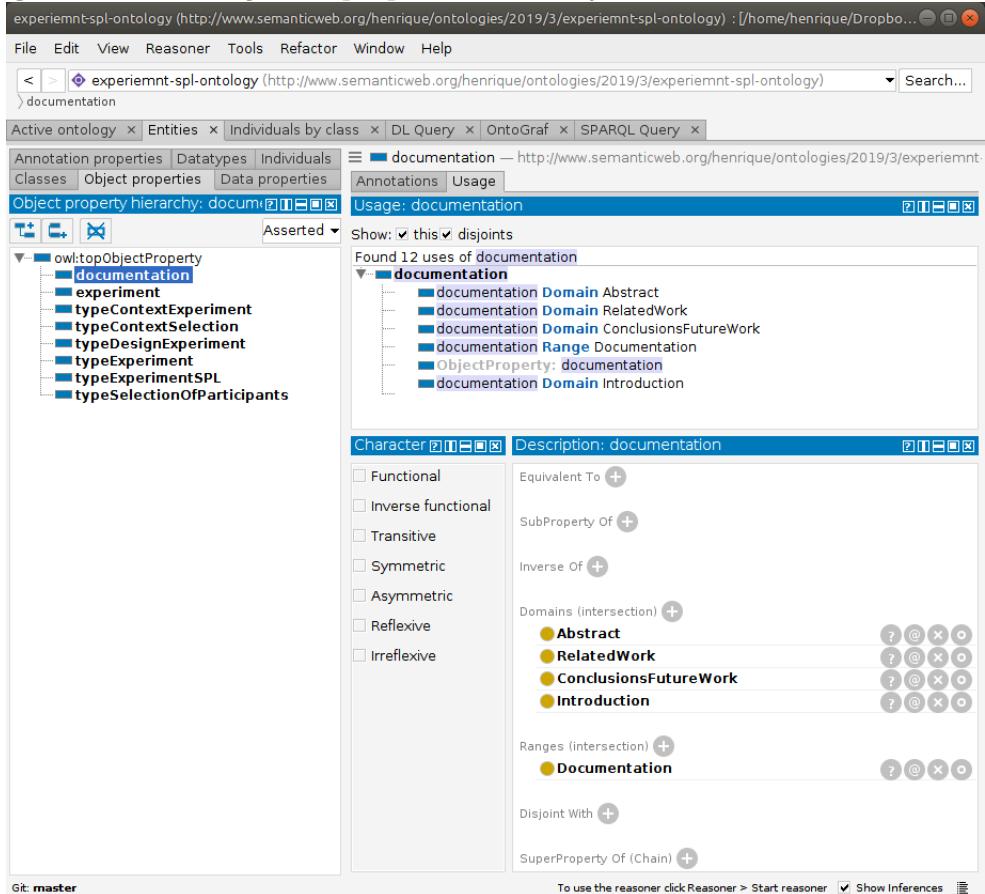


Fonte: o Autor

Definição das propriedades dos dados

Uma propriedade de dados raiz chamada `topDataProperty` é definida na qual todas as outras propriedades são sub-propriedades dela. Para cada Propriedade de dado é definido um conjunto de classes de seu domínio e um conjunto de tipos de dados (predefinido) de seu alcance, por exemplo, a propriedade de dado `nameSPLUsed` possui como classe de domínio `ExperimentSPL` e como alcance um `xsd:string`. A Figura - 3.6 apresenta a definição de propriedade de dado chamada `nameSPLUsed`, neste momento a definição é composta pelo nome da propriedade e seus relacionamentos ((Domains) e (Ranges) no painel `Description: nameSPLUsed` do lado direito inferior).

Figura 3.5: Definição da propriedade de objeto documentation no Protégé



Fonte: o Autor

Artefato gerado pelo Protégé

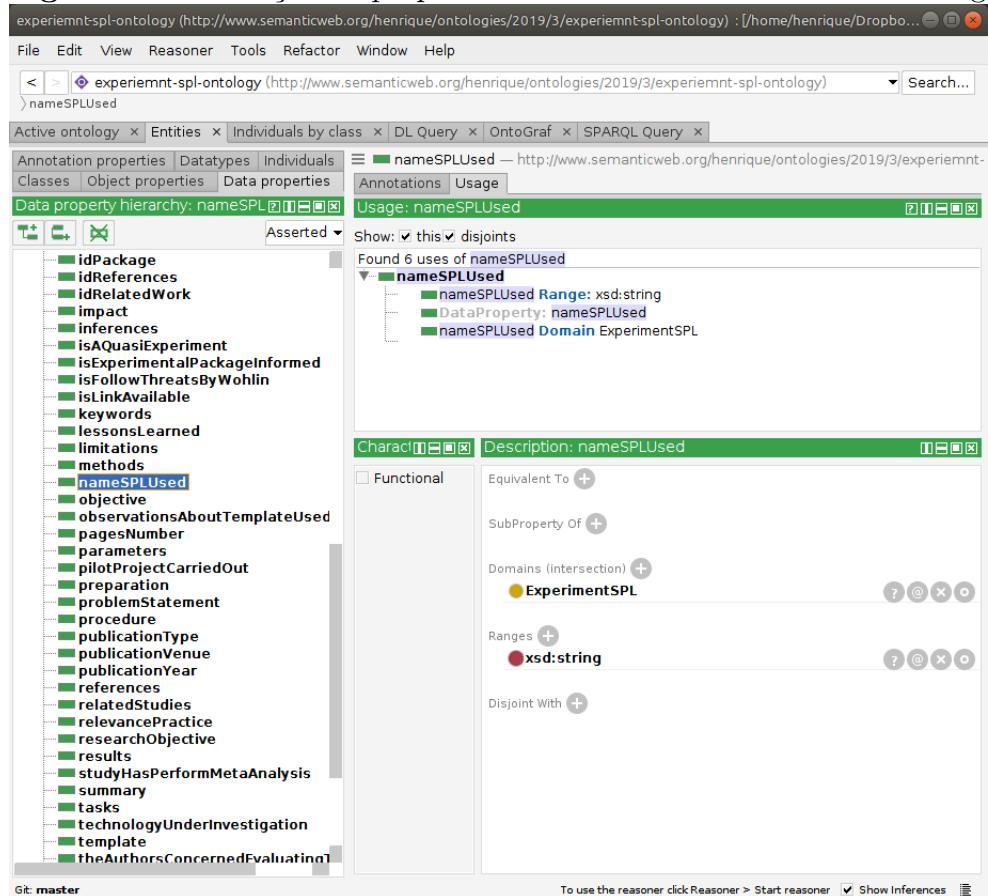
Ao final, o Protégé gera um arquivo *.owl* contendo a definição da ontologia, no . A Figura - 3.7 fornece uma visão geral no formato de grafo do projeto, contendo todas as classes. Foi usado a ferramenta WebVOWL¹ (Lohmann et al., 2016) para gerar essa visão.

3.3.2 Povoamento com Python

A próxima etapa após a modelagem da ontologia foi inserir os indivíduos na mesma na ontologia para a realização de inferências. Apesar do Protégé ter capacidade para realizar essa operação, optou-se por utilizar um script, visto que, no Protégé o processo de inserir indivíduos na ontologia é realizado de modo manual, por meio do menu **Individuals by Class**, em que é preciso selecionar propriedade por propriedade para cada indivíduo. Sabendo que cada indivíduo tem 84 propriedades de dados mais oito propriedade de

¹<http://www.visualdataweb.de/webowl/>

Figura 3.6: Definição da propriedade de dado nameSPLUsed no Protégé



Fonte: o Autor

objetos, somando um total de 92 relacionamentos para cada indivíduo. Portanto, sabendo que tem-se aproximadamente duzentos indivíduos a serem inseridos em uma carga inicial, seria aproximadamente mais de 184.000 operações manuais no Protégé.

Dada essa estimativa, foi escolhida a linguagem de programação Python por fornecer bibliotecas prontas, para manipulação de dados em planilhas, Pandas (McKinney, 2010) e para ontologia em arquivos .owl OwlReady2 (Lamy, 2017).

Script

Para o ambiente de desenvolvimento do script, foi utilizado outra ferramenta do Python chamada Jupyter Notebook (Pérez e Granger, 2007) para auxiliar na execução e validação de código. O processo do script se assemelha a um processo de ETL (*Extract Transform Load*) (Denney et al., 2016), em que extraíram-se os dados originais da planilha (Furtado, 2018), e manipulam-se estes dados para inserir na ontologia, seguindo a modelagem inicial construída no Protégé.

Para leitura dos dados da planilha foi usado a biblioteca Pandas, que retorna um objeto do tipo *Dataframe*, que é a representação da própria planilha no ambiente de programação Python.

Os dados da planilha estão estruturados na seguinte forma: cada linha representa um estudo encontrado no mapeamento sistemático e cada coluna representa um dado extraído deste artigo.

Com relação ao *Experiment*: *ID*, *Title*, *Authorship*, *Publication year*, *Publication type*, *Publication venue*, *Pages number*, *dados de Experimentos em LPS: Is it Real or Academic SPL?*, *SPL Name used*, *Was the SPL source used informed? (Y/N) (If yes, which one?)*.

Com relação a *Documentation*: *Does it use template? (Y/N)?*, *If yes, what template?*, *Observations about the template used*:

- *Abstract*: *Objective (What is the question addressed with this research?)*, *Abstract - Background (Why is this research important?)*, *Methods (What is the statistical context and methods applied?)*, *Results (What are the main findings? Practical implications?)*, *Limitations (What are the weakness of this research?)*, *Conclusions (What is the conclusion?)* e *Keywords*;
- *Introduction Section*: *Problem statement (What is the problem? Where does it occur? Who has observed it? Why is it important to be solved?)*, *Research objective (GQM) (What is the research question to be answered by this study?)*, *Context (What information is necessary to understand whether the research relates to a specific situation (environment)?)*;
- *Related Work Section*: *Technology under investigation (What is necessary for a reader to know about the technology to reproduce its application?)*, *Alternative technologies (How does this research relate to alternative technologies? What is the control treatment?)*, *Related studies (How this research relates to existing research (studies)? What were the results from these studies?)*, *Relevance to practice (How does it relate to state of the practice?)*; e
- *Conclusions and Future Work Section*: *Summary (The purpose of this section is to provide a concise summary of the research and its results as presented in the former sections)*, *Impact (Description of impacts with regard to cost, schedule, and quality, circumstances under which the approach presumably will not yield the expected benefit)*, *Future Work (What other experiments could be run to further investigate the results yielded or evolve the Body of Knowledge)*.

Com relação ao *Experiment Planning*:

- *Goals* (*Formalization of goals, refine the important constructs of the experiment's goal*);
- *Experimental Units* (*From which population will the sample be drawn? How will the groups be formed (assignment to treatments)? Any kind of randomization and blinding has to be described?*);
- *Experimental Material* (*Which objects are selected and why?*);
- *Tasks* (*Which tasks have to be performed by the subjects?*);
- *Hypotheses* (*What are the constructs and their operationalization? They have to be traceable derived from the research question respectively the goal of the experiment*);
- *Parameters* (*What are the constructs and their operationalization? They have to be traceable derived from the research question respectively the goal of the experiment*);
- *Variables* (*What are the constructs and their operationalization? They have to be traceable derived from the research question respectively the goal of the experiment*);
- *Experiment Design* (*What type of experimental design has been chosen?*);
- *Procedure* (*How will the experiment (i.e., data collection) be performed? What instruments, materials, tools will be used and how?) Analysis Procedure* (*How will the data be analyzed?*);
- *Is it a quasi-experiment? (Y/N) If yes, is it explicit in the study?, Is it an Original or Replicated Experiment?*;
- *How was the selection of participants/experimental objects? - Simple random sampling; - Systematic sampling; - Stratified random sampling; - Convenience sampling; - Quota sampling, Context of the experiment (in vivo, in vitro, ...)*;
- *Design Experimental:* - One factor with two treatments; - One factor with more than two treatments; - Two factors with two treatments; - More than two factors each with two treatments, SPL artifact used; e
- *Context Selection (Off-line vs. on-line, Student vs. professional, Toy vs. real problems, Specific vs. general).*

Com relação a *Execution (Operation)*: *Preparation* (*What has been done to prepare the execution of the experiment (i.e., schedule, training), Deviations from the Plan* (*Describe any deviations from the plan, e.g., how was the data collection actually performed?*)), *The pilot project was carried out?* (Y/N) *If Yes, how many?*.

Com relação a *Analysis (Analysis and Interpretation)*: *Descriptive statistics* (*What are the results from descriptive statistics?*), *Data set preparation* (*What was done to prepare the data set, why, and how?*), *Hypothesis testing* (*How was the data evaluated and was the analysis model validated?*), *Do it have qualitative analysis of the experiment?* (Y/N), *If yes, what qualitative analysis was performed?*, *How the data has been analyzed?* (Ex: *Correlation, Hypothesis Test, meta-analysis*), *Is the conclusion of the experiment analysis based on P-value?* (Y/N), *Did the study perform meta-analysis?* (Y/N).

Com relação a *Discussion*: *Evaluation of Results and Implications* (*Explain the results and the relation of the results to earlier research, especially those mentioned in the Background section*), *Threats to Validity* (*How is validity of the experimental results assured? How was the data actually validated?*) (*Follow are the 4 threats proposed by Wohlin: internal, external, construct and conclusion?* (Y/N)), *Inferences* (*Inferences drawn from the data to more general conditions*), *Lessons learned* (*Which experience was collected during the course of the experiment*), *Threats to validity in SPL*.

Com relação a *Evaluation*: *The authors were concerned with evaluating the quality of the experiments?* (Y/N).

Com relação a *Package*: *Is the experimental package informed?* (Y/N) (*If yes, what URL? And the link is still available?* (Y/N))

Outros dados: *Acknowledgements Section* (*Sponsors, participants, and contributors who do not fulfil the requirements for authorship should be mentioned*), *References Section* (*All cited literature has to be presented in the format requested by the publisher*), *Appendices Section* (*Experimental materials, raw data, and detailed analyses, which might be helpful for others to build upon the reported work should be provided*).

Manipulação dos Dados

Para inserir os indivíduos na OntoExper-SPL foi preciso manipular os dados da planilha com as seguintes operações.

Primeira operação: separação (*split*) de dados de uma única coluna para duas propriedades de dados da ontologia. Este caso ocorreu para os seguintes dados, as propriedades e prefixadas com “_” foram tratadas no próximo passo:

- Was the SPL source used informed? (Y/N) (If yes, which one?) para `_informedSPL` e `sourceSPL`;
- The pilot project was carried out? (Y/N) If Yes, how many? para `_hasPilot` e `_howManyPilot`;
- Is it a quasi-experiment? (Y/N) If yes, is it explicit in the study? para `_hasQuasiExperiment` e `quasiExperiment`;
- Threats to Validity (How is validity of the experimental results assured? How was the data actually validated?) (Follow are the 4 threats proposed by Wohlin: internal, external, construct and conclusion? (Y/N))) para `_hasThreatsValidityByWolin` e `threatsValidity`.

Segunda operação: transformação de dados booleanos, strings vazias e números. Foram desenvolvidos três funções de conversão, (i) `convert.ToBoolean`, (ii) `convertStringEmpty` e (iii) `convertToNumber`. Este caso ocorreu para os seguintes dados:

- Does it use template? (Y/N)? para `useTemplate` aplicando o método `convert.ToBoolean`;
- If yes, what template? para `template` aplicando o método `convertStringEmpty`;
- Do it have qualitative analysis of the experiment? (Y/N) para `hasQualitativeAnalysis` aplicando o método `convert.ToBoolean`;
- Did the study perform meta-analysis? (Y/N) para `hasMetaAnalysis` aplicando o método `convert.ToBoolean`;
- `_informedSPL` para `informedSPL` aplicando o método `convert.ToBoolean`;
- `_hasQuasiExperiment` para `hasAQuasiExperiment` aplicando o método `convert.ToBoolean`;
- `_hasPilot` para `hasPilot` aplicando o método `convert.ToBoolean`;
- `_howManyPilot` para `howManyPilot` aplicando o método `convertToNumber`;
- `_hasThreatsValidityByWolin` para `hasThreatsValidityByWolin` aplicando o método `convert.ToBoolean`;
- `_hasPackage` para `hasExperimentalPackage` aplicando o método `convert.ToBoolean`;

Terceira operação: separação do conjunto de dados com informação explícita de LPS. Separou-se o conjunto total de estudos em um conjunto que contém informações explícitas de LPS, e outro conjunto de artigos que não contêm informação explícita de LPS para criação dos indivíduos conforme as classes modeladas na ontologia com atributos pertinentes a LPS.

Quarta operação: padronização de dados constantes e faltantes. Foi preciso padronizar os dados e em casos faltantes foi atribuído um padrão. Isso ocorreu para os seguintes dados:

- *Is it Real or Academic SPL?* foi definido o seguinte conjunto de dados [REAL, ACADEMY] sendo o padrão "ACADEMY";
- *Is it an Original or Replicated Experiment?* foi definido o seguinte conjunto de dados [ORIGINAL, REPLICATED] sendo o padrão "REPLICATED"
- *How was the selection of participants/experimental objects?* - Simple random sampling; - Systematic sampling; - Stratified random sampling; - Convenience sampling; - Quota sampling. foi definido o seguinte conjunto de dados [SIMPLE_RANDOM_SAMPLING, SYSTEMATIC_SAMPLING, STRATIFIED_RANDOM_SAMPLING, CONVENIENCE_SAMPLING, QUOTA_SAMPLING] sendo o padrão "CONVENIENCE_SAMPLING";
- *Context of the experiment (in vivo, in vitro, ...)* foi definido o seguinte conjunto de dados [IN_VIVO, IN_VITRO] sendo o padrão "IN_VITRO";
- *Design Experimental:* - One factor with two treatments; - One factor with more than two treatments; - Two factors with two treatments; - More than two factors each with two treatments. foi definido o seguinte conjunto de dados [ONE_FACTOR_WITH_TWO_TREATMENTS, ONE_FACTOR_WITH_MORE_THAN_TWO_TREATMENTS, TWO_FACTORS_WITH_TWO_TREATMENTS, MORE_THAN_TWO_FACTORS_EACH_WITH_TWO_TREATMENTS] sendo o padrão "ONE_FACTOR_WITH_TWO_TREATMENTS";
- *Context Selection (Off-line vs. on-line, Student vs. professional, Toy vs. real problems, Specific vs. general)* foi definido o seguinte conjunto de dados [OFFLINE, ONLINE, STUDENT, PROFESSIONAL, TOY, REAL_PROBLEMS, SPECIFIC, GENERAL] sendo o padrão "GENERAL";

Inserção do indivíduos e criação dos relacionamentos

Inicialmente executa a leitura do modelo da ontologia gerada pelo Protégé (arquivo .owl) por meio da biblioteca Owlready2, com isso tem-se um objeto na programação que representa o modelo, em que partir dele podemos executar operações de inserção de indivíduos e outras operações na ontologia no ambiente de programação Python.

O processo de inserção dos dados extraídos da planilha se dá por percorrer linha a linha da planilha e criar os indivíduos um a um de cada classe modelada na ontologia com seus respectivos atributos. Para isso foi necessário criar vários métodos a fim de facilitar a compreensão do script.

Foram criados dois laços para percorrer os dois subconjuntos de dados, um com informações explícitas de LPS e outro sem. O primeiro laço sobre o conjunto de LPS, invoca a seguinte sequência de métodos: *registreExperimentSPL > registreExperimentPlanningSPL > registreDiscussionSPL > registerCommons*. A Listagem 3.1 apresenta este laço. O segundo laço sobre o conjunto sem informações de LPS, invoca a seguinte sequência de métodos: *registreExperiment > registreExperimentPlanning > registreDiscussion > registerCommons*. A Listagem 3.2 apresenta este laço.

Listing 3.1: Primeiro Laço

```
for idx, row in df_spl.iterrows():
    experimentSPL = registreExperimentSPL(idx, row)

    experimentPlanningSPL =
        registreExperimentPlanningSPL(idx, row)
    experimentPlanningSPL.experiment.append(experimentSPL)

    discussion = registreDiscussionSPL(idx, row)
    discussion.experiment.append(experimentSPL)

    registerCommons(experimentSPL, idx, row)
```

Listing 3.2: Segundo Laço

```
for idx, row in df_exp.iterrows():
    experiment = registreExperiment(idx, row)

    experimentPlanning = registreExperimentPlanning(idx, row)
    experimentPlanning.experiment.append(experiment)
```

```

discussion = registreDiscussion(idx, row)
discussion.experiment.append(experiment)

registerCommons(experiment, idx, row)

```

O método *registreExperimentSPL* executa o método *registreExperimentCommons* e retorna uma instância de indivíduo da ontologia da classe `onto.ExperimentSPL`.

O método *registreExperiment* executa o método *registreExperimentCommons* e retorna uma instância de indivíduo da ontologia da classe `onto.Experiment`.

O método *registreExperimentCommons* recebe uma instância tanto de `onto.ExperimentSPL` ou `onto.Experiment` e atribui as variáveis comuns para ambas as classes.

O método *registreExperimentPlanningSPL* executa o método *registreExperimentPlanningCommons* e retorna uma instância de indivíduo da ontologia da classe `onto.ExperimentPlanningSPL`.

O método *registreExperimentPlanning* executa o método *registreExperimentPlanningCommons* e retorna uma instância de indivíduo da ontologia da classe `onto.ExperimentPlanning`.

O método *registreExperimentPlanningCommons* recebe uma instância tanto de `onto.ExperimentPlanningSPL` ou `onto.ExperimentPlanning` e atribui as variáveis comuns para ambas as classes.

O método *registreDiscussionSPL* executa o método *registreDiscussionCommons* e retorna uma instância de indivíduo da ontologia da classe `onto.DiscussionSPL`.

O método *registreDiscussion* executa o método *registreDiscussionCommons* e retorna uma instância de indivíduo da ontologia da classe `onto.Discussion`.

O método *registreDiscussionCommons* recebe uma instância tanto de `onto.DiscussionSPL` ou `onto.Discussion` e atribui as variáveis comuns para ambas as classes.

O método *registerCommons* que recebe uma instância tanto de `onto.ExperimentSPL` ou `onto.Experiment` e executa a sequência de métodos: *registreDocumentation* - que retorna uma instância de `onto.Documentation` e atribui a instância de `experiment` a ela > *registreAbstract* - que retorna uma instância de `onto.Abstract` e atribui a instância de `documentation` a ela > *registreIntroduction* - que retorna uma instância de `onto.Introduction` e atribui a instância de `documentation` a ela > *registreRelatedWork* - que retorna uma instância de `onto.RelatedWork` e atribui a instância de `documentation` a ela > *registreConclusionsFutureWork* - que retorna uma instâ-

cia de `onto.ConclusionsFutureWork` e atribui a instância de `documentation` a ela > `registreExecutionSection` - que retorna uma instância de `onto.ExecutionSection` e atribui a instância de `experiment` a ela > `registreAnalysis` - que retorna uma instância de `onto.Analysis` e atribui a instância de `experiment` a ela > `registreAcknowledgements` - que retorna uma instância de `onto.Acknowledgements` e atribui a instância de `experiment` a ela > `registreReferences` - que retorna uma instância de `onto.References` e atribui a instância de `experiment` a ela > `registreAppendices` - que retorna uma instância de `onto.Appendices` e atribui a instância de `experiment` a ela > `registreEvaluation` - que retorna uma instância de `onto.Evaluation` e atribui a instância de `experiment` a ela > `registrePackage` - que retorna uma instância de `onto.Package` e atribui a instância de `experiment` a ela.

Artefato e validação

Ao final dos dois laços tem-se o objeto que representa a ontologia populado com os indivíduos, e por meio da biblioteca Owlready2 foi gerado um novo arquivo .owl contendo a modelagem da ontologia mais a população de indivíduos. O arquivo da modelagem inicial contém aproximadamente 66 kB, e o arquivo populado contém aproximadamente 5 MB.

No final do script existe um passo de validação onde confere-se a quantidade de linhas da planilha com a quantidade de indivíduos inserido na ontologia, que pode ser visto na Listagem 3.3.

Listing 3.3: Passo de Validação

```
assert len(onto.Experiment.instances()) == df.shape[0]
```

3.4 Exemplo de Aplicação

Foi criado um exemplo simples de consulta que retorna a quantidade dos templates experimentais usados pelos indivíduos da ontologia. Foi utilizada a tecnologia SPARQL para executar a consulta.

SPARQL é uma linguagem de consulta e um protocolo para acesso a RDF elaborado pelo *W3C RDF Data Access Working Group*¹. Como uma linguagem de consulta, SPARQL é orientada a dados de forma que só consulta as informações presentes nos modelos, não há inferência propriamente dita nesta linguagem de consulta (Jena, 2007)

¹<https://www.w3.org/2003/12/swa/dawg-charter>

Listing 3.4: Consulta SPARQL exemplo

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://www.semanticweb.org/henrique/ontologies/
          2019/3/onto-exper-lps#>
SELECT ?template (count(?template) as ?count)
WHERE {
    ?doc rdf:type :Documentation .
    ?doc :template ?template .
}
GROUP BY ?template
  
```

A consulta SPARQL é apresentada pela Listagem 3.4 em que são definidos quais prefixos RDF será usada, bem como a OntoExper-SPL. A primeira linha realiza contagem do termo *template*. O bloco *WHERE* aplica a restrição da consulta, nesta restrição só buscam-se elementos RDF do tipo *Documentation* e que possuem a propriedade *template*. Por fim, a cláusula *GROUP BY* realiza o agrupamento da restrição, assim podendo executar a operação *count* da primeira linha.

Esse exemplo percorre uma classe (*Documentation*) das 24 classes existentes na ontologia, uma propriedades de dados (*template*) de 87 propriedades de dados. A formula descrita em 3.1, estima que nesse exemplo foi usando 0,0004% da capacidade de resposta que o modelo permite. Esse cálculo verifica a possibilidades de caminhos entre classes e propriedades da ontologia.

$$\% \text{ uso da ontologia} = \% \text{ classe} * \% \text{ prop de objeto} * \% \text{ prop de dados} \quad (3.1)$$

Esse exemplo simples de consulta mostrou como se pode criar mecanismos de inferência no modelo de ontologia proposto neste trabalho. Dessa forma, nota-se que é possível extrair informações sobre experimentos em LPS usando a OntoExper-SPL.

3.5 Avaliação Preliminar

Para essa avaliação preliminar da proposta de ontologia, foram adotadas duas estratégias: (i) uma para avaliar as possíveis armadilhas, e (ii) outra para identificar falhas no modelo.

Foi utilizada a ferramenta OOPS! para gerar a avaliação do modelo de ontologia proposto. A ferramenta ajuda a detectar algumas das armadilhas mais comuns que

aparecem ao desenvolver ontologias (Poveda-Villalón et al., 2014). Seguem os pontos de avaliação da ferramenta:

- P01. Criando elementos polissêmicos (ou polissêmica lexical, é o fato de uma determinada palavra ou expressão adquirir um novo sentido);
- P02. Criando sinônimos como classes;
- P03. Criando o relacionamento “is” em vez de usar “rdfs: subClassOf”; “rdf: type” ou “owl: sameAs”;
- P04. Criando elementos de ontologia não conectados;
- P05. Definindo relações inversas erradas;
- P06. Incluindo ciclos em uma hierarquia de classes;
- P07. Mesclar diferentes conceitos na mesma classe;
- P08. Anotações ausentes;
- P09. Informações de domínio ausentes;
- P10. Desarticulação em falta;
- P11. Domínio ou intervalo ausente nas propriedades;
- P12. Propriedades equivalentes não explicitamente declaradas;
- P13. Relações inversas não explicitamente declaradas;
- P14. Uso indevido de “owl: allValuesFrom”;
- P15. Usando “alguns não” no lugar de “não alguns”;
- P16. Usando uma classe primitiva no lugar de uma definida;
- P17. Superespecialização de uma hierarquia;
- P18. Superespecialização do domínio ou intervalo;
- P19. Definir vários domínios ou intervalos nas propriedades;
- P20. Uso indevido de anotações de ontologia;
- P21. Usando uma classe diversa;

- P22. Usando diferentes convenções de nomenclatura na ontologia;
- P23. Duplicando um tipo de dados já fornecido pela linguagem de implementação;
- P24. Usando definições recursivas;
- P25. Definir um relacionamento como inverso de si mesmo;
- P26. Definindo relações inversas para uma simétrica;
- P27. Definir propriedades equivalentes erradas;
- P28. Definindo relacionamentos simétricos errados;
- P29. Definindo relacionamentos transitivos errados;
- P30. Classes equivalentes não explicitamente declaradas;
- P31. Definir classes equivalentes erradas;
- P32. Várias aulas com o mesmo rótulo;
- P33. Criando uma cadeia de propriedades com apenas uma propriedade;
- P34. Classe sem tipografia;
- P35. Propriedade não tipificada;
- P36. URI contém extensão de arquivo;
- P37. Ontologia não disponível na Web;
- P38. Nenhuma declaração de ontologia OWL;
- P39. Namespace ambíguo;
- P40. Sequestro de namespace; e
- P41. Nenhuma licença declarada.

Apesar da ferramenta OOPS! ter 41 pontos de avaliação ela executa apenas 34 pontos semi-automaticamente, pois os outros dependem de domínio específico da ontologia e eles encorajam os usuários a melhorarem a ferramenta. O resultado dado pela ferramenta sugere como os elementos da ontologia poderiam ser modificados para melhorar sua. No

entanto, nem todas as armadilhas identificadas devem ser interpretadas como falha, mas como sugestões que devem ser revisadas manualmente em alguns casos.

Essa avaliação pode ajudar a descobrir falhas que foram escondidos por causa da falta de informação preliminar. Por exemplo, algumas armadilhas são detectadas pela comparação de domínios e intervalos em propriedades, se não estiverem definidas, as armadilhas não podem ser identificadas. Nesse sentido, corrigindo a armadilha “Falta do domínio ou intervalo em propriedades” faz com que a ferramenta pare de encontrar outras armadilhas, como por exemplo, “Definir relações simétricas que não possuem o mesmo domínio e alcance”.

A ferramenta elenca os resultados de cada armadilha como:

- ***critico***: Corrigir a armadilha é crucial. Caso contrário, isso poderia afetar a consistência, raciocínio, aplicabilidade, etc. da ontologia;
- ***importante***: Embora não seja crítico para a função de ontologia, é importante corrigir este tipo de armadilha;
- ***baixo***: Não é realmente um problema, mas ao consertá-lo, tornaremos a ontologia mais agradável.

A Tabela - 3.1 apresenta o resumo do resultado ao rodar nosso proposta de modelo de ontologia na ferramenta OOPS!.

Tabela 3.1: Resumo dos resultado da avaliação executada por OOPS!

Armadilha	Descrição	Nível de Criticidade
P08	Anotações ausentes em 119 casos	<i>baixo</i>
P10	Falta de disjunção na ontologia *	<i>importante</i>
P13	Relações inversas não declaradas explicitamente em 8 casos	<i>baixo</i>
P19	Definindo vários domínios ou intervalos nas propriedades em 6 casos	<i>critico</i>
P41	Nenhuma licença declarada na ontologia *	<i>importante</i>

Fonte: o Autor

*Armadilha que se aplica à ontologia em geral, em vez de elementos específicos.

No caso P08, essa armadilha consiste em criar um elemento de ontologia e não fornecer anotações legíveis a ele. Consequentemente, os elementos de ontologia não possuem propriedades de anotação que os identificam (por exemplo, `rdfs: label`, `lemon: LexicalEntry`, `skos: prefLabel` ou `skos: altLabel`) ou que os definem

(por exemplo, `rdfs: comment` ou `dc: description`). Essa armadilha está relacionada às diretrizes fornecidas em (Vrandecic, 2010).

No caso P10, a ontologia não possui axiomas desarticulados entre classes ou entre propriedades que devem ser definidas como disjuntas. Essa armadilha está relacionada com as orientações fornecidas em (Gómez-Pérez, 2004), (Noy et al., 2001) e (Rector et al., 2004).

No caso P13, essa armadilha aparece quando qualquer relacionamento (exceto aqueles que são definidos como propriedades simétricas usando `owl: SymmetricProperty`) não tem uma relação inversa (`owl: inverseOf`) definida dentro da ontologia. Esta armadilha aparece nos seguintes elementos: `typeSelectionOfParticipants`, `typeExperimentSPL`, `typeExperiment`, `typeDesignExperiment`, `typeContextSelection`, `typeContextExperiment`, `experiment`, `documentation`.

No caso P19, o domínio ou intervalo (ou ambos) de uma propriedade (relacionamentos e atributos) é definido informando mais de uma instrução `rdfs: domain` ou `rdfs: range`. Em OWL vários `rdfs: domain` ou `rdfs: range` axiomas de alcance são permitidos, mas eles são interpretados como conjunção, sendo, portanto, equivalentes ao constructo `owl: intersectionOf`. Essa armadilha está relacionada ao erro comum que aparece ao definir domínios e intervalos descritos em (Rector et al., 2004). Esta armadilha aparece nos seguintes elementos: `documentation`, `experiment`, `typeContextExperiment`, `typeDesignExperiment`, `typeExperiment`, `typeSelectionOfParticipants`.

No caso P41, os metadados de ontologia omitem informações sobre a licença que se aplica à ontologia.

Dada a análise da ferramenta OOPS! o próximo passo será corrigir as armadilhas apresentada nesta avaliação. Esses pontos de melhorias serão tratados na Seção 4.9.

3.6 Considerações Finais

Este capítulo apresentou a elaboração da proposta de ontologia OntoExper-SPL. foram discutidas quatro partes importantes neste capítulo, (i) concepção do modelo da OntoExper-SPL, (ii) construção do modelo e povoamento da OntoExper-SPL, (iii) um exemplo de inferência sobre a ontologia com SPARQL e (iv) uma avaliação preliminar apontando 41 pontos de possíveis falhas na OntoExper-SPL.

Em relação à concepção do modelo, foi apresentado uma tipologia na qual se descreve como uma ontologia de domínio semi-formal, uma ontologia de especificação que serve tanto para modelagem do conhecimento como para aplicação de domínio. Para concepção

do modelo, também foi elaborado um modelo conceitual clusterizado sobre os metadados, bem como um diagrama de classe, que serviram de apoio para elaboração do modelo.

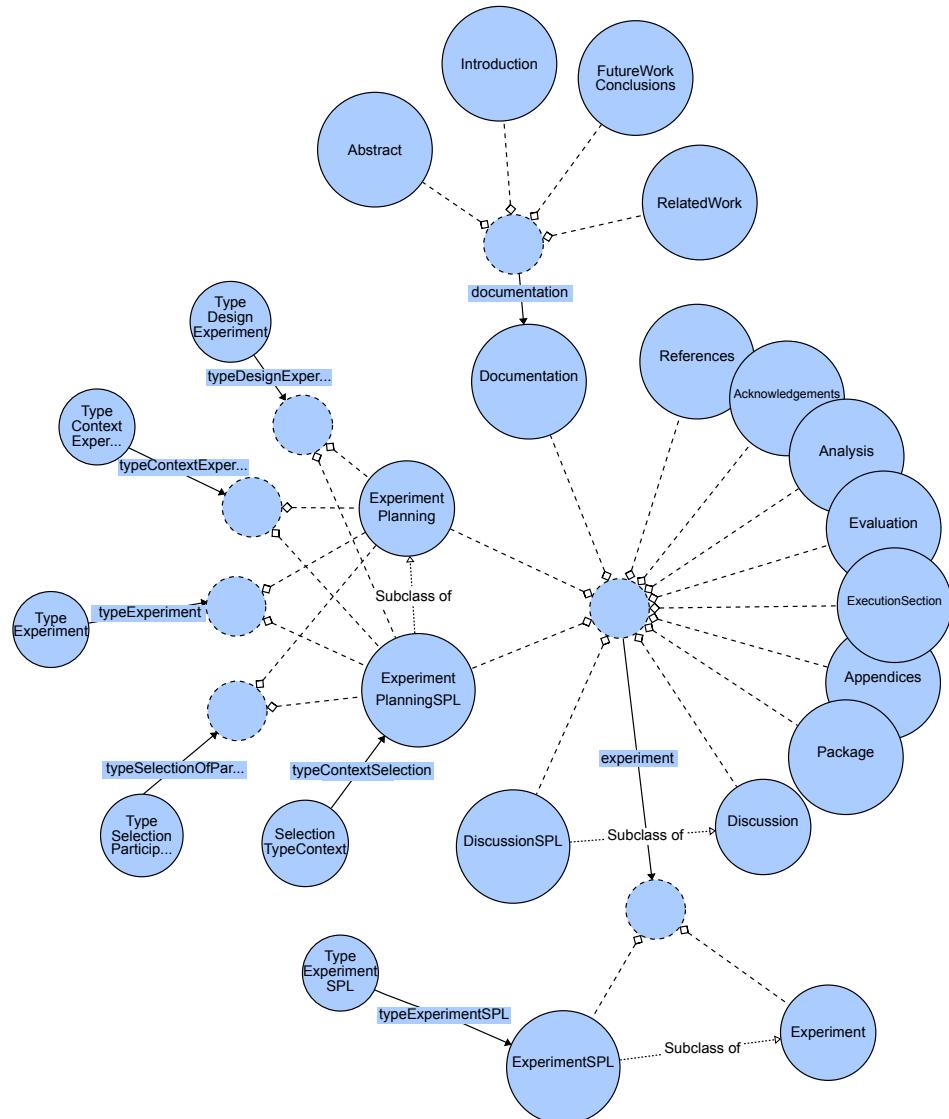
Para construção do modelo, além do modelo conceitual e do diagrama de classe, foi utilizado a tecnologia OWL e a ferramenta Protégé como instrumentos efetivos de apoio, por meio deste, foi possível entregar um artefato do tipo OWL contendo toda modelagem. Ainda nessa seção foi elaborado um programa automatizada que, utilizando os metadados pré definidos, popula o modelo proposto como indivíduos da ontologia, este programa também é armazenado como um artefato resultado deste trabalho de mestrado.

Na seção posterior, foi tratado um exemplo de inferência sobre a OntoExer-SPL. Neste exemplo, usando a tecnologia SPARQL, foi possível consultar do modelo, indivíduos com determinados critérios, provando que a realização de inferência no modelo funciona.

Por fim, tem-se uma breve avaliação preliminar do modelo proposto, em que foi aplicada a ferramenta OOPS! que avalia 41 pontos de possíveis falhas na ontologia. Os pontos de falhas relatados pela ferramenta, serão tratados na seção 4.9.

No próximo capítulo, será apresentada uma avaliação empírica da OntoExer-LPS como um estudo quali/qualitativo.

Figura 3.7: Grafo da modelagem da ontologia gerado pela ferramenta WbVOWL



Fonte: o Autor

Avaliação Empírica da OntoExper-SPL

4.1 Considerações Iniciais

Este tópico apresenta um estudo empírico realizado para avaliar a OntoExper-SPL, com base em um questionário contendo onze questões, oito com critérios de qualidade no estilo escala Likert (Allen e Seaman, 2007), uma avaliação de frequência da escala Likert, teste estatístico e correlação entre os oito critérios.

4.2 Definição do Estudo

O propósito principal deste estudo é caracterizar a OntoExper-SPL. Baseado no modelo *Goal-Question-Metric* (GQM) (Basili e Rombach, 1988), este estudo tem por objetivo:

Avaliar a ontologia OntoExper-SPL

Com o propósito de caracterizar a sua viabilidade

Referente à critérios de avaliação definidos

Do ponto de vista de especialistas em LPS e Ontologias

No contexto de pesquisadores das seguintes instituições: Universidade Estadual de Londrina (UEL), Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC-USP), Pontifícia Universidade Católica do Paraná (PUC-PR), Universidade Estadual de Maringá (UEM), Universidade Tecnológica Federal do Paraná (UTFPR), Universidade de São Paulo (USP), Sidia Instituto de Ciência e Tecnologia

(SIDIA), Pontifícia Universidade Católica do Rio Grande do Sul (PUC-RS), Instituto Federal do Paraná (IFPR) e Universidade Paranaense (Unipar).

4.3 Planejamento do Estudo

O planejamento do estudo é composto das seguintes etapas:

- **Seleção dos Participantes:** os especialistas foram convidados de forma conveniente não-probabilística, sendo pesquisadores da área de LPS e Ontologia. Dos participantes, 4 são pós-doutor (23,5%), 6 são doutores (35,3%), 6 são mestres (35,3%), 1 é mestrandos (5,9%).
- **Treinamento:** por causa do nível de conhecimento dos participantes não foi necessária a realização desta etapa.
- **Instrumentação:** todos os especialistas receberam os seguintes documentos: i) um documento contendo um breve descriptivo da avaliação, com os link para ferramentas e tecnologias necessárias; (ii) uma cópia do Instrumento de Avaliação caracterizado por um questionário - Google Forms¹; e (iii) uma cópia dos metadados da ontologia, compostos de: um arquivo do tipo OWL do modelo da OntoExper-SPL, um arquivo do tipo OWL da OntoExper-SPL povoada com 174 indivíduos, um arquivo do tipo planilha do Excel contendo os dados originais dos experimentos, a cópia da Figura - 3.3, a cópia da Figura - 3.2, a cópia da Figura - 3.1 e a cópia da Figura - 3.7. A instrumentação completa pode ser encontrada no Apêndice E.

A instrumentação fornecida aos especialistas foi suficiente para responder o questionário da avaliação.

Critérios de Avaliação

Os oito critérios de qualidade adotados se baseiam nos trabalhos de *Ontology Evaluation* de Vrandecic (2010):

- **Precisão:** um critério que determina se os axiomas da ontologia estão em conformidade com o conhecimento das partes interessadas sobre o domínio. Uma maior precisão vem das definições e descrições corretas de classes, propriedades e indivíduos. A correção neste caso pode significar conformidade com “padrões-ouro”

¹<https://www.google.com/forms>

definidos, sejam outras fontes de dados, conceituações ou mesmo realidade (Ceusters e Smith, 2006). introduz uma abordagem para usar a realidade como referência, ou seja, os termos da ontologia captura as partes pretendidas da realidade. Os axiomas devem restringir as possíveis interpretações de uma ontologia para que os modelos resultantes sejam compatíveis com as conceituações dos usuários;

- **Adaptabilidade:** mede até que ponto a ontologia antecipa seus usos. Uma ontologia deve oferecer a base conceitual para uma série de tarefas antecipadas (idealmente, na Web, também deve oferecer a base para tarefas nunca antecipadas). Deve ser possível estender e especializar a ontologia monotonicamente, ou seja, sem a necessidade de remover axiomas (observe que em OWL, a monotonicidade semântica é dada pela monotonicidade sintática, ou seja, para retrair inferências, axiomas explícitos e explícitos precisam ser retraídos). Uma ontologia deve reagir de forma previsível e intuitiva a pequenas mudanças nos axiomas. Deve permitir metodologias para extensão, integração e adaptação, ou seja, incluir metadados necessários. Novas ferramentas e situações inesperadas devem poder usar a ontologia;
- **Clareza:** mede com que eficácia a ontologia comunica o significado pretendido dos termos definidos. As definições devem ser objetivas e independentes do contexto. Os nomes dos elementos devem ser compreensíveis e inequívocos. Uma ontologia deve usar definições em vez de descrições para classes. As entidades devem ser documentadas o suficiente e estar totalmente rotuladas em todos os idiomas necessários. Axiomas complexos devem ser documentados. As escolhas de representação não devem ser feitas para a conveniência da notação ou implementação, ou seja, o viés de codificação deve ser minimizado;
- **Completude:** O critério Completude mede se o domínio de interesse está coberto adequadamente. Todas as perguntas que a ontologia deve poder responder podem ser respondidas. Existem diferentes aspectos da completude: completude no que diz respeito ao idioma (tudo o que é declarado que poderia ser declarado usando o idioma fornecido?), Completude no que diz respeito ao domínio (todos os indivíduos estão presentes, todos os conceitos relevantes são capturados?), Completude com em relação aos requisitos de aplicação (todos os dados necessários estão presentes?), etc. A abrangência também abrange a granularidade e a riqueza da ontologia;
- **Eficiência Computacional:** mede a capacidade das ferramentas usadas de trabalhar com a ontologia, em particular a velocidade que os raciocinadores precisam para executar as tarefas necessárias, seja resposta de consulta, classificação ou verificação

de consistência. Alguns tipos de axiomas podem causar problemas para certos racionadores. O tamanho da ontologia também afeta a eficiência da ontologia;

- **Concisão:** determina se a ontologia inclui elementos irrelevantes em relação ao domínio a ser coberto (ou seja, uma ontologia sobre livros, incluindo axiomas sobre leões africanos) ou representações redundantes da semântica. Uma ontologia deve impor um compromisso ontológico mínimo, ou seja, especificar a teoria mais fraca possível. Somente termos essenciais devem ser definidos. As suposições subjacentes da ontologia sobre o domínio mais amplo (especialmente sobre a realidade) devem ser tão fracas quanto possível, a fim de permitir a reutilização interna e a comunicação entre as partes interessadas que se comprometem com diferentes teorias;
- **Consistência:** mede que a ontologia não inclui ou permite contradições. Enquanto a precisão declara a conformidade da ontologia com uma fonte externa, a consistência indica que a própria ontologia pode ser interpretada. A consistência lógica é apenas uma parte dela, mas também as descrições formais e informais na ontologia devem ser consistentes, ou seja, a documentação e os comentários devem estar alinhados com os axiomas. Outros princípios de ordenação podem ser definidos com os quais a ontologia deve ser consistente, como as restrições OntoClean à taxonomia (Guarino e Welty, 2002); e
- **Capacidade Organizacional:** agrupa vários critérios que decidem com que facilidade uma ontologia pode ser implantada dentro de uma organização. Ferramentas, bibliotecas, fontes de dados e outras ontologias usadas restringem a ontologia, e a ontologia deve atender a essas restrições. As ontologias são frequentemente especificadas usando uma metodologia de engenharia de ontologia ou usando conjuntos de dados específicos. Os metadados da ontologia podem descrever as metodologias, ferramentas e fontes de dados aplicadas e a organização. Esses metadados podem ser usados pela organização para decidir se uma ontologia deve ser aplicada ou não.

Escala Likert

A tese de Vrandecic (2010) não expõe uma forma clara para mensurar os oito critérios. Neste ponto foi definido que uma escala Likert poderia auxiliar. Segundo Allen e Seaman (2007) uma escala Likert para gerar bons resultados deve possuir três elementos fundamentais, (i) um negativo, (ii) um neutro e (iii) um positivo.

Os itens Likert são usados para medir as atitudes dos entrevistados em relação a uma pergunta ou afirmação específica. Essas escalas permitem determinar o nível de concordância ou discordância dos respondentes.

Dessa forma, a escala Likert pressupõe que a força e a intensidade da experiência são lineares. Portanto passa de uma concordância total a uma discordância total, assumindo que as atitudes podem ser medidas.

As respostas podem ser oferecidas em diferentes níveis de medição, permitindo escalas de 5, 7 e 9 elementos previamente configurados. Para analisar os dados, codificados esses itens da seguinte forma:

- 1: Discordo totalmente;
- 2: Discordo parcialmente;
- 3: Nem concordo nem discordo (neutro);
- 4: Concordo parcialmente; e
- 5: Concordo totalmente.

4.4 Execução do Estudo

Foi realizado um **Projeto Piloto** com a intenção de avaliar a instrumentação do estudo, um projeto piloto foi conduzido em Outubro de 2019, com um Mestre e um Doutor em Ciências da Computação da Universidade Estadual de Maringá (UEM). Os dados obtidos foram descartados, mas, as considerações sobre erros e melhorias nos questionários foram consideradas.

Procedimentos de Participação: a participação de cada especialista no estudo ocorreu da seguinte forma:

1. o especialista recebe os documentos, via e-mail, que compõem o estudo (Seção 4.3);
2. o especialista faz um estudo prévio dos metadados, esclarece possíveis dúvidas;
3. o especialista faz a leitura e preenche o Formulário de Avaliação - Google Forms, de acordo com as suas experiências.

Os especialistas tiveram um prazo de 40 dias para a finalização do questionário de avaliação. Todo o acompanhamento do estudo foi realizado online.

4.5 Análise dos Resultados

Esta seção apresenta a análise dos resultados com relação às seguintes informações:

- perfil dos especialistas;
- apresentação da frequência da escala Likert;
- apresentação e análise do teste normalidade dos resultados;
- apresentação e análise do teste de hipótese aplicado aos resultados; e
- apresentação e análise da correlação entre os critérios de avaliação;

4.5.1 Perfil dos Especialistas

Foram convidados 40 especialistas em ES, porém apenas 17 aceitaram participar do estudo quantitativo. A Tabela - 4.1 apresenta o perfil dos participantes que realizaram o estudo. Foi possível observar que a média de experiência em anos dos participantes é de 7,2. Destacam-se os especialistas E3 e E12 com 15 anos de experiência e o E1 com 0 anos de experiência. Com relação ao nível de formação, temos quatro Pós-Doutor, seis doutores, 6 mestres, e um graduado.

4.5.2 Frequência Likert

Os dados coletados dos especialistas foram analisados de forma exploratória usando a ferramenta Jupyter-Lab¹ (Kluyver et al., 2016), como linguagem de programação Python juntamente com as bibliotecas Pandas, Matplotlib (Hunter, 2007), SeaBorn (Waskom et al., 2017) e Scipy (Jones et al., 2001–).

A Tabela - 4.2 apresenta a frequência das respostas dos especialistas de cada critério em relação à escala de respostas. As opções de respostas estão mapeadas com os seguintes valores na tabela:

- **%DT:** Discordo Totalmente, valor: 1;
- **%DP:** Discordo Parcialmente, valor: 2;
- **%N:** Nem Concordo nem Discordo (neutro), valor: 3;
- **%CP:** Concordo Parcialmente, valor: 4; e

¹<https://jupyter.org/>

Tabela 4.1: Perfil dos Participantes do Estudo

Especialista (E)	Nível de Formação	Experiência em LPS e/ou Ontologias (em anos)
E1	Pós-Doutorado	0
E2	Pós-Doutorado	10
E3	Pós-Doutorado	15
E4	Mestrado	5
E5	Mestrado	5
E6	Mestrado	3
E7	Doutorado	8
E8	Mestrado	5
E9	Doutorado	10
E10	Doutorado	9
E11	Doutorado	6
E12	Pós-Doutorado	15
E13	Graduação	3
E14	Mestrado	5
E15	Doutorado	12
E16	Mestrado	6
E17	Doutorado	6

Fonte: o Autor

- **%CT:** Concordo Totalmente, valor: 5.

A moda representa o valor mais frequente respondido pelos especialistas. Sendo assim, segue o resultado de cada critério:

- **Precisão** teve a moda 5, com uma frequência de 52,94% para CT;
- **Adaptabilidade** teve a moda 5, com uma frequência de 64,71% para CT;
- **Clareza** teve a moda 4, porém com uma frequência igual 47,06% para CT e CP.;
- **Completude** teve a moda 4, com uma frequência de 35,29% para CP;
- **Eficiência Computacional** teve a moda 3, com uma frequência de 47,06% para N;
- **Concisação** teve a moda 5, com uma frequência de 58,82% para CT;
- **Consistência** teve a moda 5, com uma frequência de 64,71% para CT; e
- **Capacidade Organizacional** teve a moda 5, com uma frequência de 52,94% para CT.

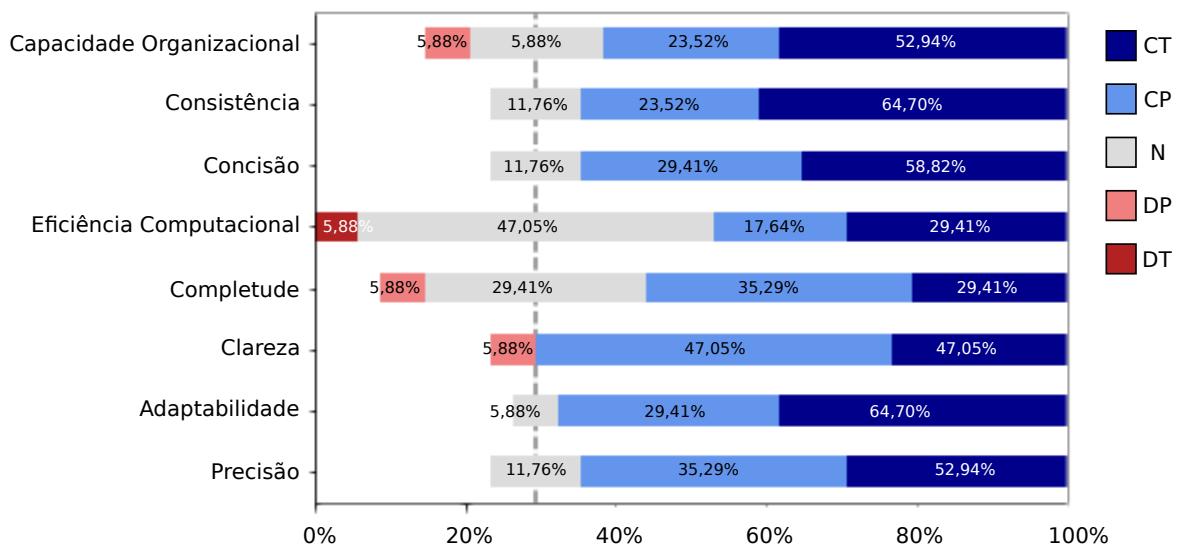
Tabela 4.2: Porcentagem de Frequência Amostral das Respostas dos participantes

	%DT	%DP	%N	%CP	%CT	Moda
Precisão	0(0.00%)	0(0.00%)	2(11.76%)	6(35.29%)	9(52.94%)	5.0
Adaptabilidade	0(0.00%)	0(0.00%)	1(5.88%)	5(29.41%)	11(64.71%)	5.0
Clareza	0(0.00%)	1(5.88%)	0(0.00%)	8(47.06%)	8(47.06%)	4.0
Completude	0(0.00%)	1(5.88%)	5(29.41%)	6(35.29%)	5(29.41%)	4.0
Eficiência Computacional	1(5.88%)	0(0.00%)	8(47.06%)	3(17.65%)	5(29.41%)	3.0
Concisão	0(0.00%)	0(0.00%)	2(11.76%)	5(29.41%)	10(58.82%)	5.0
Consistência	0(0.00%)	0(0.00%)	2(11.76%)	4(23.53%)	11(64.71%)	5.0
Capacidade Organizacional	0(0.00%)	1(5.88%)	3(17.65%)	4(23.53%)	9(52.94%)	5.0

Fonte: o Autor

O gráfico da Figura - 4.1 apresenta a distribuição da frequência em barras. Dessa forma foi possível visualizar que o critério que teve o melhor resultado positivo foi **Adaptabilidade** com as frequências de, 64,71% para CT, 29,41% para CP e 5,88% para N. E o critério que teve o pior resultado negativo foi **Eficiência Computacional** com as frequências de, 29.41% para CT, 17,65% para CP, 47,06% para N e 5,88% para DT.

O critério **Eficiência Computacional** teve o pior desempenho, pois os especialistas E4, E10, E13 e E17 relataram problemas para executar a consulta SPARQL, todos eles marcaram como resposta N para este critério.

Figura 4.1: Frequência das Respostas dos Participantes

Fonte: o Autor

4.5.3 Teste de Normalidade

Para descrever o teste de normalidade dos resultados foi aplicada uma abordagem estatística do método aditivo com uma variável dependente (VD) contínua. A VD é o resultado da somatória do valor dado a cada critério por especialista. A Tabela - 4.3 apresenta esses valores.

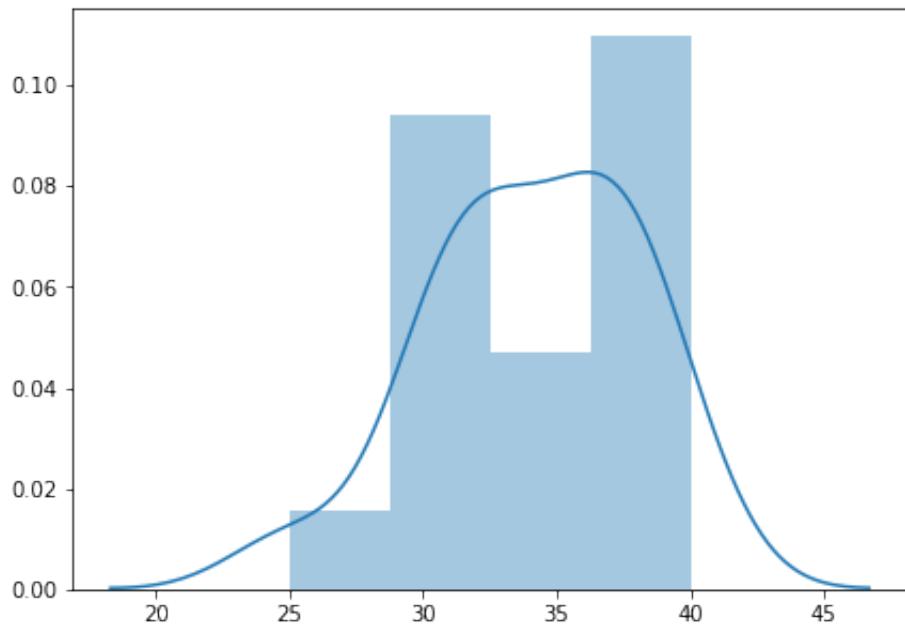
Tabela 4.3: Somatório das Resposta dos Especialistas

Especialista (E)	Somatório dos Critérios
E1	32
E2	37
E3	30
E4	32
E5	34
E6	40
E7	37
E8	37
E9	32
E10	31
E11	30
E12	25
E13	35
E14	34
E15	39
E16	37
E17	38

Fonte: o Autor

Foram plotados esses valores no gráfico de histograma apresentado na Figura - 4.2. É necessário executar o teste de normalidade. Para isso foi usado o teste de Shapiro-Wilk (Shapiro e Wilk, 1965). O teste Shapiro-Wilk testa a hipótese nula de que os dados foram extraídos de uma distribuição normal, considerando que $p\text{-value} > 0,05$.

Figura 4.2: Histograma da Somatória



Fonte: o Autor

O resultado do teste estatístico de Shapiro-Wilk foi 0,94 e o *p-value* foi 0,44. Sendo assim, a distribuição da VD é considerado **normal**.

4.5.4 Teste de Hipótese

Considerando que a amostra de VD segue uma distribuição normal, pode-se aplicar o teste paramétrico de Mann-Whitney (Mann e Whitney, 1947). Para aplicar este teste foi preciso elaborar hipóteses para este questionário. Considera-se que *p-value*: $> 0,05$ pode-se rejeitar H_0 e quando *p-value* $< 0,05$ não se rejeita H_0 .

O primeiro par de hipótese é em relação a **Nível de Formação** dos especialistas:

- **Hipótese Nula (H_0)**: não existe diferença na formação do especialista com relação a VD.
- **Hipótese Alternativa**: existe diferença significativa na formação do especialista com relação a VD.

O resultado do teste de Mann-Whitney foi $u = 0,000$, $p\text{-value} = 0,00000029 < 0,05$, logo é possível rejeitar H_0 e afirmar que, **existe diferenças nos resultado dado o Nível de Formação**.

O outro par de hipótese é em relação a **Experiência em LPS e/ou Ontologias (em anos)** dos especialista:

- **Hipótese Nula (H0):** não existe diferença na experiência do especialista com relação a VD.
- **Hipótese Alternativa:** existe diferença significativa na experiência do especialista com relação a VD.

Para realização deste teste foi necessário criar 3 grupos representativos dos anos de experiência, os grupos são:

- **1:** 0 à 5 anos de experiências;
- **2:** 6 à 10 anos de experiências; e
- **3:** 11 à 15 anos de experiências.

O resultado do teste de Mann-Whitney foi $u = 0,000$, $p\text{-value} = 0,00000027 < 0,05$, logo é possível rejeitar H0 e afirmar que **existe sim diferenças nos resultado dado o Experiência em LPS e/ou Ontologias (em anos)**.

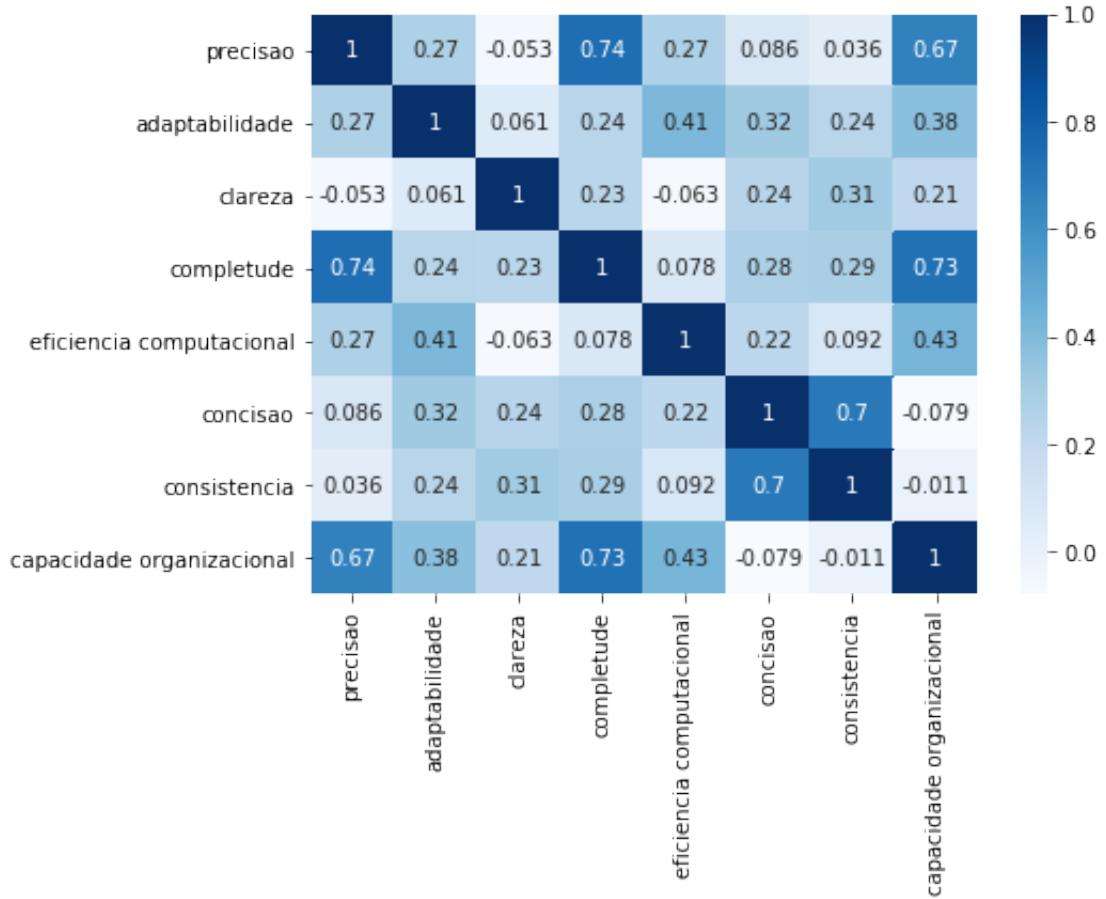
4.5.5 Correlação dos Critérios

Foi aplicado a técnica de correlação de Pearson (Kirch, 2008), pode-se realizar a seguinte interpretação:

- **0.9** para mais ou para menos indica uma correlação muito forte.
- **0.7 a 0.9** positivo ou negativo indica uma correlação forte.
- **0.5 a 0.7** positivo ou negativo indica uma correlação moderada.
- **0.3 a 0.5** positivo ou negativo indica uma correlação fraca.
- **0 a 0.3** positivo ou negativo indica uma correlação desprezível

A Figura - 4.3 apresenta um mapa de calor indicando o grau de correlação entre os critérios de qualidade. Foi possível observar que os critérios com maior grau de correlação foram **Completude** com **Precisão** com o valor de correlação forte **0,74**, **Capacidade Organizacional** com **Completude** com o segundo valor de correlação forte **0,73** e **Consistência** com **Concisação** com o terceiro maior valor de correlação moderada **0,70**.

Figura 4.3: Correlação dos critérios de avaliação.



Fonte: o Autor

Tabela 4.4: Ranking de correlação dos critérios de avaliação

Posição	Critério Y	Critério X	Valor	Interpretação
1	Completude	Precisão	0,74	forte
2	Capacidade Organizacional	Completude	0,73	forte
3	Consistência	Concisão	0,70	forte
4	Capacidade Computacional	Eficiência Computacional	0,43	fraca
5	Eficiência Computacional	Adaptabilidade	0,41	fraca
6	Capacidade Computacional	Adaptabilidade	0,38	fraca
7	Concisão	Adaptabilidade	0,32	fraca
8	Consistência	Clareza	0,31	fraca
9	Consistência	Completude	0,29	desprezível
10	Concisão	Completude	0,28	desprezível
11	Adaptabilidade	Precisão	0,27	desprezível
12	Eficiência Computacional	Precisão	0,27	desprezível
13	Completude	Adaptabilidade	0,24	desprezível
14	Consistência	Adaptabilidade	0,24	desprezível
15	Concisão	Clareza	0,24	desprezível
16	Consistência	Adaptabilidade	0,24	desprezível
17	Completude	Clareza	0,23	desprezível
18	Concisão	Eficiência Computacional	0,22	desprezível
19	Capacidade Computacional	Clareza	0,21	desprezível
20	Consistência	Eficiência Computacional	0,092	desprezível
21	Concisão	Precisão	0,086	desprezível
22	Eficiência Computacional	Completude	0,078	desprezível
23	Clareza	Adaptabilidade	0,061	desprezível
24	Consistência	Precisão	0,036	desprezível
25	Capacidade Organizacional	Consistência	-0,011	desprezível
26	Consistência	Precisão	-0,053	desprezível
27	Eficiência Computacional	Clareza	-0,063	desprezível
28	Capacidade Organizacional	Concisão	-0,079	desprezível

Fonte: o Autor

A Tabela - 4.4 apresenta o *ranking* de correlação dos critérios. Segundo a interpretação de Kirch (2008) só serão consideradas as correlações de 1 a 8 no *ranking*. Dessa forma, pode-se afirmar que:

- quanto melhor a **Completude** melhor a **Precisão**;
- quanto melhor a **Consistência** melhor a **Concisão**;
- quanto melhor a **Capacidade Computacional** melhor a **Eficiência Computacional**;
- quanto melhor a **Eficiência Computacional** melhor a **Adaptabilidade**;
- quanto melhor a **Capacidade Computacional** melhor a **Adaptabilidade**;
- quanto melhor a **Concisão** melhor a **Adaptabilidade**;
- quanto melhor a **Consistência** melhor a **Clareza**; e
- quanto melhor a **Consistência** melhor a **Completude**.

Os valores de correlação negativa não foram o suficiente para serem considerados. Dessa forma, pode-se afirmar não existe critérios com avaliações inversas, como por exemplo, quanto melhor a **Eficiência Computacional** pior a **Concisão**.

4.6 Discussão dos Resultados

Na Seção 4.5.1 foi relatado o perfil dos especialista, pode-se notar que os especialistas possuem em média 7,2 anos de experiência em LPS e/ou Ontologias e que 70,61% são doutores e Pós-doutores. Tal resultado corrobora para o alto grau de especialidades dos participantes.

Durante a Seção 4.5.2 foi apresentado os resultados do estudo a análise dos demonstra que, segundo os especialistas, a onto possui um grau de qualidade relevante. Isso fica claro quando visualizamos que 70,15% das respostas foram “positivas”, ou seja, estão entre CP e CT, o critério **Adaptabilidade** se destacou nessa avaliação. E apenas 2,95% foram “negativas”, ou seja, estão entre DP e DT. Porém o valor de 16,91% no elemento N apresenta um ponto de atenção, vimos que o critério de qualidade que mais afetou este item foi a **Eficiência Computacional**. Estes valores estão destacado na Tabela - 4.5

A Seção 4.5.2 foi aplicado o método aditivo para elaborar uma variável dependente. Por meio do teste de Shapiro-Wilk pode-se identificar e a VD é uma distribuição normal.

Tabela 4.5: Total de respostas positivas, neutras e negativas

	Negativo (DT e DP)	Neutro (N)	Positivo (CP e CT)
Total de respostas	4	23	109
%	2,95%	16,91%	70,15%

Fonte: o Autor

Uma distribuição normal apresenta que os dados estão distribuídos em 68,26% por 1 desvio, um conjunto central, 95,44% por 2 desvios e 99,73% por 3 desvios. Sendo assim essa informação guia a próxima técnica de testes de hipótese.

Logo, na Seção 4.5.4 foram elaboradas duas hipóteses para averiguação por meio de teste estatístico de hipótese. Primeira hipótese **não existe diferença na formação do especialista com relação a VD** que por meio do teste de Mann-Whitney retornou $p\text{-value} < 0,05$, dessa forma rejeitando H_0 , tal resultado demonstra que a VD tem impacto sobre a formação do especialista. Segunda hipótese: **não existe diferença na experiência do especialista com relação a VD**, que por meio do teste de Mann-Whitney retornou o $p\text{-value} < 0,05$, da mesma forma que o anterior, rejeita H_0 , tal resultado demonstra que a VD tem impacto sobre a formação os anos de experiência do especialista.

Por fim foi na Seção 4.5.5 apresentado o nível de correlação entre os critérios, segundo as respostas dos especialistas. Foi possível notar que das 28 possibilidades só foram encontradas correlações relevantes em oito delas. Foi elaborado um *ranking* das correlações, por meio dele foi possível identificar quais critérios atacar em conjunto para potencializar a uma melhoria na ontologia.

Feito essa análise é possível afirmar que a OntoExper-SPL é altamente adaptável a novas extensões, tornando ela coesa e de fácil extensão a novos axiomas. Porém constata-se que há uma falha com relação à eficiência computacional, ou seja, ela não possui um bom desempenho em sua execução (execução de inferências).

4.7 Ameaças à Validade

Esta seção discute a validade identificadas durante o estudo. Para apresentar as ameaças à validade, foram categorizá-las em interna, externa, de constructo e de conclusão, de acordo com Wohlin et al. (2012b).

4.7.1 Ameaças à Validade Interna

A seguir estão listadas as ameaças à validade interna:

- **Diferenças entre os especialistas:** devido do tamanho da amostra, as variações entre as habilidades dos especialistas foram poucas, assim, os especialistas não foram divididos em grupos. Dessa forma, as tarefas designadas foram idênticas;
- **Acurácia das respostas dos participantes:** uma vez que as informações referentes da OntoExper-SPL são apresentadas juntamente com seus metadados e considerando que os participantes são especialistas em ES, LPS e Ontologia, considera-se que as respostas fornecidas são válidas;
- **Efeito de Fadiga:** os arquivos da OntoExper-SPL (modelagem e povoados com indivíduos) são de difícil leitura pelos seres humanos, logo se faz necessária a utilização de ferramentas para exibição da OntoExper-SPL, bem como os textos auxiliares, para uma maior compreensão dos critérios de avaliação, e das imagens anexadas ao questionário. Os efeitos de fadiga são uma das principais ameaças, por isso, para tentar minimizar esse problema, a documentação foi enviada aos especialistas por e-mail com um prazo de 40 dias para enviarem as respostas, assim, eles poderiam responder conforme a sua disponibilidade;
- **Influência de outros especialistas na avaliação:** a influência entre os especialistas foi mitigada, pois eles não sabiam quem foram convidados para tal estudo; e
- **Análise dos dados:** em relação ao critério de avaliação **Eficiência Computacional** existe uma ameaça pois os participantes relataram que não foi possível executar a tarefa, para tentar minimizar esse problema, a escala Likert atribui o valor N, considerando com baixa significância no resultado. A análise dos dados dos especialistas foi realizada somente pelo autor desta dissertação, sem a realização de uma revisão. Dessa forma, tal análise deve ser revisada em trabalhos futuros.

4.7.2 Ameaças à Validade Externa

A obtenção de especialistas qualificados nas área de ES, LPS e Ontologia foi uma das dificuldades encontradas neste estudo e muitos especialistas convidados não puderem participar do estudo por conta dos seus compromissos. Apesar de poucos especialistas participarem do estudo, a qualidade do perfil deles é a variável mais importante nessa avaliação.

4.7.3 Ameaças à Validade de *Constructo*

A instrumentação foi avaliada e adequada, conforme projeto piloto realizado. Quanto ao nível de conhecimento dos especialistas em ES, LPS e Ontologia demonstraram satisfatórios para avaliarem as diretrizes propostas. Os especialistas que não tinha familiaridade com Ontologia, pode encontrar no material de apoio um respaldo para execução das tarefas.

4.7.4 Ameaças à Validade de Conclusão

A principal ameaça para este estudo está relacionada ao número de especialistas (17). Este número é pequeno, mas o conhecimento dos especialistas que colaboraram com este estudo foi significativo. Além disso, pela característica própria do estudo por prezar a qualidade dos especialistas, muitas vezes o número baixo de especialistas pode ser considerado. Assim, não é possível generalizar as conclusões.

4.8 Empacotamento e Compartilhamento

Os documentos disponibilizados aos participantes e resultados deste estudo estão disponíveis em: <https://doi.org/...>, visando disseminação dos dados.

4.9 Propostas de Melhorias com Base na Discussão dos Resultados

Esta Seção discute as melhorias apresentada pelos especialistas que participaram do estudo empírico, bem como armadilhas que a ferramenta OOPS!.

4.9.1 Melhorias Apontada pelos Especialistas

Durante a execução do estudo os especialistas puderam contribuir com um campo aberto a comentários gerais, neste campos eles comentaram os pontos positivos e negativos encontrado durante avaliação dos critérios de qualidade:

- Comentário do E4:

Acredito que a Ontologia esteja bastante genérica e adequada para o contexto da Engenharia de Software. Conseguí navegar e me entender me-

lhor os axiomas usando as ferramentas propostas (Protégé e WebVOWL), mas tive problemas ao tentar executar as queries SPARQL online (usando o <http://atted.jp/sparql> por exemplo).

- Comentário do E5:

As estruturas e relações foram bem estruturadas na ontologia. Os tópicos básicos de experimentação em engenharia de software estão bem representados e associados a literatura básica de experimentação quantitativa.

Quanto aos termos, é necessária uma revisão dos termos utilizados na Ontologia, por exemplo, "Keyworld" para "Keywords".

Na modelagem, a "SPL artifacts" está sozinha, por que? Será que poderia estar associada ao "SPL domain"?

Senti falta de "Results" que poderia estar associada de algum modelo com "Discussion". Acredito que isso poderia ser uma recomendação de melhoria para a ontologia.

Pensando análise qualitativa dos dados, acho que nenhuma modelagem ou representação foi criada em "Analyze" na Ontologia?

Hoje existe um grande interesse na comunidade científica em análises qualitativas, uma sugestão seria estender a OntoExper-SPL para tal contexto de análise. Entretanto, eu entendo que a ontologia foi proposta para experimentos quantitativos, mas isso poderia ser uma extensão ou trabalho futuro bem interessante para a comunidade científica de LPS ou até mesmo para a área de Ontologia em geral.

- Comentário do E8:

O entendimento da ontologia é fácil para aqueles que conhecem o domínio de experimentação em engenharia de software e a subárea de linha de produto de software.

- Comentário do E10:

Gostaria apenas de comentar um detalhe. O id da ontologia tem um typo: "experiemnt".

- Comentário do E11:

Achei confusa a notação para expansão da documentação. Senti falta de cobrir os tipos de estudo (primários, secundários e terciários) Assim como, notei a ausência de "Ameaças a Validade". Os pesquisadores se baseiam somente no Wohlin, sugiro que olhem outras propostas e referências para complementar os domínios e aplicações.

- Comentário do E13:

O questionário está muito denso, e parece ser necessário ter usado a ontologia para responder de uma maneira apropriada. Ficou muito pesado.

- Comentário do E15:

Do meu conhecimento em experimentação e em LPS eu considero que a ontologia proposta está coerente e satisfatoriamente modelada. Senti falta da modelagem dos "subjects" envolvidos no experimento. Não sei esse conceito foi modelado com outro nome e eu não identifiquei, se não foi incluído intencionalmente ou se foi esquecido.

- Comentário do E16:

Talvez seja interessante modificar a classe Package para Replication-Package ou algum termo relacionado com Open Science.

- Comentário do E17:

Me deparei com o problema da definição do Reasoner para executar a query. Se executado no arquivo que não está populado, o resultado da query atende aos 2s esperados, contudo, ao meu ver uma query que limita-se em retornar apenas o nome das classes relacionadas não é válida para a utilização da ontologia definida. Como tal problema pode ser resultado de algo específico na configuração do ambiente de execução (o quê acredito que não seja o caso), assinalei a opção 3.

Executando a query: Experiment and ExperimentSPL and pagesNumber value 8 não obtive resultado, mesmo tendo navegado e visualizado que há respostas que satisfaçam tal pesquisa. Adicionalmente, depois de selecionar para inicializar o reasoner e, após alguns minutos ser apresentada uma lista de inconsistências (para o arquivo smart-ontology-populate-validated.owl), o Protege para de responder.

Tentei executar ainda a query presente no artigo da ontologia, também sem sucesso.

Ainda sobre o processo de avaliação, a definição de queries para serem executadas deveriam estar mais explícitas.

No geral foi possível notar que a OntoExper-SPL precisa ser melhorada quanto a sua eficiência computacional. Os pontos de melhoria são, corrigir os termos *Keyword* para *Keywords*, associar *SPL artifacts* a *SPL domain*, incluir os axiomas *Results*, *Analyze*, “Ameaças a Validade”, *Subject*se modificar o axioma *Package* para *Replication-Package*.

4.9.2 Melhorias com Base em Armadilhas Identificadas na Ferramenta do OOPS!

Na Tabela - 3.1 foi relatado as armadilhas que a ferramenta encontrou na OntoExper-SPL, a seguir estão descritos as melhorias que devem ser realizada na ontologia para melhorar estes pontos:

- No caso **P08**: deve-se criar anotações para os axiomas;
- No caso **P10**: deve-se criar as disjunções dos axiomas;
- No caso **P13**: deve-se criar relacionamentos inverso para os axiomas: *typeSelectionOfParticipants*, *typeExperimentSPL*, *typeExperiment*, *typeDesignExperiment*, *typeContextSelection*, *typeContextExperiment*, *experiment*, *documentation*;
- No caso **P19**: deve-se remover os domínios e alcances desnecessários para os axiomas: *documentation*, *experiment*, *typeContextExperiment*, *typeDesignExperiment*, *typeExperiment*, *typeSelectionOfParticipants*;
- No caso **P41**: deve-se atribuir uma licença a OntoExper-SPL.

4.10 Considerações Finais

Este capítulo apresentou a avaliação quantitativa sobre oito critérios de qualidade aplicados a OntoExper-SPL com especialistas em ES, LPS e Ontologia. Os resultados obtidos permitiram identificar a qualidade da OntoExper-SPL. Assim, os resultados dos dados analisados forneceu evidências que as OntoExper-SPL possui um grau de qualidade relevante. Permitindo assim a evolução de novos estudos sobre a OntoExper-SPL, expandindo-a e melhorando seus pontos de baixa qualidade.

Foram sugeridas melhorias à OntoExper-SPL, tanto pelos especialistas como pela ferramenta OOPS!, que devem ser aplicadas em trabalhos futuros. Por fim, comprehende-se que novos estudos devem ser realizados com outros especialistas para expandir o corpo de conhecimento.

Protótipo de um Sistema de Recomendação para OntoExper-SPL

5.1 Considerações Iniciais

Este capítulo apresenta um protótipo de Sistema de Recomendação (SR) para a realização de inferências no modelo de ontologia proposto neste trabalho de mestrado, a OntoExper-SPL. O protótipo serve como uma forma de demonstrar a importância do modelo ontológico e também como uma forma de avaliar a viabilidade de SR para experimento de LPS.

5.2 Sistema de Recomendação

Os sistemas de recomendação são aplicativos de software que visam dar suporte para o usuário na tomada de decisões ao interagir com grandes espaços de informação. Esses softwares recomendam itens de interesse para os usuários com base em preferências que tenham sido expressas explicitamente ou implicitamente (Ricci et al., 2011). Segundo Mahmood e Ricci (2009) os sistemas de recomendação são técnicas ou ferramentas de software, que podem reduzir a sobrecarga de informações para os usuários, sugerindo itens, conteúdos ou serviços, entre outros.

Os sistemas de recomendação surgiram nos trabalhos extensivos das ciências cognitivas, teoria de aproximação, recuperação da informação e teoria de previsões e também possuem influências das ciências de administração e marketing (Allen e Fildes, 2001;

Murthi e Sarkar, 2003). O primeiro sistema de recomendação proposto que se tem conhecimento, foi o *Tapestry*. Nesse sistema criou-se o modelo mais usado em sistemas de recomendação, em que a recomendação de conteúdo é auxiliada pela colaboração de um grupo de pessoas, batizado como “filtragem colaborativa”. Neste trabalho, iniciou-se o desafio de congregar corretamente os dados recomendados com os usuários que o recebem, analisando o real relacionamento de interesse (Kwong et al., 1992; Resnick e Varian, 1997).

Uma definição formal para sistema de recomendação é:

Definição 1 Seja C o conjunto de todos os usuários de um determinado sistema, e seja S' o conjunto de todos os possíveis itens que podem ser recomendados como livros, filmes, restaurantes etc. Seja u a função utilidade que mede o quanto útil é um determinado item s para um determinado usuário c , $u:C \times S \rightarrow R$, onde R é um conjunto totalmente ordenado segundo a função utilidade. Então, para cada usuário $c \in C$, procura-se um item $s' \in S$ que maximiza a utilidade do usuário. Isto pode ser expressado pela equação 5.1:

$$\forall c \in C, s'_c = \operatorname{argmax}_{s \in S} u(c, s) \quad (5.1)$$

Em um sistema de recomendação a utilidade de um item é geralmente representada por uma avaliação que indica o quanto um determinado usuário gosta de um item. No entanto, conforme descrito na definição acima, a função de utilidade pode ser uma função arbitrária.

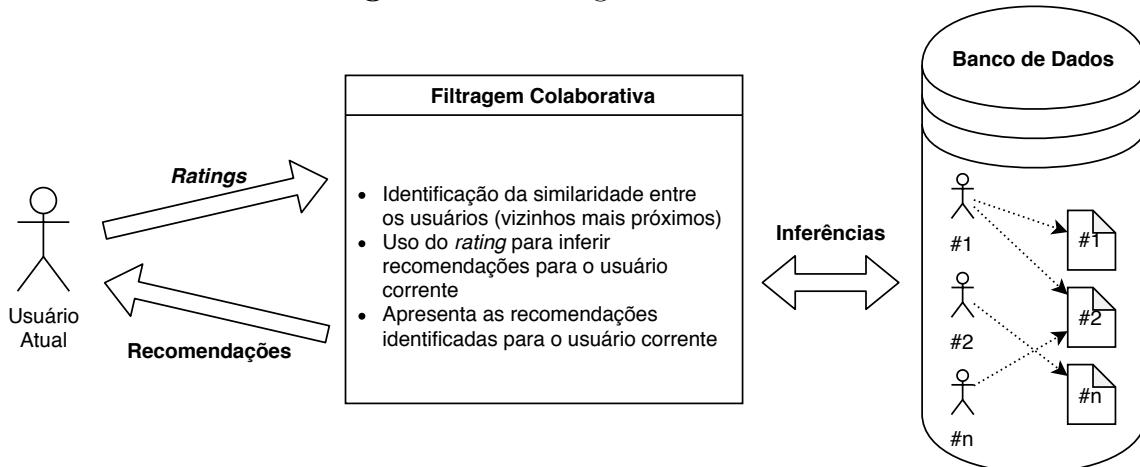
Cada elemento dos usuários C pode ser definido por um perfil que inclui as características do usuário, por exemplo, a sua idade, sexo, etc. No caso mais simples, o perfil pode conter um único elemento como um identificador único (ID). Da mesma forma, cada item de S pode ser definido por um conjunto de características. Por exemplo, na recomendação de filmes, na qual S é a coleção de filmes, cada filme pode ser representado não apenas pelo seu ID, mas também pelo seu título, gênero, diretor, ano de lançamento, etc.

Existem cinco abordagens mais usadas em sistemas de recomendação, Três consideradas como tradicionais: Filtragem Colaborativa (*Collaborative Filtering*), Filtragem Baseada em Conteúdo (*Content-based Filtering*) e Recomendação Baseada no Conhecimento (*Knowledge-Based Recommendation*). Duas consideradas mais atuais: Sistemas de Recomendação Híbridos (*Hybrid Recommender Systems*) e Sistemas de Recomendação usando Informações de Contexto (*Context-aware Recommender Systems*).

5.2.1 Collaborative Filtering

A Filtragem Colaborativa baseia-se na idéia de “boca-a-boca” em que a informação passada de pessoa a pessoa desempenha um papel importante ao tomar uma decisão. Abstraindo, as pessoas são substituídas pelos, chamados, vizinhos mais próximos (NN) que são usuários com um padrão de preferência ou comportamento semelhante ao usuário atual (Robillard et al., 2010). A filtragem Colaborativa depende de dois tipos diferentes de dados: (1) um conjunto de usuários e (2) um conjunto de itens. A relação entre usuários e itens é expressada principalmente em termos de *ratings* fornecidos pelos usuários e explorados em futuras sessões de recomendação para prever a classificação de um usuário (Robillard et al., 2010). A Figura - 5.1 apresenta o comportamento desse modelo.

Figura 5.1: Filtragem Colaborativa



Fonte: Traduzido e Adaptado de Robillard et al. (2010)

Dado que este trabalho trata o sistema de recomendação como uma forma de realização de inferência sobre a OntoExper-SPL, utilizou-se a abordagem Filtragem Colaborativa.

5.2.2 Sistemas de Recomendação em Engenharia de Software

Em Engenharia de Software (*Recommendation System in Software Engineering - RSSEs*), sistemas de recomendação desempenham importantes funções a fim de ajudar a equipe de software a lidar com sobrecarga de informações, filtrando e fornecendo informações úteis. São ferramentas de software introduzidas especificamente para ajudar equipes de desenvolvimento de software e partes interessadas a lidar com a busca de informações em um determinado contexto em ES (Robillard et al., 2010).

Robillard et al. (2010) comentam que, em um ambiente de desenvolvimento de software aplicando ES existe um ampla gama de informações sobre o projeto em desenvolvimento, e este espaço de informações pode ser categorizados por:

- Código fonte do projeto;
- História do projeto;
- Arquivos de comunicação;
- Dependências de API em outras fontes;
- Ambiente de desenvolvimento;
- Logs de interação entre os usuários;
- Logs de execução e;
- A web.

Um RSSE pode trazer simultaneamente dois aspectos distintos: (i) novidade e surpresa, porque os RSSEs ajudam a descobrir novas informações e (ii) familiaridade e reforço, pois as RSSEs suportam a confirmação do conhecimento existente. Referenciar uma tarefa e um contexto específico distingue RSSEs de ferramentas de pesquisa genéricas, por exemplo, uma ferramentas de RSSR para ajudar os desenvolvedores a encontrar exemplos de código fonte (Robillard et al., 2010).

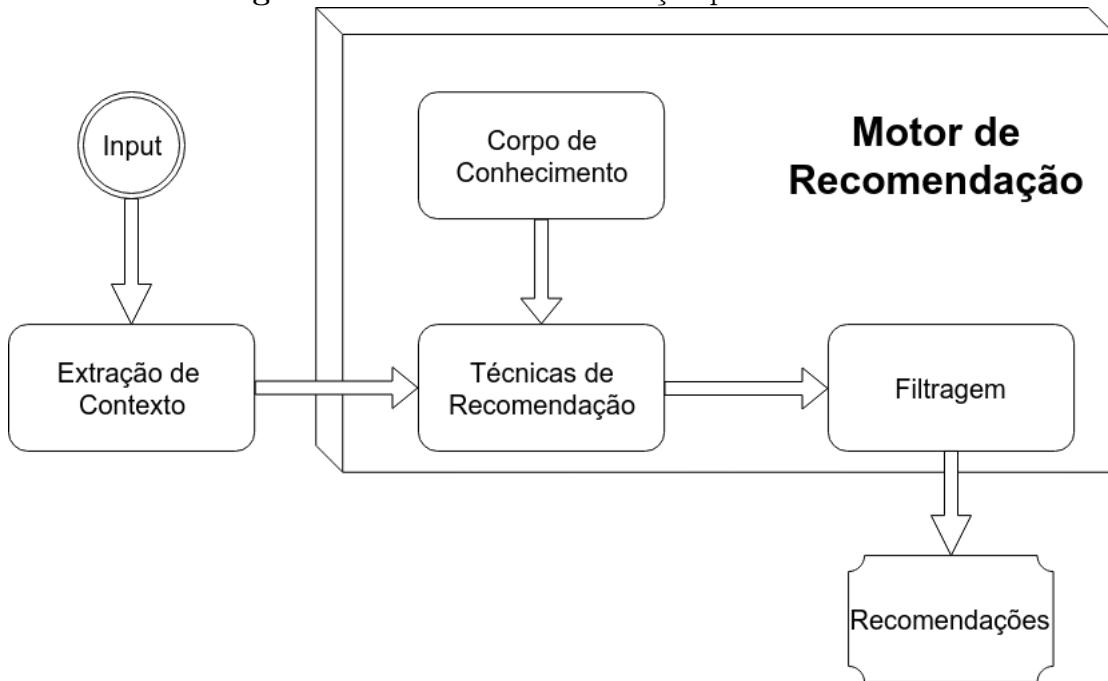
Um RSSE comprehende três componentes principais, (i) um mecanismo para coletar dados, (ii) um mecanismo de recomendação para analisar dados e gerar recomendações e (iii) uma interface de usuário para fornecer recomendações (Rahman et al., 2014).

A Figura - 5.2 apresenta de forma geral como é construído um RSSE, partindo da entradas dos dados pelo *input*, passando pela extração de contexto, seguindo para aplicação de alguma técnica de recomendação, na qual sofre um inferência do corpo de conhecimento (normalmente específico para cada área de ES), depois segue para um processo de filtragem dos resultados, e como saída a recomendação em si.

Foi encontrada uma revisão sistemática (Maki, 2016), que aborda métodos e modelos de implementação de um RSSE apresentando vários aspectos de SR em ES, principalmente no tipo de corpo de conhecimento aplicado a RSSE. Nessa revisão foi possível identificar algumas áreas da ES que utilizam SR:

- para exploração código fonte;

Figura 5.2: Passos de Construção para um RSSE



Fonte: Traduzido e Adaptado de Maki (2016)

- para reuso de software;
- para refatoração de código fonte (por exemplo, *class* em POO);
- para reuso de componentes de software;
- para exploração de APIs;
- para depuração de código (*debugging*)
- para recomendação de agentes *Agile*
- para descoberta de requisitos;
- para mudança do ciclo de vida;
- para evolução do ciclo de vida e;
- para busca de *bugs*.

Por meio deste estudo, foi possível identificar qual técnica de SR para qual domínio de aplicação na indústria.

Tabela 5.1: Sumário de técnicas de recomendação em cada domínio

Domínios	Técnicas							Número de Referências
	CBF	CF	KBF	Hibrido	IA	Redes Sociais	Info. de Contexto	
Governo	1	5	1	5	4			9
Negócios		1	3	3	4			5
Comercio	3	1	4	1	4	2		8
Livraria	2	2		3	1			6
Escolas	2		11		1			10
Turismo	5	9	9	9	2	2	11	18
Pesquisa	9	16	6	15	3	1	1	27
Grupo de Atividade	9	5	2	5	8			21
Total	31	39	36	41	27	5	12	2
								104

Fonte: Traduzido e Adaptado de Maki (2016)

5.3 Concepção

Após o desenvolvimento da ontologia e seu povoamento, iniciou-se um processo empírico de validação dos dados inseridos, realizando inferências a OntoExper-SPL por meio de consultas, como apresentado na Seção 3.4. No decorrer deste estudo foi utilizada a biblioteca OWLReady2 para criação das consultas mais elaboradas, algumas destas serviram de bases para implementar o SR.

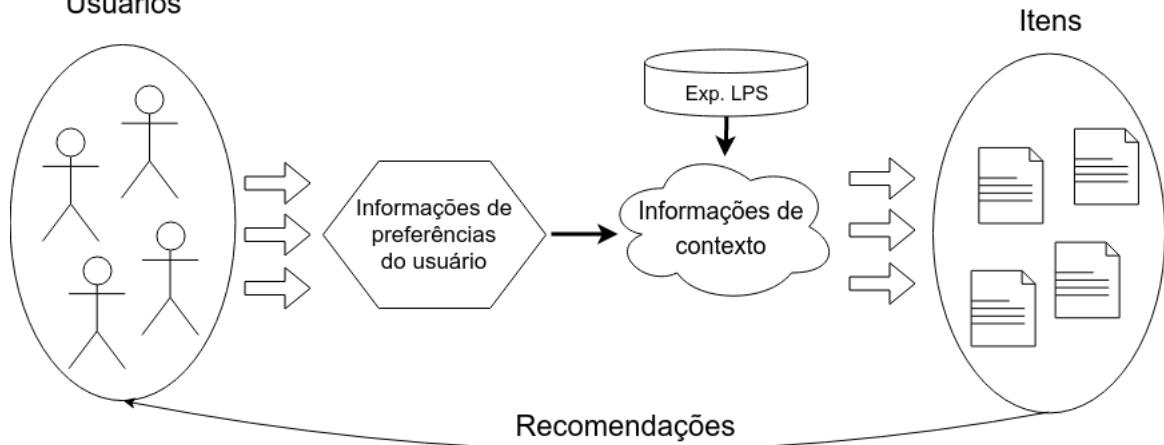
dadas essas considerações iniciais sobre consultas preliminares na ontologia proposta, foi possível esboçar um SR para realizar, como prova de conceito, inferência na OntoExper-SPL. Dessa forma foi elaborado um projeto de SR que possa conectar os usuários aos itens (experimentos de LPS).

A Figura - 5.3 apresenta o conceito geral da metodologia deste projeto. Iniciando pela entrada dos usuários, depois extraem-se as preferências dele (por meio de ratings), e em seguida é realizada a extração de informações de contexto utilizando a base de metadados em LPS (OntoExper-SPL), para então fazer inferência nos itens. Por meio dessa inferência é possível obter as recomendações (itens ranqueadas para cada usuário).

Inicialmente, a entrada de dados dos usuários está definida da seguinte forma:

- Artefatos de LPS:
- Tipo do experimento LPS

Figura 5.3: Modelagem geral do SR como prova de conceito para OntoExer-LPS



Fonte: o Autor

Posteriormente, definiu-se algumas tecnologias para o desenvolvimento do SR, seguindo as boas práticas de desenvolvimento de sistemas. Por meio dessas tecnologias foi possível construir o SR descrito na próxima seção.

5.4 Projeto

Para o desenvolvimento do SR foi necessário encontrar tecnologias que atendessem os requisitos, tanto de SR como de ontologias. A linguagem de programação Python foi escolhida novamente por atender estes dois requisitos. Quanto à ontologia, Python já foi utilizada para manipulação da ontologia por meio da biblioteca OWLReady2. Quanto ao SR por possuir facilidade de implementação de algoritmos complexos e manipulação de dados com Pandas. Um viés para esta implementação foi o conhecimento do pesquisador em desenvolvimento web, isso motivou o uso do Django Framework¹ para desenvolvimento web com Python. Dessa forma, toda *stack* de desenvolvimento do SR foi baseada na linguagem Python.

O desenvolvimento deste projeto foi dividido em quatro fases:

- **Fase 1:** criação do projeto de aplicação web usando Django Framework e implementação de um protótipo da tela de interação com usuário;
- **Fase 2:** carregamento da OntoExer-SPL e implementação dos mecanismos de consultas;

¹<https://djangoproject.com>

- **Fase 3:** *ratings* dos usuários e armazenamento da avaliação no SR; e
- **Fase 4:** implementação do modelo de recomendação e implementação do conceito de filtragem colaborativa.

5.4.1 Fase 1: O Projeto de SR Usando Django Framework

Django Framework é um framework *open source* para desenvolvimento de aplicações web para linguagem Python. Ele implementa o modelo para desenvolvimento em camadas chamado MTV (*Model, Template, View*). Este conceito é semelhante a arquitetura MVC (*Model, View, Controller*) (Krasner et al., 1988), porém com outra nomenclatura. A seguir é descrita qual a responsabilidade de cada camada no Django:

- **A camada do modelo:** fornece uma camada de abstração para estruturar e manipular os dados da aplicação da web;
- **A camada de visualização:** tem o conceito de "visualizações" para encapsular a lógica responsável pelo processamento da solicitação de um usuário e pelo retorno da resposta; e
- **A camada de template:** fornece uma sintaxe amigável para o desenvolvedor renderizar as informações a serem apresentadas ao usuário.

Além de implementar este padrão de modelo de desenvolvimento, o Django oferece outras características, como um processo intuitivo e claro de configuração por meio de um arquivo *settings.py*, a possibilidade de encapsulamento de processos com criação de pequenas aplicações dentro do projeto, possibilidade de internacionalização, implementação de autenticação e segurança, configuração de *timezone*, além de uma implementação pronta para administração da aplicação, chamado Django Admin.

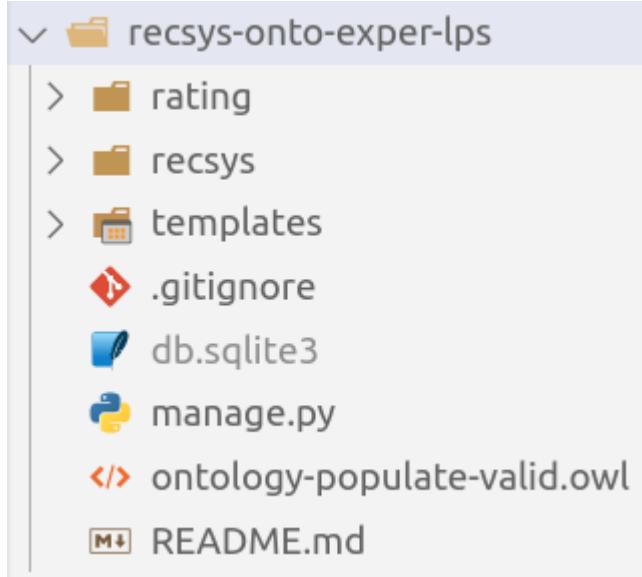
Uma das partes do Django admin é o site, ela oferece a geração de páginas automáticas com base nos metadados dos modelos de negócio criado pelo desenvolvedor.

Outro recurso que o Django trás é parte de autenticação de segurança, por meio deste recurso é possível criar e gerenciar contas de usuários, grupo de usuários, permissões de acesso e os *cookies* de sessão do usuário.

Dessa forma, com o Django Framework, foi possível implementar um SR robusto, garantindo as principais funcionalidades de uma aplicação web.

A aplicação construída conta com a seguinte árvore de diretórios apresentada na Figura - 5.4

Figura 5.4: Árvore de diretórios da aplicação SR com Django Framework



Fonte: o Autor

O diretório `recsys-onto-exper-lps` é o diretório raiz do projeto, todo código fonte da aplicação se encontra nele.

O diretório `ratings` é uma aplicação dentro do projeto, nela contém a modelagem de dados para o armazenamento das avaliações dos usuários

O diretório `recsys` é o diretório de configuração principal do projeto, nela contém o arquivo `settings.py` (configuração global do projeto) e `views.py` que é o principal arquivo deste projeto. Nele contém toda regra e controle das funcionalidades implementadas neste projeto - nas próximas subseções será detalhada estas funcionalidades. Neste diretório tbm contém o arquivo `collaborativeFilter.py` que implementa o modelo de recomendação.

O diretório `templates` é o diretório responsável por encapsular todas as partes da única tela que o projeto implementa. São cinco arquivos HTML que são apresentados em uma única tela para o usuário no navegador.

O arquivo `.gitignore` é responsável por configurar quais arquivos e diretórios não serão versionados pelo sistema de controle de versionamento (Git).

O arquivo `db.sqlite3` é próprio banco de dados. O banco de dados do projeto será detalhado na Seção 5.5.

O arquivo `manage.py` é responsável por prover a execução da aplicação Django Framework.

O arquivo `ontology-populate-valid.owl` é a própria OntoExper-SPL povoada com seus indivíduos, na qual o projeto realiza inferências.

O arquivo *README.md* é um arquivo de marcação que descreve o projeto.

5.4.2 Fase 2: Carregamento da OntoExper-SPL no SR

Como foi apresentado na seção anterior, o arquivo *ontology-populate-valid.owl* está localizado na raiz do projeto, para o SR manusear este arquivo, foi preciso criar uma configuração no arquivo *settings.py* apresentado na Listagem 5.1

Listing 5.1: Configuração do arquivo da OntoExper-SPL

```
ONTOLOGY_FILE = 'ontology-populate-valid.owl'
```

Dada essa configuração foi possível no arquivo *views.py* realizar o carregamento da ontologia utilizando a biblioteca OWLReady2, dessa forma, transformando a OntoExper-SPL em um objeto manipulável pela linguagem de programação. Este passo é apresentado na Listagem 5.2

Listing 5.2: Carregando a OntoExper-SPL

```
import os
from owlready2 import *

def home(request):
    onto = get_ontology(
        os.path.join(settings.BASE_DIR,
                    settings.ONTOLOGY_FILE)).load()
```

A função *get_ontology()* foi importada da biblioteca OWLReady2, ela retorna um objeto que chamamos de **onto**, com ele é possível executar outras operações que a biblioteca oferece.

A partir desse momento foi possível realizar as inferências na OntoExper-SPL. Foi criado um método *_getTypesFrom()* que recebe como parâmetro uma classe da ontologia e retorna seus indivíduos, onde dessa forma, foi possível buscar as informações de tipagem encontrados na ontologia. Os tipos mapeados em classes que consultamos neste momento são:

- onto.TypeExperiment;
- onto.TypeContextExperiment;

- onto.TypeContextSelection;
- onto.TypeDesignExperiment;
- onto.TypeSelectionParticipantsObjects; e
- onto.TypeExperimentSPL.

Figura 5.5: Tela do SR

The screenshot shows a web browser window for 'OntoExper-RecSys' at '127.0.0.1:8000'. The title bar says 'OntoExper-RecSys'. The main content area has a header 'Experiments in SPL'. Below it is a section titled 'Recomendações' (Recommendations) showing four experiment cards:

- #164: Toward recovering component-based software product line architecture from object-oriented product variants
- #55: Comparison of Exact and Approximate Multi-objective Optimization for Software Product Lines
- #129: Product Line Variability Modeling Based on Model Difference and Merge
- #117: On Extracting Feature Models from Product Descriptions

Below these cards is a 'Consulta' (Query) section with two dropdown menus:

- Artefatos de LPS: Selecionar
- Tipo do experimento LPS: Selecionar

Fonte: o Autor

A Figura - 5.5 apresenta a tela construída para o SR. Nessa tela existem dois campos de consultas, (i) Artefatos de LPS e (ii) Tipo de experimento em LPS. Quando o usuário seleciona qualquer um dos dois campos, o SR invoca o método *filter_experiment()* passando os dados selecionados como parâmetro. Nesse método é realizado a inferência na ontologia, segundo os valores que o usuário selecionou na tela. A listagem 5.3 apresenta o código da consulta realizada.

Listing 5.3: Inferência na OntoExper-SPL

```
result = onto.search(is_a=onto.Abstract,
                     documentation=onto.search(is_a=onto.Documentation,
                     experiment=onto.search(
```

```
    wasTheSPLSourceUsedInformed="*%s*" % sourceSPL ,  
    typeExperimentSPL=instance_typeExperimentSPL ,  
    _case_sensitive=False  
)  
)
```

5.4.3 Fase 3: *Ratings* dos Usuários no SR

Existem duas formas de gerar avaliação aos itens em SR, o *rating* explícito e o *rating* implícito:

- No *rating* explícito o usuário dá uma nota a um item manualmente;
 - No *rating* implícito a nota de um item é inferida a partir do comportamento do usuário.

Neste projeto de SR foi implementado o modelo de avaliação explícito, onde cada usuário dá uma nota a um experimento em LPS. Após o usuário realizar um consulta ele pode navegar nos resultados e ao selecionar um experimento na lateral direita é carregado o resumo do mesmo. Nessa lateral foi implementada a opção do usuário dar sua nota. Aqui foi utilizado o modelo estrela, ou seja, o usuário tem a opção de dar de uma a cinco estrelas por experimento. A Figura - 5.6 apresenta a tela que com as opções de *rating* explícito.

Quando o usuário dá a sua nota, o SR invoca o método *rating_experiment()* que recebe como parâmetro o experimento e a nota dada pelo usuário e realiza a persistência desses dados em um banco de dados relacional, na Seção 5.5 está detalhada a modelagem desses dados.

5.4.4 Fase 4: Modelagem de Recomendação, Um Algoritmo para Filtragem Colaborativa

Neste projeto foi utilizada como abordagem de recomendação a filtragem colaborativa. Esse modelo foi implementado utilizando o cálculo da distância Euclidiana a fim de encontrar a similaridade entre os usuários do SR, em seguida, calcular as possíveis notas dos itens que o usuário em questão ainda não avaliou e ranquea-las. Dessa forma sendo possível recomendar, sugerindo determinada prioridade, os itens que o usuário ainda não se relacionou no SR e provavelmente gostaria de interagir.

Figura 5.6: Rating Explícito no SR

Consulta

Use os campos para consultar experimentos em LPS

Artefatos de LPS

Tipo do experimento LPS

7 A hybrid approach to suggest software product line portfolios	Resumo:
94 Feature location in a collection of product variants: Combining information retrieval and hierarchical clustering	Software Product Line Engineering (SPLE) is a systematic reuse approach to develop a short time-to-market and quality products, called Software Product Line (SPL). Usually, the SPL is not developed from scratch but it is developed by reusing features (resp. their source code elements) of existing similar systems developed by ad-hoc reuse techniques. The feature implementations may be changed for adapting SPLE context. The change may impact other features that are not interested in the change, as a feature's implementation spans multiple code elements and shares code elements with other features. Therefore, feature-level Change Impact Analysis (CIA) is needed to predict affected features for change management purpose.
97 Feature-level change impact analysis using formal concept analysis	★★★★☆
98 Feature-to-Code Traceability in Legacy Software Variants	
164 Toward recovering component-based software product line architecture from object-oriented product variants	

Fonte: o Autor

Para tal resultado o algoritmo está dividido em três partes: (i) criação do *dataset* para o algoritmo, (ii) cálculo da similaridade - utilizando o cálculo da distância Euclidiana e (iii) cálculo da nota para os itens ainda não avaliados:

- **Criação do *dataset* para o algoritmo:** esta parte consulta o banco de dados e agrupa as notas dos experimentos por usuário, dessa forma gerando uma lista de usuário e as notas que ele deu para cada experimento. Por exemplo a lista: ['usuário-1': 69: 5.0, 1: 4.0, 5: 1.0, 129: 5.0, 170: 2.0, 97: 4.0, 'usuário-2': 102: 3.0, 117: 4.0, 125: 3.0, 13: 2.0, 9: 3.0, 129: 4.0, 170: 4.0], pode-se observar que o usuário-1 avaliou seis experimentos e o usuário-2 sete sendo eles #102, #117, #125, #13, #9, #129 e o #170 (esses números representam o identificador de cada experimento no SR), cada um com sua respectiva nota. Essa modelagem de dados é necessária para os próximos passos do algoritmo.
- **Cálculo da similaridade:** a fundamentação da distância Euclidiana é medir a distância entre dois pontos ou mais, como descrito na Equação 5.2.

$$DE(x, y) = \sqrt{\sum_i^p (x_i - y_i)^2} \quad (5.2)$$

Usou-se este fundamento para encontrar a distância das notas dadas pelos usuários. Por exemplo, dado um determinado experimento com identificador #170 usuário-1

deu nota 2.0 e o usuário-2 deu nota 4.0, pelo cálculo da distância Euclidiana temos que o usuário-1 é distante do usuário-2 por um valor 2. Esse cálculo fica mais interessante quando aplicamos mais dimensões, cada dimensão no nosso cenários pode ser associada a um experimento, por exemplo, se for incluido no cálculo o experimento #129, tem-se que o usuário-1 deu nota 5.0 e o usuário-2 deu nota 4.0 logo calculo fica $\sqrt{(2 - 4)^2 + (5 - 4)^2}$, retornando 3. O próximo passo é normalizar esse valor de retorno da distância Euclidiana entre 0 e 1, aplicando a Equação 5.3. Dessa forma pode-se afirmar que neste exemplo a similaridade entre o usuário-1 e o usuário-2 é 0.25

$$sim = \frac{1}{(1 + de)} \quad (5.3)$$

Todo algoritmo e cálculos que foram implementados estão no arquivo *collaborative-Filter.py* e está disponível no Apêndice ??.

- **Cálculo da nota para os itens ainda não avaliados:** esta parte do algoritmo, busca prever uma nota para os itens ainda não avaliados pelo usuário, para tal efeito é necessário realizar um cálculo de média ponderada usando o cálculo de similaridade descrito anteriormente, este cálculo está descrito na Equação 5.4. Para melhor entendimento do cálculo, segue o exemplo, dado os experimentos #129 e #170, o usuário-1 deu nota 2.0 para o #129 e nota 3.0 para #170, o usuário-2 deu nota 4.0 para o #129 e nota 5.0 para #170 e o usuário-3 não deu nota para o #129 e nota 4.0 para #170, temos um usuário-4 queremos encontrar qual nota que ele daria para os experimentos #129 e #170, sabendo que o usuário-4 tem uma similaridade 0,4 com usuário-1, 0,25 com usuário-2 e 0,18 com usuário-3. a tabela Tabela - 5.2 apresenta os resultados deste cálculo. Dessa forma podemos dizer que o usuário-4 daria nota 3,82 para #170 e 2,77 #129, portanto o SR recomendaria o experimento que #170 por ter maior nota.

$$NOTA_p() = \frac{\sum_i^{usuários} (sim_i * nota_i)}{\sum_i^{usuáriosTemNota} (sim_i)} \quad (5.4)$$

O modelo de recomendação usando foi a filtragem colaborativa com *rating* explícito gera o problema da partida fria, ou seja, não temos dados o suficientes no SR quando ainda não há avaliações lançadas, isso implica em resultados não satisfatórios de recomendação. Foi implementado um modelo de nota aleatória para resolver este problema. Porém, a

Tabela 5.2: Cálculo para previsão de nota

	Sim	Nota #129	Sim X #129	Nota #170	Sim X #170
usuário-1	0,40	2	0,80	3	1,20
usuário-2	0,25	4	1,00	5	1,25
usuário-3	0,18		0,00	4	0,72
Total			1,80		3,17
Soma Sim			0,65		0,83
Total / Soma			2,77		3,82
usuário-4		2,77		3,82	

Fonte: o Autor

melhor solução seria implementar a filtragem baseada em conteúdo, pois esse modelo não se baseia nos *ratings* dados pelos usuários.

5.5 Banco de Dados

Para o banco de dado do SR foi utilizado o SGBD SQLite², ele foi utilizado por ser o banco de dados padrão para Django Framework, dessa forma não foi preciso realizar nenhuma outra configuração. O SQLite é um dos bancos de dados mais simples, todos os registros de metadados são armazenados em um único arquivo.

As entidades existentes no banco de dados, são:

- auth_group;
- auth_group_permissions;
- auth_permission;
- auth_user;
- auth_user_groups;
- auth_user_user_permissions;
- django_admin_log;
- django_content_type;
- django_migrations;
- django_session;

²<https://www.sqlite.org/index.html>

- rating_rating; e
- sqlite_sequence.

As entidades com prefixo *auth* e *django* são geradas e mantidas pelo próprio Django Framework. A entidade *rating_rating* foi gerada pela aplicação *rating* dentro do projeto, a Tabela - 5.3 descreve sua estrutura. Sua função é armazenar as avaliações dos usuário, a coluna *user* armazena o login do usuário, a coluna *rating* armazena a nota dada, a coluna *id_experiment* armazena o id do experimento e a coluna *title_experiment* armazena o título do experimento.

Tabela 5.3: Entidade *rating_rating* do Banco de Dados

Propriedade	Tipo
<i>id</i>	integer
<i>user</i>	varchar(50)
<i>rating</i>	real
<i>id_experiment</i>	integer
<i>title_experiment</i>	varchar(255)

Fonte: o Autor

5.6 Ambiente de Desenvolvimento

Para o desenvolvimento do SR foram necessárias algumas ferramentas de apoio, além do Django Framework e do SQLite já citado, foram utilizadas outras tecnologias como HTML CSS e JS, com o apoio dos *frameworks* Bootstrap³ e jQuery⁴.

Para ambiente de desenvolvimento de código fonte foi utilizado o editor de texto VSCode⁵ da Microsoft e a ferramenta Jupyter-Notebook e Jupyter-Lab, para visualização do SR o navegador Google Chrome e FireFox e para o banco de dados o DB Browser⁶. Todas essas ferramentas são *open source* e estão disponíveis para *download* até a data de escrita deste trabalho.

Foram escolhidas essas ferramentas, pois o pesquisador já possuía habilidades nelas e por possuir maturidade e reconhecimento na indústria de desenvolvimento de software.

³<https://getbootstrap.com/>

⁴<https://jquery.com/>

⁵<https://code.visualstudio.com/>

⁶<https://sqlitebrowser.org/>

5.7 Empacotamento e Compartilhamento

Os artefatos utilizado para o desenvolvimento do SR estão disponíveis em: <https://doi.org/...>, visando disseminação dos dados.

5.8 Considerações Finais

Este capítulo apresentou um SR provando que é possível realizar inferências no modelo de ontologia proposto por este trabalho de mestrado, a OntoExper-SPL. Por meio das inferências implementadas no SR, foi possível implementar o modelo de recomendação *collaborative filtering* e gerar recomendações para os usuários que interagirem com o SR.

Inicialmente, a termo de contexto, foi apresentados os conceitos básico de sistemas de recomendação e uma breve descrição do modelo *collaborative filtering*. Na sequência foi apresentado como os sistemas de recomendação são tratados na ES e alguns exemplos de aplicação.

Foi apresentado na seção de concepção do SR, a modelagem inicial da aplicação, as principais tecnologias abordadas antes da fase de implementação, o ecossistema Python serviu como base para o desenvolvimen geral do SR, desde a inferências iniciais até a implementação de modelo de recomendação. Já na seção de projeto foi tratado especificamente de como as tecnologias e ferramentas auxiliaram em cada fase do projeto, que foram divididas em quatro fases, (i) criação do projeto de aplicação web usando Django Framework, (ii) carregamento da OntoExper-SPL no SR, (iii) *ratings* dos usuários no SR e (iv) implementação do modelo de recomendação. O Django Framework se destaca pois ele foi a base de implementação da aplicação web, pois ele foi construído para esse propósito, implementando principalmente os conceitos de MVC. Posteriormente foi apresenta o banco de dados e sua principal tabela para SR, a tabela que armazena os *ratings* do usuário. Por fim, foi apresentado o ambiente de desenvolvimento utilizado para elaborar do SR, a principal ferramente para este fim foi o editor de texto VSCode.

No próximo capítulo, será apresentado a conclusão final deste trabalho de mestrado.

Conclusão

A realização de experimentos de LPS traz informações relevantes para fornecer evidências de uma teoria para o mundo real. A capacidade de compreender, estudar, e replicar experimentos tornam esse método mais atrativo para a comunidade. Porém, tem se notado uma carência de documentação formal e estruturada para documentar e armazenar os experimentos, que por sua vez, acabam por dificultar a repetição dos estudos em LPS. Assim, com a organização dos dados e informações sobre experimentos de LPS, acredita ser possível tornar essa tarefa mais fácil e atrativa. Usando das próprias tecnologias adjacentes, como formalização do conhecimento, por meio dos modelos de ontologias, que, posteriormente possibilita a realização de inferências a eles. O resultado acaba sendo a geração de informações organizadas, estruturadas, personalizadas e úteis, principalmente pela oportunidade da criação de aplicações auxiliares neste sentido, como por exemplo sistemas de recomendações, bem como incentivar a cultura de experimentos pela indústria de software.

Nesse contexto foi desenvolvida uma ontologia, a OntoExper-SPL, com o objetivo de organizar e estruturar o conhecimento adquirido sobre experimentos de LPS. Esse corpo de conhecimento foi previamente identificado em um trabalho de dissertação de mestrado do grupo de pesquisa GReater (Furtado, 2018), em que 211 artigos relatam experimentos de LPS. Com esse levantamento foi possível elaborar um modelo de ontologia a fim de representar o conhecimento sobre experimentos de LPS e em seguida inseri-lo no modelo.

Com a composição da OntoExper-SPL foi possível responder a questão de pesquisa desta dissertação: **Como formalizar o conhecimento de experimentação em LPS?**. A OntoExper-SPL foi avaliada por um estudo empírico, o qual foram considerados oito critérios de qualidade desenvolvidos por Vrandecic (2010), por meio de uma escala Likert,

em que participaram especialistas de ES experimental, LPS e Ontologias. Também foi possível realizar uma breve avaliação em relação a algumas armadilhas que o modelo da OntoExper-SPL poderia apresentar usando a ferramenta OOPS!.

Os resultados obtidos do estudo empírico forneceram evidências preliminares que a OntoExper-SPL é viável para formalizar o conhecimento de experimentação em LPS com nível de qualidade satisfatório.

Foi elaborado um protótipo de sistema de recomendação em que é possível realizar inferência e extrair informações relevantes da OntoExper-SPL. O resultado foi um SR capaz de interagir com usuários, podendo realizar consultas e receber recomendações a partir de inferências na OntoExper-SPL.

6.1 Contribuições

A principal contribuição desta pesquisa foi a formalização do conhecimento sobre experimentação em LPS. LPSs são construídas por meio de um domínio de aplicação, similaridades, artefatos e variabilidades, o que distingue um produto do outro dentro da família de produtos. Todas essas características estão presentes na OntoExper-SPL. Entretanto, esse modelo proposto pode ser facilmente estendido para outros domínios de experimentos em ES, pois todas as classes dentro da ontologia que tratam de LPS são sub-classes de representações de experimentos em geral. Outras contribuições podem ser elencadas:

- apresentação dos resultados de um estudo empírico aplicando escala Likert para avaliação de critérios de qualidade a ontologias;
- um protótipo de sistema de recomendação com a finalidade de recomendar experimentos em LPS de maneira mais relevante para o usuário que está interagindo com o SR;
- possibilidade de incentivo à cultura de experimentos em LPS na academia e na indústria; e
- possibilidade de melhoria no ensino de ES Experimental por meio da formalização do conhecimento e do protótipo de SR.

6.2 Limitações

As limitações deste trabalho são discutidas nesta seção.

A OntoExper-SPL considera elementos experimentais gerais e da abordagem de LPS, portanto, não é possível generalizar para outros domínios específicos.

O número de especialistas que participaram do estudo quantitativo para avaliar os critérios de qualidade da OntoExper-SPL. Apesar do conhecimento avançado dos participantes, acredita-se que novas avaliações são necessárias, bem como a melhoria do questionário e replicação de tal estudo com mais de participantes para expandir a base de conhecimento.

Não foi possível realizar uma avaliação para o protótipo de SR.

6.3 Trabalhos Futuros

A seguir são apresentadas algumas sugestões de trabalhos futuros:

Identificou-se que na avaliação empírica existem vários pontos de melhoria na modelagem da OntoExper-SPL. Esses ajustes devem ser realizados para padronizar o modelo com o objetivo de tornar possível a extensão e divulgação da mesma, principalmente em relação à eficiência computacional, dessa forma viabilizando a implementação de novas aplicações de inferência.

Estender a OntoExper-SPL para incluir metadados de qualidade já definidos no trabalho de Furtado (2018).

O desenvolvimento de um sistema de recomendação mais amplo, considerando outros modelos de recomendação que possam explorar mais a OntoExper-SPL, como por exemplo a filtragem baseada em conteúdo. Dessa forma, pode-se realizar explorações mais profundas no modelo proposto.

REFERÊNCIAS

- ALLEN, I. E.; SEAMAN, C. A. Likert scales and data analyses. *Quality progress*, v. 40, n. 7, p. 64–65, 2007.
- ALLEN, P. G.; FILDES, R. Econometric forecasting. In: *Principles of forecasting*, Springer, p. 303–362, 2001.
- ALMEIDA, M. B.; BAX, M. P. Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção. *Ciência da informação*, v. 32, n. 3, p. 7–20, 2003.
- APEL, K.-O. *Analytic philosophy of language and the geisteswissenschaften*. Springer, 2013.
- BASILI, V. R.; ROMBACH, H. D. The TAME project: Towards improvement-oriented software environments. *IEEE Transactions on software engineering*, v. 14, n. 6, p. 758–773, 1988.
- BLONDET, G.; LE DUIGOU, J.; BOUDAOU, N. Ode: an ontology for numerical design of experiments. *Procedia CIRP*, v. 50, p. 496–501, 2016.
- CALVANESE, D.; DE GIACOMO, G.; LEMBO, D.; LENZERINI, M.; ROSATI, R. Dl-lite: Tractable description logics for ontologies. In: *AAAI*, 2005, p. 602–607.
- CAPILLA, R.; BOSCH, J.; KANG, K.-C. *Systems and software variability management: Concepts, tools and experiences*. Springer Publishing Company, Incorporated, 2013.
- CEUSTERS, W.; SMITH, B. Strategies for referent tracking in electronic health records. *Journal of biomedical informatics*, v. 39, n. 3, p. 362–378, 2006.
- CLEMENTS, P.; NORTHRUP, L. *Software product lines: practices and patterns*, v. 3. Addison-Wesley Reading, 2002.

- DA CRUZ, S. M. S.; CAMPOS, M. L. M.; MATTOSO, M. A foundational ontology to support scientific experiments. In: *ONTOBRAS-MOST*, ACM: USA, 2012, p. 144–155.
- CRUZES, D.; MENDONCA, M.; BASILI, V.; SHULL, F.; JINO, M. Extracting information from experimental software engineering papers. In: *XXVI International Conference of the Chilean Society of Computer Science (SCCC'07)*, USA: IEEE, 2007, p. 105–114.
- DENNEY, M. J.; LONG, D. M.; ARMISTEAD, M. G.; ANDERSON, J. L.; CONWAY, B. N. Validating the extract, transform, load process used to populate a large clinical research database. *International journal of medical informatics*, v. 94, p. 271–274, 2016.
- FERNANDA, M. Modelagem conceitual: a construção de uma ontologia sobre avaliação do ciclo de vida (acv) para fomentar a disseminação de seus conceitos. 2007.
- FURTADO, HENRIQUE VIGNANDO, V. F.; OLIVEIRAJR, E. Comparing approaches for quality evaluation of software engineering experiments: An empirical study on software product line experiments. *Journal of Computer Science*, v. 15, n. 10, p. 1396–1429, 2019.
- FURTADO, V. R. *Guidelines for software product line experiment evaluation*. Dissertação de Mestrado, State University of Maringá, Maringá-PR. Brazil, in Portuguese, 2018.
- GARCIA, R. E.; HÖHN, E. N.; BARBOSA, E. F.; MALDONADO, J. C. An ontology for controlled experiments on software engineering. In: *SEKE*, USA: ACM, 2008, p. 685–690.
- GELERNTER, J.; JHA, J. Challenges in ontology evaluation. *Journal of Data and Information Quality (JDIQ)*, v. 7, n. 3, p. 11, 2016.
- GERALDI, R. T. *SMartyCheck: uma Técnica de Inspeção baseada em Checklist para Diagramas de Casos de Uso e de Classes da Abordagem SMarty*. Dissertação de Mestrado, Universidade Estadual de Maringá, Maringá, 241 p., 2015.
- GÓMEZ-PÉREZ, A. Ontology evaluation. In: *Handbook on ontologies*, Springer, p. 251–273, 2004.
- GRUBER, T. What is an ontology. WWW Site <http://www-ksl.stanford.edu/kst/whatis-an-ontology.html> (accessed on 07-09-2004), 1993.

GUIZZARDI, G. *Desenvolvimento para e com reuso: um estudo de caso no domínio de vídeo sob demanda.* 2000. 202 f. Tese de Doutoramento, Dissertação (Mestrado)–Universidade Federal do Espírito Santo, 2000.

HUNTER, J. D. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, v. 9, n. 3, p. 90–95, 2007.

ISOTANI, S.; BITTENCOURT, I. I.; BARBOSA, E. F.; DERMEVAL, D.; PAIVA, R. O. A. Ontology driven software engineering: a review of challenges and opportunities. *IEEE Latin America Transactions*, v. 13, n. 3, p. 863–869, 2015.

JENA, A. semantic web framework for java. 2007.

JONES, E.; OLIPHANT, T.; PETERSON, P.; ET AL. SciPy: Open source scientific tools for Python. [Online; accessed <today>], 2001–.

Disponível em <http://www.scipy.org/>

KIRCH, W., ed. *Pearson's correlation coefficient* Dordrecht: Springer Netherlands, p. 1090–1091, 2008.

Disponível em https://doi.org/10.1007/978-1-4020-5614-7_2569

KITCHENHAM, B.; BUDGEN, D.; BRERETON, P.; TURNER, M.; CHARTERS, S.; LINKMAN, S. Large-scale software engineering questions—expert opinion or empirical evidence? *IET software*, v. 1, n. 5, p. 161–171, 2007.

KITCHENHAM, B. A.; BUDGEN, D.; BRERETON, P. *Evidence-based software engineering and systematic reviews*, v. 4. CRC Press, 2015.

CLUYVER, T.; RAGAN-KELLEY, B.; PÉREZ, F.; GRANGER, B.; BUSSONNIER, M.; FREDERIC, J.; KELLEY, K.; HAMRICK, J.; GROUT, J.; CORLAY, S.; IVANOV, P.; AVILA, D.; ABDALLA, S.; WILLING, C. Jupyter notebooks – a publishing format for reproducible computational workflows. In: LOIZIDES, F.; SCHMIDT, B., eds. *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, IOS Press, 2016, p. 87 – 90.

KRASNER, G. E.; POPE, S. T.; ET AL. A description of the model-view-controller user interface paradigm in the smalltalk-80 system. *Journal of object oriented programming*, v. 1, n. 3, p. 26–49, 1988.

KWONG, K. K.; BELLIVEAU, J. W.; CHESLER, D. A.; GOLDBERG, I. E.; WEISSKOFF, R. M.; PONCELET, B. P.; KENNEDY, D. N.; HOPPEL, B. E.; COHEN, M. S.; TURNER,

- R. Dynamic magnetic resonance imaging of human brain activity during primary sensory stimulation. *Proceedings of the National Academy of Sciences*, v. 89, n. 12, p. 5675–5679, 1992.
- LAMY, J.-B. Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies. *Artificial intelligence in medicine*, v. 80, p. 11–28, 2017.
- VAN DER LINDEN, F.; SCHMID, K.; ROMMES, E. The product line engineering approach. In: *Software Product Lines in Action*, Springer, p. 3–20, 2007.
- LOHMANN, S.; NEGRU, S.; HAAG, F.; ERTL, T. Visualizing ontologies with vowl. *Semantic Web*, v. 7, n. 4, p. 399–419, 2016.
- MAHMOOD, T.; RICCI, F. Improving recommender systems with adaptive conversational strategies. In: *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, ACM, 2009, p. 73–82.
- MAKI, S. *Systematic review of recommendation systems in software engineering*. Tese de Doutoramento, École de technologie supérieure, 2016.
- MANN, H. B.; WHITNEY, D. R. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, p. 50–60, 1947.
- MCKINNEY, W. Data structures for statistical computing in python. In: *Proceedings of the 9th Python in Science Conference*, USA, 2010, p. 51 – 56.
- MENDONCA, F. M. Ontoforinfoscience: metodologia para construção de ontologias pelos cientistas da informação-uma aplicação prática no desenvolvimento da ontologia sobre componentes do sangue humano (hemonto). 2015.
- MORAIS, E. A. M.; AMBRÓSIO, A. P. L. Ontologias: conceitos, usos, tipos, metodologias, ferramentas e linguagens. *Relatório Técnico-RT-INF-001/07, dez*, 2007.
- MURTHI, B.; SARKAR, S. The role of the management sciences in research on personalization. *Management Science*, v. 49, n. 10, p. 1344–1362, 2003.
- MUSEN, M. A. The protégé project: a look back and a look forward. *AI Matters*, v. 1, n. 4, p. 4–12, 2015.

Disponível em <https://doi.org/10.1145/2757001.2757003>

NOY, N. F.; MCGUINNESS, D. L.; ET AL. Ontology development 101: A guide to creating your first ontology. 2001.

OLIVEIRAJR, E.; MALDONADO, J. C.; GIMENES, I. M. D. S. Empirical validation of complexity and extensibility metrics for software product line architectures. *Fourth Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS)*, p. 31–40, 2010.

PÉREZ, F.; GRANGER, B. E. IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, v. 9, n. 3, p. 21–29, 2007.

Disponível em <https://ipython.org>

POHL, K.; BÖCKLE, G.; VAN DER LINDEN, F. J. *Software product line engineering: foundations, principles and techniques*. Springer Science & Business Media, 2005.

POVEDA-VILLALÓN, M.; GÓMEZ-PÉREZ, A.; SUÁREZ-FIGUEROA, M. C. Oops!(ontology pitfall scanner!): An on-line tool for ontology evaluation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, v. 10, n. 2, p. 7–34, 2014.

RAHMAN, M. M.; YEASMIN, S.; ROY, C. K. Towards a context-aware ide-based meta search engine for recommendation about programming errors and exceptions. In: *Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week-IEEE Conference on*, IEEE, 2014, p. 194–203.

RECTOR, A.; DRUMMOND, N.; HORRIDGE, M.; ROGERS, J.; KNUBLAUCH, H.; STEVENS, R.; WANG, H.; WROE, C. Owl pizzas: Practical experience of teaching owl-dl: Common errors & common patterns. In: *International Conference on Knowledge Engineering and Knowledge Management*, Springer, 2004, p. 63–81.

RESNICK, P.; VARIAN, H. R. Recommender systems. *Communications of the ACM*, v. 40, n. 3, p. 56–58, 1997.

RICCI, F.; ROKACH, L.; SHAPIRA, B. Introduction to recommender systems handbook. In: *Recommender systems handbook*, Springer, p. 1–35, 2011.

ROBILLARD, M.; WALKER, R.; ZIMMERMANN, T. Recommendation systems for software engineering. *IEEE software*, v. 27, n. 4, p. 80–86, 2010.

SCATALON, L. P.; GARCIA, R. E.; CORREIA, R. C. M. Packaging controlled experiments using an evolutionary approach based on ontology (s). In: *SEKE*, 2011, p. 408–413.

- SHAPIRO, S. S.; WILK, M. B. An analysis of variance test for normality (complete samples). *Biometrika*, v. 52, n. 3/4, p. 591–611, 1965.
- SOLDATOVA, L. N.; KING, R. D. An ontology of scientific experiments. *Journal of the Royal Society Interface*, v. 3, n. 11, p. 795–803, 2006.
- SOMMERVILLE, I. Software engineering, boston, massachusetts: Pearson education. 2011.
- VRANDECIC, D. Ontology evaluation [phd thesis]. *KIT, Karlsruhe*, 2010.
- WASKOM, M.; ET AL. mwaskom/seaborn: v0.8.1 (september 2017). 2017. Disponível em <https://doi.org/10.5281/zenodo.883859>
- WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. *Experimentation in software engineering*. Springer Science & Business Media, 2012a.
- WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. *Experimentation in Software Engineering*. Springer Science & Business Media, 2012b.
- WOLFF, C.; ÉCOLE, J. Philosophia prima sive ontologia. 1962.

Apêndice A: Ontologias - Tipologias e Metodologias de Construção

A.1 Metodologias de Construção

A.1.1 Metodologia TOVE (Grüninger Fox, 1995)

- **Origem e Propósito Principal:** criada com base no desenvolvimento do projeto Toronto Virtual Enterprise (TOVE), cujo objetivo era o de criar um modelo de senso comum ou conhecimento compartilhado sobre empresas. Esta metodologia serviu de base para o projeto e avaliação de ontologias integradas em domínios corporativos, incluindo propostas de construção de ontologias e extensões de ontologias já existentes;
- **Domínio de aplicação:** Negócios (empresarial);
- **Etapas:**
 1. Elaboração de cenários de motivação, que objetivam identificar problemas no ambiente atual;
 2. Especificação de questões de competência informal, que objetivam especificar em linguagem natural os requisitos que a ontologia deverá ser capaz de atender;
 3. Concepção da terminologia formal, em que, mediante declarações em lógica de primeira ordem, os conceitos e suas propriedades são organizados em uma taxonomia;
 4. Especificação de questões de competência formal, em que problemas são definidos de modo consistente perante os axiomas na ontologia;
 5. Especificação de axiomas formais, que restringem a interpretação dos termos envolvidos nas questões de competência formal;

6. Verificação de teoremas completos, que determinam as condições sobre as quais as soluções das questões são completas.

A.1.2 Metodologia de Uschold e King - ENTERPRISE (USCHOLD e KING, 1995)

- **Origem e Propósito Principal:** Método desenvolvido com base na prática da construção da ontologia de alto nível Enterprise Ontology. Tem como propósito principal descrever o conhecimento sobre domínios corporativos ou de negócios;
- **Domínio de aplicação:** Negócios (empresarial);
- **Etapas:**
 1. Identificação do propósito da ontologia, que objetiva identificar a necessidade de construção, o grau de formalismo (desde o informal com uso de linguagem natural até o rigorosamente formal com uso de declarações lógicas) e as classes de usuários da ontologia, incluindo desenvolvedores, mantenedores e usuários das aplicações;
 2. Construção da ontologia em três etapas:
 - captura ou concepção da conceitualização da ontologia;
 - codificação ou implementação através de uma linguagem de representação de ontologias;
 - integração com ontologias já existentes;
 3. Avaliação da ontologia através dos requisitos especificados;
 4. Documentação acerca das pretensões da ontologia e das primitivas usadas para expressar as definições na ontologia.

A.1.3 Methontology (GÓMEZ-PEREZ, FERNANDEZ-LOPES e VICENTE, 1996)

- **Origem e Propósito Principal:** Metodologia desenvolvida no Laboratório de Inteligência Artificial da Universidade Politécnica de Madri entre 1996 e 1997, possibilita a construção de uma ontologia por reengenharia sobre outra ontologia ou a criação de uma ontologia totalmente nova, utilizando-se do conhecimento do domínio tratado. Tal metodologia pode ser usada, segundo seus autores, em quaisquer domínios do conhecimento;

- **Domínio de aplicação:** Diversos;
- **Etapas:**
 1. **Especificação:** identificar o propósito da ontologia, incluindo os usuários pretendidos, cenários de uso, o grau de formalidade requerido, etc., e o escopo da ontologia, incluindo o conjunto de termos a ser representados.
 2. **Aquisição de Conhecimento:** etapa realizada em paralelo com a etapa (1). Esta etapa é não-prescritiva e, assim, qualquer método pode ser usado, embora seja mais comum as entrevistas com especialistas e análises de textos do domínio tratado.
 3. **Conceitualização:** identificação de termos do domínio, tais como conceitos, instâncias, relações verbais e propriedades, com cada termo tendo uma representação intermediária.
 4. **Integração:** deve-se analisar termos de outras ontologias, tal como o padrão Ontolingua, que possam ser reutilizados na ontologia em construção.
 5. **Implementação:** a ontologia deve ser representada formalmente através de uma linguagem formal, tais como a Ontolingua e a lógica descritiva.
 6. **Avaliação:** são usadas técnicas baseadas nos métodos de validação e verificação dos sistemas de base de conhecimento. Há também um conjunto de diretrizes para avaliar incompletudes, inconsistências e redundâncias.
 7. **Documentação:** recomenda-se a especificação de uma documentação em linguagem natural ao final de cada fase do ciclo de vida do desenvolvimento da ontologia.

A.1.4 Método Kactus (BERNARAS, LARESGOTTI, CORERA, 1996)

- **Origem e Propósito Principal:** Método recursivo derivado do projeto Kactus que permitiu a reutilização de conhecimento em sistemas de complexidade técnica, tal como o domínio de redes elétricas, e a construção de ontologias nesse domínio como suporte a tais sistemas;
- **Domínio de aplicação:** Sistemas de complexidade técnica;
- **Etapas:**

1. Desenvolvimento de uma lista de necessidades ou requisitos que precisam ser atendidos pela aplicação;
2. Identificação de termos relevantes para o domínio da aplicação a partir de tais requisitos, construindo, assim, um modelo preliminar;
3. Refinar e estruturar a ontologia a fim de obter um modelo definitivo;
4. Buscar por ontologias já desenvolvidas por outras aplicações no sentido de sua reutilização.

A.1.5 Método Sensus (SWARTOUT et al., 1996)

- **Origem e Propósito Principal:** Método derivado da ontologia Sensus, a qual foi desenvolvida pelo grupo Information Sciences Institute (ISI) com o propósito de ser usada para fins de processamento de linguagem natural. O método Sensus propõe alguns processos para estabelecer as ligações entre os termos específicos e os termos da ontologia de alto nível, que corresponde à ontologia Sensus. Na prática, o método Sensus foi aplicado no desenvolvimento de uma ontologia no domínio de planejamento de uma operação militar aérea;
- **Domínio de aplicação:** Diversos;
- **Etapas:**
 1. Identificar termos-chave do domínio;
 2. Ligar manualmente os termos-chave à ontologia SENSUS;
 3. Adicionar caminhos até o conceito de hierarquia superiorda Sensus;
 4. Adicionar novos termos para o domínio;
 5. Adicionar subárvores completas.

A.1.6 Método 101 (NOY e GUINNESS, 2001)

- **Origem e Propósito Principal:** Método concebido a partir da experiência no desenvolvimento de uma ontologia de vinhos e alimentos, utilizando o editor de ontologias Protégé;
- **Domínio de aplicação:** Diversos;
- **Etapas:**

1. Determinar escopo da ontologia;
2. Considerar o reuso de termos de outras ontologias;
3. Enumerar termos;
4. Definir classes na ontologia;
5. Organizar as classes em uma taxonomia;
6. Definir propriedades (slots) e descrever seus valores permitidos (facetas), através de restrições;
7. Adicionar valores de slots para as instâncias, isto é, criar instâncias.

A.1.7 Método CYC (REED, LENAT, 2002)

- **Origem e Propósito Principal:** Método usado na construção da ontologia CYC, que considera o conhecimento consensual do mundo e é indicada pelos autores na criação de ontologias para fundamentar sistemas inteligentes;
- **Domínio de aplicação:** Diversos;
- **Etapas:**
 1. Extração manual do conhecimento de senso comum;
 2. Extração auxiliada por computador de senso comum;
 3. Extração gerenciada por computador de senso comum;
 4. Desenvolvimento de representação do conhecimento e ontologia de alto nível (CYC) contendo os conceitos mais abstratos;
 5. Representação do conhecimento de diferentes domínios usando tais primitivas.

A.1.8 On-to-Knowledge Methodology (OTKM) (SURE, STAAB e STUBER, 2003)

- **Origem e Propósito Principal:** Metodologia desenvolvida para a construção de ontologias para aplicações de gestão do conhecimento, com o foco em Processo de Conhecimento (Knowledge Process) e em Conhecimento do Processo Meta (Knowledge Meta Process). Na prática, tal metodologia foi aplicada em um estudo de caso em gestão de competências de uma empresa internacional localizada na Suíça ? a Swiss Life;

- **Domínio de aplicação:** Gestão do conhecimento empresarial;

- **Etapas:**

Conhecimento do Processo Meta (Knowledge Meta Process)

1. **Estudo de viabilidade:** identificação de problemas e oportunidades das áreas e potenciais soluções.
2. **Kickoff:** compreende dois passos:(i) captura da especificação de requisitos a partir do documento de especificação de requisitos da ontologia - Ontology Requirements Specification Document(ORSD); e (ii) criação da descrição de uma ontologia semi-formal.
3. **Refinamento:** engloba três passos: (i) refinamento da descrição da ontologia semi-formal; (ii) formalização da ontologia semi-formal para ontologia-alvo; (iii) criação do protótipo.
4. **Avaliação:** consiste na avaliação do projeto em três tipos de avaliação: (i) focada em tecnologia; (ii) focada no usuário; (iii) focada na ontologia.
5. **Aplicação e Evolução:** nesta etapa tem-se a aplicação da ontologia nos sistemas produtivos ou, mais especificamente, o uso dos sistemas baseados em ontologias. Além dessa tarefa, a evolução da ontologia como um processo organizacional.

Processo de Conhecimento (Knowledge Process)

1. Geração de conhecimento e/ou importação de documentos e meta-dados.
2. Captura dos itens de conhecimento a fim de elucidar a importância ou a interligação entre eles.
3. Recuperação e acesso aos conhecimentos requeridos.
4. Recuperação dos conhecimentos para posterior utilização em seu contexto.

A.1.9 Metodologia UP for ONtology (UPON) (DE NICOLA, MISSIKOFF e NAVIGLI, 2009)

- **Origem e Propósito Principal:** Metodologia de construção de ontologias derivada e baseada no padrão de engenharia de software conhecido como Processo Unificado ? do inglês Unified Software Development Process ou Unified Process (UP) ? do qual foram derivados metodologias de software como o Rational Unified

Process (RUP) e o PRocesso para Aplicativos eXtensíveis e Interativos (PRAXIS). Apesar da UP for ONtology poder ser usada em diferentes domínios do conhecimento, segundo seus autores, seu uso mais comum é no domínio do e-bussines;

- **Domínio de aplicação:** Negócios (e-business);
- **Etapas:**
 1. **Workflow de Requisitos:** envolve as seguintes atividades: (i) determinação do domínio de interesse e escopo da ontologia; (ii) definição do propósito do negócio ou cenário de motivação, com usuários e seus objetivos; (iii) especificação de um ou mais storyboards (contextos de modelagem e situação de maneira narrativa); (iv) criação de um Léxico de Análise, usando ferramentas automáticas para extração de conhecimento de documentos textuais (OntoLearn, Text-to-Onto, etc.); (v) identificação de questões de competência; (vi) identificação e priorização de casos de uso.
 2. **Workflow de Análise:** consiste no refinamento e estruturação dos requisitos ontológicos identificados no workflow anterior. Engloba as seguintes tarefas: (i) aquisição dos recursos do domínio e construção de um Léxico do Domínio; (ii) construção de um Léxico de Referência; (iii) modelagem do cenário de aplicação da ontologia usando diagramas UML; (iv) construção do Glossário de Referência.
 3. **Workflow de Desenvolvimento:** o objetivo deste workflow é construir a estrutura ontológica para o conjunto de entradas do Glossário de Referência. Para tanto, tem-se as seguintes atividades: (i) modelagem de conceitos, categorizando-os em três tipos primários: ator do negócio, objeto do negócio e processo do negócio; e dois complementares: mensagem e atributo; (ii) modelagem de hierarquias de conceitos e relacionamentos específicos do domínio.
 4. **Workflow de Implementação:** consiste em codificar a ontologia em uma rigorosa linguagem formal. Tem-se duas etapas neste workflow: (i) seleção de uma linguagem formal; e (ii) formalização da ontologia nesta linguagem, por exemplo em OWL.
 5. **Workflow de Teste:** o objetivo deste workflow é verificar a qualidade semântica e pragmática da ontologia desenvolvida. A qualidade pragmática é assegurada pela codificação em OWL, enquanto a qualidade semântica é medida pela verificação da consistência da ontologia, realizada da seguinte

maneira: verificação da cobertura da ontologia e das respostas da ontologia às questões de competência elaboradas anteriormente.

A.1.10 Metodologia NeON (SUARÉZ - FIGUEROA, 2010)

- **Origem e Propósito Principal:** Metodologia para construção de redes ontológicas baseado em um desenvolvimento colaborativo e argumentativo de ontologias. Tal metodologia foi desenvolvida em uma abordagem híbrida que combina o trabalho metodológico da área de Engenharia de Software e algumas metodologias para construção de ontologias, especificamente, a Methontology, a On-To-Knowledge, a DILIGENT e outros métodos ontológicos, como o de Grüninger e Fox. A NeOn Methontology inclui: (i) o Glossário NeOn de Processos e Atividades (ao todos tem-se 59 processos e atividades definidos), o qual identifica e define os processos e atividades potencialmente envolvidos quando redes ontológicas são construídas colaborativamente; e (ii) duas redes ontológicas sobre os modelos de ciclo de vida;

- **Domínio de aplicação:** Diversos

- **Etapas:**

Os 59 processos e atividades contidos no Glossário NeOn de Atividades e Processos são agrupados em três grandes grupos de atividades, conforme a seguir:

1. **Gerenciamento de processos e atividades:** inclui o escalonamento, controle e medida da qualidade da ontologia.
2. **Desenvolvimento orientado de processos e atividades:** é dividido em três partes:
 - **Pré-desenvolvimento de processos e atividades:** (i) estudo do ambiente; (ii) estudo de viabilidade; (iii) reuso de ontologias; (iv) reengenharia ontológica; (v) reuso de recursos não-ontológicos; e (vi) reengenharia de recursos não-ontológicos.
 - **Desenvolvimento de processos e atividades:** (i) especificação de requisitos da ontologia; (ii) conceitualização da ontologia; (iii) formalização; (iv) implementação da ontologia; (v) integração com outras ontologias; (vi) modularização da ontologia; (vii) reestruturação da ontologia; (viii) merge da ontologia; (ix) alinhamento da ontologia; (x) atualização da ontologia; (xi) modificação da ontologia; (xii) localização da ontologia; (xiii) tradução da ontologia; (xiv) anotação da ontologia; (xv) customização.

- **Pós-desenvolvimento de processos e atividades:** (i) atualização da ontologia; (ii) versionamento da ontologia; (iii) evolução da ontologia.
- 3. **Suporte aos processos e atividades:** engloba as seguintes etapas: (i) aquisição de conhecimento; (ii) avaliação da ontologia; (iii) documentação da ontologia; (iv) resumo da ontologia; (v) avaliação da ontologia; e (vi) gerenciamento da configuração da ontologia.

A.1.11 Metodologia MFPFO (LIM, LIU e LEE, 2011)

- **Origem e Propósito Principal:** Metodologia de construção de ontologia multi-facetada, anotada semanticamente, para a modelagem de uma família de produtos. Tal metodologia é capaz de sugerir, automaticamente, anotações semanticamente relacionadas, baseadas no design e no repositório de construção;
- **Domínio de aplicação:** Domínios que possuem uma família de produtos;
- **Etapas:**
 1. Construção de uma taxonomia da família de produtos;
 2. Extração de entidades;
 3. Identificação do conceito e geração da unidade facetada;
 4. Anotação semântica e modelagem faceta;
 5. Construção de uma ontologia de família de produtos multi-facetada e anotada semanticamente;
 6. Avaliação e validação da ontologia.

A.1.12 Ciclo de Vida de Schiessl e Bräscher (2011)

- **Origem e Propósito Principal:** Embora não seja propriamente uma metodologia de construção de ontologias ou associado ao ciclo de vida ontológico, Schiessl e Bräscher (2011) incluem todas as etapas necessárias no processo de construção de ontologias, destacando o papel de cada etapa e as tarefas contidas em cada uma delas;
- **Domínio de aplicação:** Diversos;
- **Etapas:**

1. **Especificação:** identifica o propósito e o âmbito da ontologia;
2. **Conceitualização:** descreve, em modelo conceitual, a ontologia a ser construída, de forma que atenda às especificações do passo anterior. O modelo conceitual de ontologia consiste no domínio de conceitos e as relações entre eles. As relações reforçam as conexões mais fortes entre grupos de conceitos. Os grupos de conceitos fortemente relacionados geralmente correspondem a diferentes módulos (subontologias) em que o domínio pode ser decomposto;
3. **Formalização:** transforma a descrição conceitual em modelo formal, isto é, a descrição do domínio no passo anterior é representada em linguagem formal, ainda que não seja a forma final. Conceitos são normalmente definidos através de axiomas que delimitam as interpretações possíveis para o significado desses conceitos. Conceitos são geralmente organizados hierarquicamente através de uma relação estruturante, tal como ?é-um? (classe-superclasse, instância-classe) ou ?parte-de?;
4. **Aplicação:** implementa a ontologia formalizada em linguagem de representação de conhecimento. Para isso, escolhe-se uma linguagem para representação e escreve-se o modelo formal na linguagem escolhida;
5. **Manutenção:** atualiza e corrige a ontologia aplicada. Reforça-se que o encadeamento das atividades não pretende ser o melhor recurso, pois não é este o foco. Elos são uma opção viável e consolidada por estudos anteriores e, como tudo em ciência, são passíveis de melhoramentos. Logo, esses passos não esgotam o tema, pois há atividades paralelas ao ciclo de vida que podem e devem ser realizadas. São elas:
 - **Aquisição de conhecimento:** adquire o conhecimento sobre o assunto utilizando técnicas de dedução junto aos especialistas de domínio ou por referência à bibliografia relevante. Várias técnicas podem ser utilizadas para a aquisição de conhecimentos, tais como o brainstorming, entrevistas, questionários, análise de textos, análise e técnicas de indução;
 - **Avaliação:** Julgamento técnico, baseado em técnicas disponíveis, da qualidade da ontologia;
 - **Documentação:** relata o que foi realizado, como foi feito e o porquê. A documentação associada aos termos representados na ontologia é particularmente importante não apenas para melhorar sua clareza, mas também para facilitar a manutenção, a utilização e a reutilização.

Apêndice B: Código Fonte da OntoExper-SPL

```

<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#"
           xml:base="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology"
           xmlns:owl="http://www.w3.org/2002/07/owl#"
           xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:xml="http://www.w3.org/XML/1998/namespace"
           xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
           xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
    <owl:Ontology rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology"/>
    <!--
    //////////////////////////////////////////////////////////////////
    // Object Properties
    //////////////////////////////////////////////////////////////////
    -->
    <!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#documentation -->
    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#documenta
        <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Abstract"
        <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Conclusion"
        <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Introduction"
        <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#RelatedWork"
        <rdfs:range rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Documenta
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#experiment -->
    <owl:ObjectProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#experiment"
        <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Acknowledgment"
        <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Analysis"
        <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Appendix"
        <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Discussion"
        <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Discussion"
        <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Documentation"
        <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Evaluation"
        <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#ExecutiveSummary"
        <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Experiment"
        <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Experiment"
        <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Package"
        <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Reference"
        <rdfs:range rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Experiment"
        <rdfs:range rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Experiment"
    </owl:ObjectProperty>
    <!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#typeContextExperiment -->

```

```

<owl:ObjectProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#typeContextSelection">
  <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TypeContextSelection"/>
  <rdfs:range rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Experiment"/>
  <rdfs:range rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Experiment"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#typeContextSelection -->
<owl:ObjectProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#typeDesignExperiment">
  <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TypeDesignExperiment"/>
  <rdfs:range rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Experiment"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#typeExperiment -->
<owl:ObjectProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#typeExperiment">
  <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TypeExperiment"/>
  <rdfs:range rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Experiment"/>
  <rdfs:range rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Experiment"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#typeExperimentSPL -->
<owl:ObjectProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#typeExperimentSPL">
  <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TypeExperimentSPL"/>
  <rdfs:range rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Experiment"/>
</owl:ObjectProperty>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#typeSelectionOfParticipants -->
<owl:ObjectProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#typeSelectionOfParticipants">
  <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TypeSelectionOfParticipants"/>
  <rdfs:range rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Experiment"/>
  <rdfs:range rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Experiment"/>
</owl:ObjectProperty>
<!--
///////////
// Data properties
// 
///////////
-->
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#abstractBackground -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#abstractBackground">
  <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#AbstractBackground"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#acknowledgements -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#acknowledgements">
  <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Acknowledgements"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#alternativeTechnologies -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#alternativeTechnologies">
  <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#AlternativeTechnologies"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>

```



```

<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#relevancePractice -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#relevan
    <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#RelatedW
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#researchObjective -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#rese
    <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Introduc
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#results -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#result
    <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Abstract
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#studyHasPerformMetaAnalysis -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#studyH
    <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Analysis
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#summary -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#summar
    <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Conclusion
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#tasks -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Tasks
    <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Experiment
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#technologyUnderInvestigation -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Technolog
    <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#RelatedWork
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#template -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Template
    <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Document
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#theAuthorsConcernedEvaluatingThe
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#theAutho
    <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Evaluation
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#threatsValidity -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Threats
    <rdfs:domain rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#Discussion
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#threatsValiditySPL -->
<owl:DatatypeProperty rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#ThreatsV

```



```

</owl:NamedIndividual>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#REAL -->
<owl:NamedIndividual rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#REAL">
  <rdf:type rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TypeExperiemnt">
</owl:NamedIndividual>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#REAL_PROBLEMS -->
<owl:NamedIndividual rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#REAL_PROBLEMS">
  <rdf:type rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TypeContext">
</owl:NamedIndividual>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#REPLICATED -->
<owl:NamedIndividual rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#REPLICATED">
  <rdf:type rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TypeExperiment">
</owl:NamedIndividual>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#SIMPLE_RANDOM_SAMPLING -->
<owl:NamedIndividual rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#SIMPLE_RANDOM_SAMPLING">
  <rdf:type rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TypeSelectivity">
</owl:NamedIndividual>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#SPECIFIC -->
<owl:NamedIndividual rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#SPECIFIC">
  <rdf:type rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TypeContext">
</owl:NamedIndividual>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#STRATIFIED_RANDOM_SAMPLING -->
<owl:NamedIndividual rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#STRATIFIED_RANDOM_SAMPLING">
  <rdf:type rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TypeSelectivity">
</owl:NamedIndividual>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#STUDENT -->
<owl:NamedIndividual rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#STUDENT">
  <rdf:type rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TypeContext">
</owl:NamedIndividual>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#SYSTEMATIC_SAMPLING -->
<owl:NamedIndividual rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#SYSTEMATIC_SAMPLING">
  <rdf:type rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TypeSelectivity">
</owl:NamedIndividual>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TOY -->
<owl:NamedIndividual rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TOY">
  <rdf:type rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TypeContext">
</owl:NamedIndividual>
<!-- http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TWO_FACTORS_WITH_TWO_TREATMENT -->
<owl:NamedIndividual rdf:about="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TWO_FACTORS_WITH_TWO_TREATMENT">
  <rdf:type rdf:resource="http://www.semanticweb.org/henrique/ontologies/2019/3/experiemnt-spl-ontology#TypeDesign">
</owl:NamedIndividual>
</rdf:RDF>
<!-- Generated by the OWL API (version 4.5.9.2019-02-01T07:24:44Z) https://github.com/owlcs/owlapi -->
```

Apêndice C: Código Fonte do Povoamento de Indíviduos na OntoExper-SPL

```

#!/usr/bin/env python
# coding: utf-8
# # CARREGA DADOS
# In[ ]:

import pandas as pd
import re
df = pd.read_excel("../Dominio/systematic_mapping_SPL_experiments_extracted_data.xlsx", sheet_name="Extracted Data", index_col=0)
df.shape
df.head()

# In[ ]:

# # normaliza os caracteres especiais
# def removeUnicodeChar(s):
#     if (type(s) == str):
#         return re.sub(''', ''', s)
#     return s
# In[ ]:

# def filter_nonprintable(text):
#     # import string
#     # # Get the difference of all ASCII characters from the set of printable characters
#     # nonprintable = set([chr(i) for i in range(128)]).difference(string.printable)
#     # Use translate to remove all non-printable characters
#     if (type(text) == str):
#         return text.translate({ord(character):None for character in nonprintable})
#     return text

# In[ ]:

# for c in df.columns:
#     df[c] = df.apply(lambda row: filter_nonprintable(row[c]), axis=1)

# In[ ]:

```

```

# splita os campos com mais de um valor
df['_informedSPL'] = df["Was the SPL source used informed? (Y/N) (If yes, which one?)"].str[0:1]
df['sourceSPL'] = df["Was the SPL source used informed? (Y/N) (If yes, which one?)"].str[1:]
df['_hasPilot'] = df["The pilot project was carried out? (Y/N) If Yes, how many?"].str[0:1]
df['_howManyPilot'] = df["The pilot project was carried out? (Y/N) If Yes, how many?"].str[1:]
df['_hasQuasiExperiment'] = df["Is it a quasi-experiment? (Y/N) If yes, is it explicit in the study?"].str[0:1]
df['quasiExperiment'] = df["Is it a quasi-experiment? (Y/N) If yes, is it explicit in the study?"].str[1:]

titleThreatsValidity = ('Threats to Validity (How is validity of the experimental results assured? '
                        'How was the data actually validated?) (Follow are the 4 threats proposed by Wohlin: '
                        'internal, external, construct and conclusion? (Y/N))')
df['_hasThreatsValidityByWolin'] = df[titleThreatsValidity].str[0:1]
df['threatsValidity'] = df[titleThreatsValidity].str[1:]

titlePakage = ('Is the experimental package informed? (Y/N) '
               '(If yes, what URL? And the link is still available? (Y/N))')
df['_hasPackage'] = df[titlePakage].str[0:1]
df['_package'] = df[titlePakage].str[1:]

# In[ ]:

# normaliza dados boolean
def convert.ToBoolean(s):
    return s in "Y" if True else False
def convertStringEmpty(s):
    if s == "---":
        return ""
    else:
        return s
def convertToNumber(s):
    if s.strip() and s is not None:
        return int(re.search('[0-9]', s).group())
    else: return 0

df['informedSPL'] = df.apply(lambda row: convert.ToBoolean(row._informedSPL), axis=1)
df['useTemplate'] = df.apply(lambda row: convert.ToBoolean(row["Does it use template? (Y/N)?"]), axis=1)
df['template'] = df.apply(lambda row: convertStringEmpty(row["If yes, what template?"]), axis=1)
df['hasAQuasiExperiment'] = df.apply(lambda row: convert.ToBoolean(row._hasQuasiExperiment), axis=1)
df['hasPilot'] = df.apply(lambda row: convert.ToBoolean(row._hasPilot), axis=1)
df['howManyPilot'] = df.apply(lambda row: convertToNumber(row._howManyPilot), axis=1)
df['hasQualitativeAnalysis'] = df.apply(lambda row: convert.ToBoolean(row["Do it have qualitative analysis of the experiments? (Y/N)?"]))
df['hasMetaAnalysis'] = df.apply(lambda row: convert.ToBoolean(row["Did the study perform meta-analysis? (Y/N)?"]))
df['hasThreatsValidityByWolin'] = df.apply(lambda row: convert.ToBoolean(row._hasThreatsValidityByWolin), axis=1)
df['hasAuthorEvaluating'] = df.apply(lambda row: convert.ToBoolean(row["The authors were concerned with evaluating "
                                                               "the quality of the experiments? (Y/N)?"]))
df['hasExperimentalPackage'] = df.apply(lambda row: convert.ToBoolean(row._hasPackage), axis=1)

# In[ ]:

df_spl = df.loc[(df["SPL Name used"] != '---') & (df["Is it Real or Academic SPL?"] != '---') & (df.informedSPL)]
df_exp = pd.concat([df, df_spl]).drop_duplicates(keep=False)

```

```

print(df_spl.shape, df_exp.shape)

# # CARREGA ONTOLOGIA
# In[ ]:

def registreExperimentCommons(instance, r):
    instance.title.append(r["Title"])
    instance.authorship.append(r["Authorship"])
    instance.publicationYear.append(r["Publication year"])
    instance.publicationType.append(r["Publication type"])
    instance.publicationVenue.append(r["Publication venue"])
    instance.pagesNumber.append(r["Pages number"])

def registreExperiment(i, r):
    instance = onto.Experiment("experiment_{}".format(i))
    instance.idExperiment.append(i)
    registreExperimentCommons(instance, r)
    return instance

def registreExperimentSPL(i, r):
    instance = onto.ExperimentSPL("experimentSPL_{}".format(i))
    instance.idExperimentSPL.append(i)
    registreExperimentCommons(instance, r)
    instance.nameSPLUsed.append(r["SPL Name used"])
    instance.wasTheSPLSourceUsedInformed.append(r.sourceSPL)
    field_type_experiment_SPL = r["Is it Real or Academic SPL?"]
    type_experiment = next((x for x in onto.TypeExperimentSPL.instances()
                           if x.name.lower() in field_type_experiment_SPL.lower()), onto.ACADEMY)
    instance.typeExperimentSPL.append(type_experiment)
    return instance

# In[ ]:

def registreDocumentation(i, r):
    instance = onto.Documentation("documentation_{}".format(i))
    instance.idDocumentation.append(i)
    instance.useTemplate.append(r.useTemplate)
    instance.template.append(r.template)
    instance.observationsAboutTemplateUsed.append(r["Observations about the template used"])
    return instance

def registreAbstract(i, r):
    instance = onto.Abstract("abstract_{}".format(i))
    instance.idAbstract.append(i)
    instance.objective.append(r["Objective (What is the question addressed with this research?)"])
    instance.abstractBackground.append(r["Abstract - Background (Why is this research important?)"])
    instance.methods.append(r["Methods (What is the statistical context and methods applied?)"])
    instance.results.append(r["Results (What are the main findings?Practical implications?)"])
    instance.limitations.append(r["Limitations (What are the weakness of this research?)"])
    instance.conclusions.append(r["Conclusions (What is the conclusion?)"])
    instance.keywords.append(r["Keywords"])
    return instance

def registreIntroduction(i, r):
    instance = onto.Introduction("introduction_{}".format(i))
    instance.idIntroduction.append(i)
    instance.problemStatement.append(r["Problem statement (What is the problem? Where does it occur? Who has observed")]
    instance.researchObjective.append(r["Research objective (GQM) (What is the research question to be answered by th

```

```

instance.context.append(r["Context (What information is necessary to understand whether the research relates to a specific field or discipline?)"])
return instance

def registreRelatedWork(i, r):
    instance = onto.RelatedWork("relatedWork_{}".format(i))
    instance.idRelatedWork.append(i)
    instance.technologyUnderInvestigation.append(r["Technology under investigation (What is necessary for a reader to know about the technology being studied?)"])
    instance.alternativeTechnologies.append(r["Alternative technologies (How does this research relate to alternative technologies?)"])
    instance.relatedStudies.append(r["Related studies (How this research relates to existing research (studies)? What were the findings?)"])
    instance.relevancePractice.append(r["Relevance to practice (How does it relate to state of the practice?)"])
    return instance

def registreConclusionsFutureWork(i, r):
    instance = onto.ConclusionsFutureWork("conclusionsFutureWork_{}".format(i))
    instance.idConclusionsFutureWork.append(i)
    instance.summary.append(r["Technology under investigation (What is necessary for a reader to know about the technology being studied?)"])
    instance.impact.append(r["Alternative technologies (How does this research relate to alternative technologies?)"])
    instance.futureWork.append(r["Related studies (How this research relates to existing research (studies)? What were the findings?)"])
    return instance

# In[ ]:

def registreExperimentPlanningCommons(instance, r):
    instance.goals.append(r["Goals (Formalization of goals, refine the important constructs of the experiment's goal)"])
    instance.experimentalUnits.append(r["Experimental Units (From which population will the sample be drawn? How will the sample be selected?)"])
    instance.experimentalMaterial.append(r["Experimental Material (Which objects are selected and why?)"])
    instance.tasks.append(r["Tasks (Which tasks have to be performed by the subjects?)"])
    instance.hypotheses.append(r["Hypotheses (What are the constructs and their operationalization? They have to be testable and falsifiable)"])
    instance.parameters.append(r["Parameters (What are the constructs and their operationalization? They have to be testable and falsifiable)"])
    instance.variables.append(r["Variables (What are the constructs and their operationalization? They have to be testable and falsifiable)"])
    instance.experimentDesign.append(r["Experiment Design (What type of experimental design has been chosen?)"])
    instance.procedure.append(r["Procedure (How will the experiment (i.e., data collection) be performed? What instruments will be used?)"])
    instance.analysisProcedure.append(r["Analysis Procedure (How will the data be analyzed?)"])
    instance.isAQuasiExperiment.append(r.hasAQuasiExperiment)
    if (len(r["quasiExperiment"]) > 1):
        instance.explicitQuasiExperimentInStudy.append(r.quasiExperiment.strip())
field_type_experiment = r["Is it an Original or Replicated Experiment?"]
type_experiment = next((x for x in onto.TypeExperiment.instances() if x.name.lower() in field_type_experiment.lower()))
instance.typeExperiment.append(type_experiment)
field_selection_participant = r["How was the selection of participants/experimental objects? - Simple random sampling"]
type_selection_participants = next((x for x in onto.TypeSelectionParticipantsObjects.instances() if x.name.lower() in field_selection_participant.lower()))
instance.typeSelectionOfParticipants.append(type_selection_participants)
field_type_context = r["Context of the experiment (in vivo, in vitro, ...)"]
type_context = next((x for x in onto.TypeContextExperiment.instances() if x.name.lower() in field_type_context.lower()))
instance.typeContextExperiment.append(type_context)
field_type_design = r["Design Experimental: - One factor with two treatments; - One factor with more than two treatments"]
type_design = next((x for x in onto.TypeDesignExperiment.instances() if x.name.lower() in field_type_design.lower()))
instance.typeDesignExperiment.append(type_design)

def registreExperimentPlanning(i, r):
    instance = onto.ExperimentPlanning("experimentPlanning{}".format(i))
    instance.idExperimentPlanning.append(i)
registreExperimentPlanningCommons(instance, r)
    return instance

def registreExperimentPlanningSPL(i, r):
    instance = onto.ExperimentPlanningSPL("experimentPlanningSPL_{}".format(i))
    instance.idExperimentPlanningSPL.append(i)

```

```

registreExperimentPlanningCommons(instance, r)
    instance.artifactSPLused.append(r["SPL artifact used"])
    field_type_context_selection = r["Context Selection (Off-line vs. on-line, Student vs. professional, Toy vs. real")
    type_context_selection = next((x for x in onto.TypeContextSelection.instances() if x.name.lower() in field_type_c
    instance.typeContextSelection.append(type_context_selection)
    return instance

# In[ ]:

def registreExecutionSection(i, r):
    instance = onto.ExecutionSection("executionSection_{}".format(i))
    instance.idExecutionSection.append(i)
    instance.preparation.append(r["Preparation (What has been done to prepare the execution of the experiment (i.e.,")
    instance.deviations.append(r["Deviations from the Plan (Describe any deviations from the plan, e.g., how was the")
    instance.pilotProjectCarriedOut.append(r.hasPilot)
    instance.howManyPilotProjectCarriedOut.append(r.howManyPilot)
    return instance

# In[ ]:

def registreAnalysis(i, r):
    instance = onto.Analysis("analysis_{}".format(i))
    instance.idAnalysis.append(i)
    instance.descriptiveStatistics.append(r["Descriptive statistics (What are the results from descriptive statistics?)"])
    instance.datasetPreparation.append(r["Data set preparation (What was done to prepare the data set, why, and how?)"])
    instance.hypothesisTesting.append(r["Hypothesis testing (How was the data evaluated and was the analysis model valid?)"])
    instance.howDatahasBeenAnalyzed.append(r["How the data has been analyzed? (Ex: Correlation, Hypothesis Test, meta-analysis)"])
    instance.whatQualitativeAnalysisPerformed.append(r["If yes, what qualitative analysis was performed?"])
    instance.hasQualitativeAnalysisOfExperiment.append(r.hasQualitativeAnalysis)
    instance.studyHasPerformMetaAnalysis.append(r.hasMetaAnalysis)
    return instance

# In[ ]:

def registreDiscussionCommons(instance, r):
    instance.evaluationOfResultsAndImplications.append(r["Evaluation of Results and Implications (Explain the results and implications of the study)"])
    instance.inferences.append(r["Inferences (Inferences drawn from the data to more general conditions)"])
    instance.lessonsLearned.append(r["Lessons learned (Which experience was collected during the course of the experiment)"])
    instance.isFollowThreatsByWohlin.append(r.hasThreatsValidityByWolin)
    instance.threatsValidity.append(r.threatsValidity)

def registreDiscussion(i, r):
    instance = onto.Discussion("discussion{}".format(i))
    instance.idDiscussion.append(i)
    registreDiscussionCommons(instance, r)
    return instance

def registreDiscussionSPL(i, r):
    instance = onto.DiscussionSPL("discussionSPL_{}".format(i))
    instance.idDiscussionSPL.append(i)
    registreDiscussionCommons(instance, r)
    instance.threatsValiditySPL.append(r["Threats to validity in SPL"])
    return instance

# In[ ]:

```

```

def registreAcknowledgements(i, r):
    instance = onto.Acknowledgements("acknowledgements_{}".format(i))
    instance.idAcknowledgements.append(i)
    instance.acknowledgements.append(r["Acknowledgements Section (Sponsors, participants, and contributors who do not"))
    return instance
def registreReferences(i, r):
    instance = onto.References("references_{}".format(i))
    instance.idReferences.append(i)
    instance.references.append(r["References Section (All cited literature has to be presented in the format requested"))
    return instance
def registreAppendices(i, r):
    instance = onto.Appendices("appendices_{}".format(i))
    instance.idAppendices.append(i)
    instance.appendices.append(r["Appendices Section (Experimental materials, raw data, and detailed analyses, which"))
    return instance
def registreEvaluation(i, r):
    instance = onto.Evaluation("evaluation_{}".format(i))
    instance.idEvaluation.append(i)
    instance.theAuthorsConcernedEvaluatingTheQuality.append(r.hasAuthorEvaluating)
    return instance
def registrePackage(i, r):
    instance = onto.Package("package_{}".format(i))
    instance.idPackage.append(i)
    instance.isExperimentalPackageInformed.append(r.hasExperimentalPackage)
    pks = r._package.split('\n')
    instance.url.append(pks[len(pks) - 2])
    instance.isLinkAvailable.append(convertToBoolean(pks[len(pks) - 1]))
    return instance

# In[ ]:

def registerCommons(exp, idx, row):
    documentation = registreDocumentation(idx, row)
    documentation.experiment.append(exp)
    abstract = registreAbstract(idx, row)
    abstract.documentation.append(documentation)
    introduction = registreIntroduction(idx, row)
    introduction.documentation.append(documentation)
    relatedWork = registreRelatedWork(idx, row)
    relatedWork.documentation.append(documentation)
    conclusionsFutureWork = registreConclusionsFutureWork(idx, row)
    conclusionsFutureWork.documentation.append(documentation)
    executionSection = registreExecutionSection(idx, row)
    executionSection.experiment.append(exp)
    analysis = registreAnalysis(idx, row)
    analysis.experiment.append(exp)
    acknowledgements = registreAcknowledgements(idx, row)
    acknowledgements.experiment.append(exp)
    references = registreReferences(idx, row)
    references.experiment.append(exp)
    appendices = registreAppendices(idx, row)
    appendices.experiment.append(exp)
    evaluation = registreEvaluation(idx, row)

```

```

evaluation.experiment.append(exp)
package = registrePackage(idx, row)
package.experiment.append(exp)

# In[ ]:

from owlready2 import *
onto = get_ontology("file:///home/henrique/Dropbox/UEM-Mestrado/Pesquisa/Source/Ontologia/protege/experiment-spl-ontology")

# In[ ]:

for idx, row in df_spl.iterrows():
    experimentSPL = registreExperimentSPL(idx, row)
    experimentPlanningSPL = registreExperimentPlanningSPL(idx, row)
    experimentPlanningSPL.experiment.append(experimentSPL)
    discussion = registreDiscussionSPL(idx, row)
    discussion.experiment.append(experimentSPL)
    registerCommons(experimentSPL, idx, row)

# In[ ]:

# with pd.option_context('display.max_rows', None, 'display.max_columns', None): # more options can be specified also
#     print(df)

# In[ ]:

for idx, row in df_exp.iterrows():
    experiment = registreExperiment(idx, row)
    experimentPlanning = registreExperimentPlanning(idx, row)
    experimentPlanning.experiment.append(experiment)
    discussion = registreDiscussion(idx, row)
    discussion.experiment.append(experiment)
    registerCommons(experiment, idx, row)

# In[ ]:

import sys; print(sys.getdefaultencoding())

# In[ ]:

onto.save('/home/henrique/Dropbox/UEM-Mestrado/Pesquisa/Source/Ontologia/protege/experiment-spl-ontology-02072019.owl')

# In[ ]:

assert len(onto.Experiment.instances()) == df.shape[0]

# In[ ]:

for i in onto.Experiment.instances(): print(i, i.idExperiment)

# In[ ]:

for i in onto.Package.instances(): print(i.idPackage, i.isExperimentalPackageInformed, i.url, i.isLinkAvailable)

```

```
# In[ ]:

for i in onto.Discussion.instances(): print(i.isFollowThreatsByWohlin)

# In[ ]:

for i in onto.ExperimentPlanning.instances(): print(i.isAQuasiExperiment, i.explicitQuasiExperimentInStudy)
```

Apêndice D: Questionário de Avaliação e Respostas Aplicado à OntoExper-SPL

Este apêndice apresenta o instrumento de avaliação enviado para os especialistas de LPS e/ou Ontologias responderem citados na Seção 2.5.

Figura 4.1: Instrumento de Avaliação pagina 1

Avaliação da SMartOntology

*Obrigatório

1. Endereço de e-mail *

2. Nome completo *

3. Instituição *

4. Títuloção *

Marcar apenas uma oval.

- Ensino Superior - Incompleto
- Ensino Superior - Completo
- Especialização
- Mestrado
- Doutorado
- Pós-Doutorado

5. Tempo em anos que trabalha com experimentação de software *

A SMartOntology

A imagem "Grafo da SMartOntology", apresenta a ontologia no formato de grafo, onde pode-se notar a representação de suas entidades e seus relacionamentos por meio das propriedade de objetos. Mais informações podem ser encontrada no documento "capítulo-3-dissertação.pdf" do pacote "smart-ontology-metadatas.zip" anexado abaixo, onde está descrito todo o processo de construção do modelo da ontologia e povoamento da mesma

A SMartOntology se basea no modelo conceitual desenvolvido para experimento de software em LPS. A imagem "Clusterização do modelo conceitual" apresenta esse modelo agrupado pelos elementos experimentais proposto por Wohlin.

No trabalho de "Ontology Evaluation" de Denny Vrandečić estão definidos alguns critérios para avaliação de ontologias em geral. Esta avaliação aplica esses critérios à SMartOntology. Esses critérios são, precisão, adaptabilidade, clareza, completude, eficiência computacional, concisão, consistência e capacidade organizacional. Para cada critério há um enunciado elucidativo sobre o que o critério avalia, bem como um simples exemplo de sua implementação.

Para cada critério existe uma questão relacionada. Utilizamos a escala Likert abaixo para responder cada questão:

- 1 - Discordo totalmente
- 2 - Discordo parcialmente
- 3 - Nem concordo nem discordo (neutro)
- 4 - Concordo parcialmente
- 5 - Concordo totalmente

Link para metadados

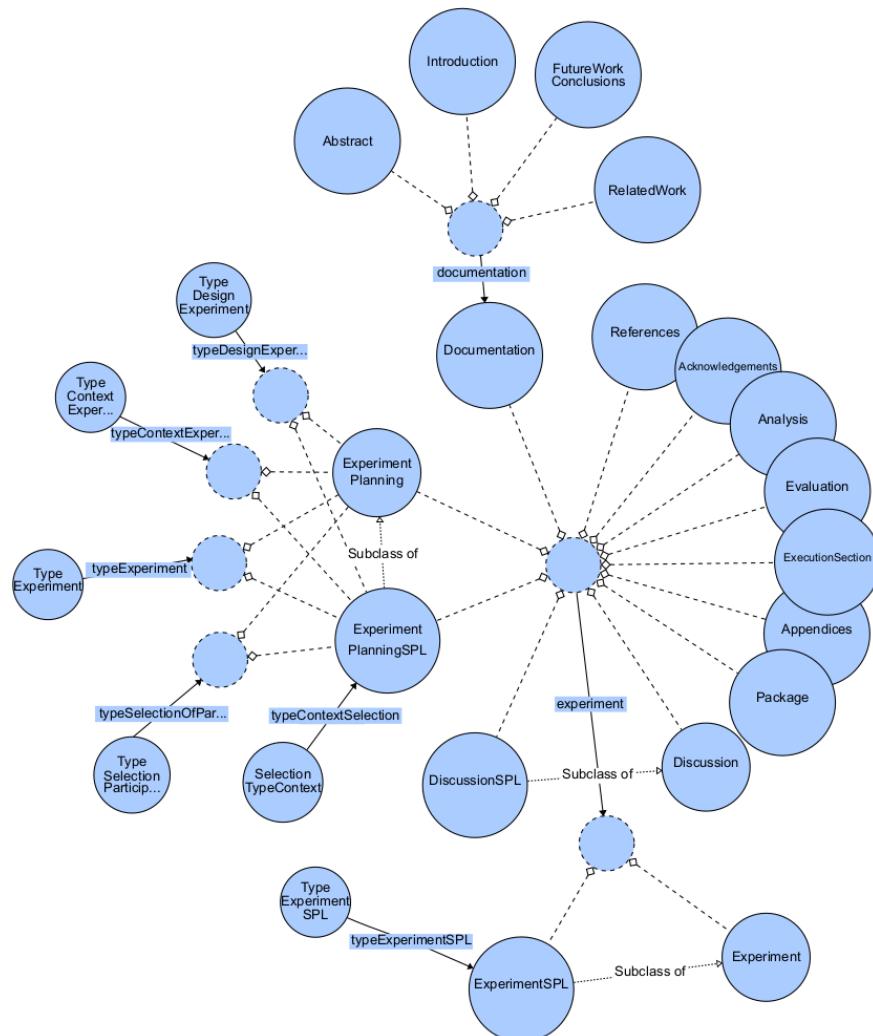
Fonte: o Autor

Figura 4.2: Instrumento de Avaliação pagina 2

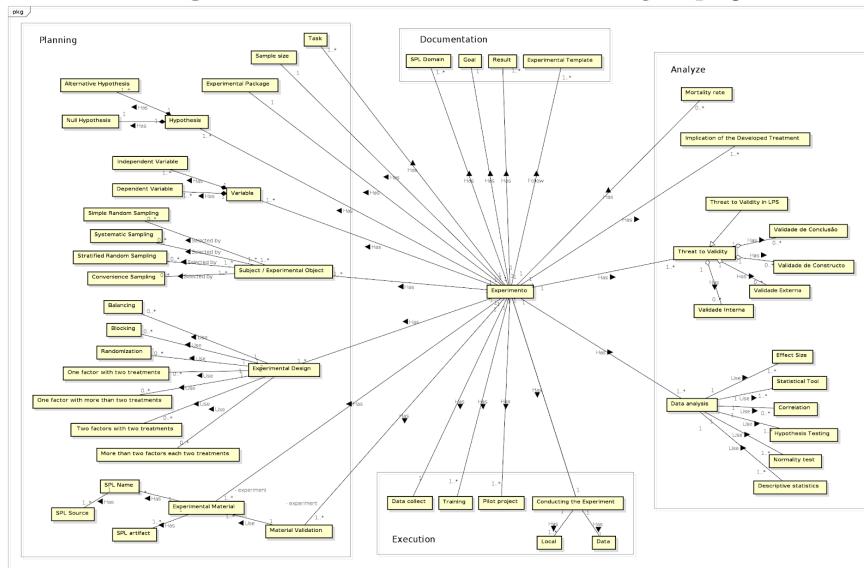
Baixe os artefato neste link.

https://drive.google.com/open?id=11wkVjGCVMvvs2lvqHMicYOis1eBv7j_g

Eles servirão de apoio para responder as questões.

Grafo da SMartOntology**Clusterização do modelo conceitual**

Fonte: o Autor

Figura 4.3: Instrumento de Avaliação pagina 3

Precisão

Critério que determina se os axiomas da ontologia estão em conformidade com o conhecimento das partes interessadas sobre o domínio. Uma maior precisão vem das definições e descrições corretas de classes, propriedades e indivíduos. A correção neste caso pode significar conformidade com “padrões-ouro” definidos, sejam outras fontes de dados, conceituações ou mesmo realidade. Ceusters e Smith (2006), introduzem uma abordagem para usar a realidade como referência, ou seja, os termos da ontologia capturam as partes pretendidas da realidade. Os axiomas devem restringir as possíveis interpretações de uma ontologia para que os modelos resultantes sejam compatíveis com as conceituações dos usuários.

6. Podemos afirmar que a SMartOntology é precisa em sua definição. *

Por exemplo: o axioma ExperimentPlanningSPL está refletindo a realidade do termo usado para a fase de planejamento de experimentos de software para linha de produto de software.
Marcar apenas uma oval.

1	2	3	4	5
Discordo totalmente				
<input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> Concordo totalmente				

Axiomas da SMartOntology

```
classes =
[smart-ontology.Abstract,
smart-ontology.ConclusionsFutureWork,
smart-ontology.Introduction,
smart-ontology.RelatedWork,
smart-ontology.Documentation,
smart-ontology.Acknowledgements,
smart-ontology.Analysis,
smart-ontology.Appendices,
smart-ontology.Discussion,
smart-ontology.DiscussionSPL,
smart-ontology.Evaluation,
smart-ontology.ExecutionSection,
smart-ontology.ExperimentPlanning,
smart-ontology.ExperimentPlanningSPL,
smart-ontology.Package,
smart-ontology.References,
smart-ontology.Experiment,
smart-ontology.ExperimentSPL,
smart-ontology.TypeContextExperiment,
smart-ontology.TypeContextSelection,
```

Fonte: o Autor

Figura 4.4: Instrumento de Avaliação pagina 4

```

smart-ontology.TypeDesignExperiment,
smart-ontology.TypeExperiment,
smart-ontology.TypeExperimentSPL,
smart-ontology.TypeSelectionParticipantsObjects]

object_properties =
[smart-ontology.documentation,
smart-ontology.experiment,
smart-ontology.typeContextExperiment,
smart-ontology.typeContextSelection,
smart-ontology.typeDesignExperiment,
smart-ontology.typeExperiment,
smart-ontology.typeExperimentSPL,
smart-ontology.typeSelectionOfParticipants]

data_properties =
[smart-ontology.abstractBackground,
smart-ontology.acknowledgements,
smart-ontology.alternativeTechnologies,
smart-ontology.analysisProcedure,
smart-ontology.appendices,
smart-ontology.artifactSPLUsed,
smart-ontology.authorship,
smart-ontology.conclusions,
smart-ontology.context,
smart-ontology.datasetPreparation,
smart-ontology.descriptiveStatistics,
smart-ontology.deviations,
smart-ontology.evaluationOfResultsAndImplications,
smart-ontology.experimentAnalysisBasedPValue,
smart-ontology.experimentDesign,
smart-ontology.experimentDomain,
smart-ontology.experimentalMaterial,
smart-ontology.experimentalUnits,
smart-ontology.explicitQuasiExperimentInStudy,
smart-ontology.futureWork,
smart-ontology.goals,
smart-ontology.hasQualitativeAnalysisOfExperiment,
smart-ontology.howDataHas Been Analyzed,
smart-ontology.howManyPilotProjectCarriedOut,
smart-ontology.hypotheses,
smart-ontology.hypothesisTesting,
smart-ontology.idAbstract,
smart-ontology.idAcknowledgements,
smart-ontology.idAnalysis,
smart-ontology.idAppendices,
smart-ontology.idConclusionsFutureWork,
smart-ontology.idDiscussion,
smart-ontology.idDiscussionSPL,
smart-ontology.idDocumentation,
smart-ontology.idEvaluation,
smart-ontology.idExecutionSection,
smart-ontology.idExperiment,
smart-ontology.idExperimentPlanning,
smart-ontology.idExperimentPlanningSPL,
smart-ontology.idExperimentSPL,
smart-ontology.idIntroduction,
smart-ontology.idPackage,
smart-ontology.idReferences,
smart-ontology.idRelatedWork,
smart-ontology.impact,
smart-ontology.inferences,
smart-ontology.isAQuasiExperiment,
smart-ontology.isExperimentalPackageInformed,
smart-ontology.isFollowThreatsByWohlin,
smart-ontology.isLinkAvailable,
smart-ontology.keywords,
smart-ontology.lessonsLearned,
smart-ontology.limitations,
smart-ontology.methods,
smart-ontology.nameSPLUsed,
smart-ontology.objective,
smart-ontology.observationsAboutTemplateUsed,
smart-ontology.pagesNumber,
```

Fonte: o Autor

Figura 4.5: Instrumento de Avaliação pagina 5

```

smart-ontology.parameters,
smart-ontology.pilotProjectCarriedOut,
smart-ontology.preparation,
smart-ontology.problemStatement,
smart-ontology.procedure,
smart-ontology.publicationType,
smart-ontology.publicationVenue,
smart-ontology.publicationYear,
smart-ontology.references,
smart-ontology.relatedStudies,
smart-ontology.relevancePractice,
smart-ontology.researchObjective,
smart-ontology.results,
smart-ontology.studyHasPerformMetaAnalysis,
smart-ontology.summary,
smart-ontology.tasks,
smart-ontology.technologyUnderInvestigation,
smart-ontology.template,
smart-ontology.theAuthorsConcernedEvaluatingTheQuality,
smart-ontology.threatsValidity,
smart-ontology.threatsValiditySPL,
smart-ontology.title,
smart-ontology.url,
smart-ontology.urlOfPackage,
smart-ontology.useTemplate,
smart-ontology.variables,
smart-ontology.wasTheSPLSourceUsedInformed,
smart-ontology.whatQualitativeAnalysisPerformed]

```

Adaptabilidade

Critério que mede até que ponto a ontologia antecipa seus usos. Uma ontologia deve oferecer a base conceitual para uma série de tarefas antecipadas (idealmente, na Web, também deve oferecer a base para tarefas nunca antecipadas). Deve ser possível estender e especializar a ontologia monotonicamente, ou seja, sem a necessidade de remover axiomas (observe que em OWL, a monotonicidade semântica é dada pela monotonicidade sintática, ou seja, para retrair inferências, axiomas explícitos e explícitos precisam ser retraídos). Uma ontologia deve reagir de forma previsível e intuitiva a pequenas mudanças nos axiomas. Deve permitir metodologias para extensão, integração e adaptação, ou seja, incluir metadados necessários. Novas ferramentas e situações inesperadas devem poder usar a ontologia.

7. A S^MartOntology oferece uma base conceitual para antecipar seu uso. É possível estendê-la e especializá-la sem a necessidades de alteração.*

Por exemplo: Assim como a classe smart-ontology.ExperimentPlanningSPL é uma extensão de smart-ontology.ExperimentPlanning posso criar uma mais uma extensão para smart-ontology.ExperimentPlanningSPLAGM, modelando especificações de experimentos voltados para LPS AGM, sem a necessidade de remoção de nenhum outro axioma. Ou ainda criar mais uma especificação smart-ontology.ExperimentPlanningSystemOfSystem para representar a fase de planejamento de experimentos voltados para área de sistema de sistemas.
Marcar apenas uma oval.

1	2	3	4	5
Discordo totalmente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Concordo totalmente			

Classes disponíveis para extensão

```

classes =
[smart-ontology.Abstract,
smart-ontology.ConclusionsFutureWork,
smart-ontology.Introduction,
smart-ontology.RelatedWork,
smart-ontology.Documentation,
smart-ontology.Acknowledgements,
smart-ontology.Analysis,
smart-ontology.Appendices,
smart-ontology.Discussion,
smart-ontology.DiscussionSPL,
smart-ontology.Evaluation,
smart-ontology.ExecutionSection,
smart-ontology.ExperimentPlanning].

```

Fonte: o Autor

Figura 4.6: Instrumento de Avaliação pagina 6

```

smart-ontology.ExperimentPlanningSPL,
smart-ontology.Package,
smart-ontology.References,
smart-ontology.Experiment,
smart-ontology.ExperimentSPL,
smart-ontology.TypeContextExperiment,
smart-ontology.TypeContextSelection,
smart-ontology.TypeDesignExperiment,
smart-ontology.TypeExperiment,
smart-ontology.TypeExperimentSPL,
smart-ontology.TypeSelectionParticipantsObjects]

```

Clareza

Critério que mede com que eficácia a ontologia comunica o significado pretendido dos termos definidos. As definições devem ser objetivas e independentes do contexto. Os nomes dos elementos devem ser compreensíveis e inequívocos. Uma ontologia deve usar definições em vez de descrições para classes. As entidades devem ser documentadas o suficiente e estar totalmente rotuladas em todos os idiomas necessários. Axiomas complexos devem ser documentados. As escolhas de representação não devem ser feitas para a conveniência da notação ou implementação, ou seja, o viés de codificação deve ser minimizado.

8. A SSmartOntology é eficaz em comunicar seus termos, suas definições são objetivas e independente do contexto. *

Por exemplo: O axioma smart-ontology.nameSPLUsed por si só é uma definição sobre o nome da SPL usada.

Marcar apenas uma oval.

1	2	3	4	5			
Discordo totalmente		<input type="radio"/>	Concordo totalmente				

Completude

Critério que mede se o domínio de interesse está coberto adequadamente (abrangência). Todas as perguntas que a ontologia deve responder podem ser respondidas. Existem diferentes aspectos da completude: (i) Completude no que diz respeito ao idioma (tudo o que é declarado que poderia ser declarado, está usando o idioma fornecido?), (ii) Completude no que diz respeito ao domínio (todos os indivíduos estão presentes, todos os conceitos relevantes são capturados?), (iii) Completude com em relação aos requisitos de aplicação (todos os dados necessários estão presentes?). A abrangência também abrange a granularidade e a riqueza da ontologia.

9. A SSmartOntology cobre os aspectos de completude, completude do idioma, completude de domínio, completude aos requisitos de aplicação e abrangência com relação a Experimentação para Engenharia de Software em LPS *

Por exemplo: para experimento em LPS é importante saber sobre qual ameaças a validade foi destacado para LPS, isso pode ser encontrado no axioma smart-ontology.threatsValiditySPL como propriedade de objeto do axioma smart-ontology.DiscussionSPL

Marcar apenas uma oval.

1	2	3	4	5			
Discordo totalmente		<input type="radio"/>	Concordo totalmente				

Eficiência Computacional

Critério que mede a capacidade das ferramentas de trabalhar usadas na ontologia, em particular a velocidade que os processadores (resonators) precisam para executar as tarefas necessárias, seja resposta de consulta, classificação ou verificação de consistência. Alguns tipos de axiomas podem causar problemas para certos processadores (resonators). O tamanho da ontologia também afeta a eficiência da ontologia.

10. A SSmartOntology possui uma eficiência computacional satisfatória.

Por exemplo: Consultas complexa SPARQL possui um tempo de resposta menor que 2s

Marcar apenas uma oval.

1	2	3	4	5			
Discordo totalmente		<input type="radio"/>	Concordo totalmente				

Fonte: o Autor

Figura 4.7: Instrumento de Avaliação pagina 7

11. Query SPARQL executada

Concisão

É o critério que determina se a ontologia inclui elementos irrelevantes em relação ao domínio a ser coberto (ou seja, uma ontologia sobre livros, incluindo axiomas sobre leões africanos) ou representações redundantes da semântica. Uma ontologia deve impor um compromisso ontológico mínimo, ou seja, especificar a teoria mais fraca possível. Somente termos essenciais devem ser definidos. As suposições subjacentes da ontologia sobre o domínio mais amplo (especialmente sobre a realidade) devem ser tão fracas quanto possível, a fim de permitir a reutilização interna e a comunicação entre as partes interessadas que se comprometem com diferentes teorias.

12. A SMartOntology não possui elementos irrelevantes para cobertura de seu domínio. *

Por exemplo: por a SMartOntology estar direcionada para domínio de experimento em LPS não existe nenhum axioma voltado para experimentos em Sistemas de Sistemas. os axiomas para LPS estão restritos a elementos pertencentes a este domínio. Por exemplo a só existe a propriedade de dados smart-ontology.threatsValiditySPL dentro do axioma smart-ontology.DiscussionSPL.
Marcar apenas uma oval.

1	2	3	4	5
Discordo totalmente <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> Concordo totalmente				

Consistência

Descreve que a ontologia não inclui ou permite contradições. Enquanto a precisão declara a conformidade da ontologia com uma fonte externa, a consistência indica que a própria ontologia pode ser interpretada. A consistência lógica é apenas uma parte dela, mas também as descrições formais e informais na ontologia devem ser consistentes, ou seja, a documentação e os comentários devem estar alinhados com os axiomas. Outros princípios de consistência podem ser definidos, como as restrições OntoClean à taxonomia (Guarino e Welty, 2002).

13. A SMartOntology não inclui ou permite contradições. (consultar ontologia) *

Por exemplo: podemos afirmar que a classe smart-ontology.DiscussionSPL é uma definição especializada de smart-ontology.Discussion exclusivamente para experimentos em LPS, logo não é possível afirmar que a classe smart-ontology.DiscussionSPL é uma especialização de smart-ontology.ExperimentPlanningSPL.
Marcar apenas uma oval.

1	2	3	4	5
Discordo totalmente <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> <input type="radio"/> Concordo totalmente				

Capacidade Organizacional

Este critério agrupa vários critérios que decidem, com que facilidade uma ontologia pode ser implantada dentro de uma organização. Ferramentas, bibliotecas, fontes de dados e outras ontologias usadas restringem a ontologia, e a ontologia deve atender a essas restrições. As ontologias são frequentemente especificadas usando uma metodologia de engenharia de ontologia ou usando conjuntos de dados específicos. Os metadados da ontologia podem descrever as metodologias, ferramentas e fontes de dados aplicadas a organização. Esses metadados podem ser usados pela organização para decidir se uma ontologia deve ser aplicada ou não.

Figura 4.8: Instrumento de Avaliação pagina 8

14. Os metadados da SMartOntology podem descrever as metodologias, ferramentas e fontes de dados aplicadas e uma organização. (consultar metadados em smart-ontology-metadata.zip)*

Por exemplo: o formato da ontologia .owl é um formato padronizado que diz sobre a sua sintaxe de implementação. Outra meta dado é a origem dos dados (planilha systematic_mapping_SPL_experiments_extracted_data.xlsx) que diz sobre a origem dos dados populado na ontologia.

Marcar apenas uma oval.

	1	2	3	4	5	
Discordo totalmente	<input type="radio"/>	Concordo totalmente				

Comentário geral

Descreva outras observações encontrada na SMartyOntology que achar relevante para esta avaliação.

15. Comentário

Powered by
 Google Forms

Fonte: o Autor

Apêndice D: Código Fonte dos Exemplos de Aplicação de Recomendação Usando OntoExper-SPL

E.1 Arquivo settings.py

```
"""
Django settings for recsys project.

Generated by 'django-admin startproject' using Django 2.2.5.

For more information on this file, see
https://docs.djangoproject.com/en/2.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/2.2/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'kw5akwqdaq^&&k2uq=e8^pr!ij1le8$1va8htv3m*##6(khb98'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []
```

```

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rating',
]
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
]

ROOT_URLCONF = 'recsys.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ['templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
]

WSGI_APPLICATION = 'recsys.wsgi.application'

# Database
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

```

```

# Password validation
# https://docs.djangoproject.com/en/2.2/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

# Internationalization
# https://docs.djangoproject.com/en/2.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.2/howto/static-files/

STATIC_URL = '/static/'

ONTOLOGY_FILE = 'ontology-populate-valid.owl'

```

E.2 Arquivo views.py

```

import os
from django.http import HttpResponse
from django.shortcuts import render
from owlready2 import *
from django.conf import settings
from rating.models import Rating
from .collaborativeFilter import CollaborativeFilter

```

```

FILTER_SOURCE_SPL = 'sources_spl'
FILTER_TYPE_EXP_SPL = 'type_experiments_spl'

def home(request):
    onto = get_ontology(os.path.join(settings.BASE_DIR, settings.ONTOLOGY_FILE)).load()

    rec_sys = CollaborativeFilter()
    recs = rec_sys.recomendation(request.user.username)
    print('rec-sys', recs)

    recomendations = []
    for rec in recs[:4]:
        id_exp = rec[1]
        exp = (onto.search(idExperiment = id_exp), onto.search(idExperimentSPL = id_exp))[len(onto.search(idExperiment =
    recomendations.append({'id': id_exp, 'title': exp[0].title[0]})

    data = {
        'recomendations': recomendations,
        'onto': onto,
        'type_experiments': _getTypesFrom(onto.TypeExperiment),
        'type_context_experiment': _getTypesFrom(onto.TypeContextExperiment),
        'type_context_selection': _getTypesFrom(onto.TypeContextSelection),
        'type_design_experiment': _getTypesFrom(onto.TypeDesignExperiment),
        'type_participants': _getTypesFrom(onto.TypeSelectionParticipantsObjects),
        FILTER_TYPE_EXP_SPL: _getTypesFrom(onto.TypeExperimentSPL),
        FILTER_SOURCE_SPL: ['AGM', 'SPLOT', 'SIMPLE', 'ArgoUML', 'Test'],
        # 'user': request.user.username
    }
    return render(request, 'index.html', data)

def filter_experiment(request):

    onto = get_ontology(os.path.join(settings.BASE_DIR, settings.ONTOLOGY_FILE)).load()

    filters = eval(request.GET.get('filters'))

    sourceSPL = ''
    typeExperimentsSpl = ''
    result = []

    for f in filters:
        if f.get('key') == FILTER_SOURCE_SPL:
            sourceSPL = f.get('val')

        if f.get('key') == FILTER_TYPE_EXP_SPL:
            typeExperimentsSpl = f.get('val')

    instance_typeExperimentSPL = list(filter(
        lambda el: el._name == typeExperimentsSpl, onto.TypeExperimentSPL.instances()))

    result = onto.search(is_a=onto.Abstract,
                         documentation=onto.search(is_a=onto.Documentation,

```

```

experiment=onto.search(wasTheSPLSourceUsedInformed="*%s*" % sourceSPL,
                      typeExperimentSPL=instance_typeExperimentSPL)

experiments = []
for r in result:
    experiments.append(
        {
            'id_experiment': r.idAbstract[0],
            'title': r.documentation[0].experiment[0].title[0],
            'abstract': r.abstractBackground[0],
        }
    )

data = {'experiments': experiments}
return render(request, 'experiments.html', data)

def rating_experiment(request):
    onto = get_ontology(os.path.join(settings.BASE_DIR, settings.ONTOLOGY_FILE)).load()
    id_experiment = int(request.GET.get('id'))
    rating = int(request.GET.get('rating'))
    username = request.user.username

    try:
        rating_persist = Rating.objects.get(user=username, id_experiment=id_experiment)
        rating_persist.rating = rating
        rating_persist.save()
        retorno = 'update'
    except Rating.DoesNotExist:
        rating_persist = Rating(user=username,
                               id_experiment=id_experiment,
                               rating=request.GET.get('rating'),
                               title_experiment=onto.search(is_a=onto.ExperimentSPL, idExperimentSPL=id_experiment)[0].title[0])
        rating_persist.save()
        retorno = 'create'

    return HttpResponse(retorno)

def _getTypesFrom(clazz):
    o_types = clazz.instances()
    types = map(lambda t: t._name, o_types)
    return list(types)

```

E.3 Arquivo collaborativeFilter.py

```

import os
from django.conf import settings
from owlready2 import *
from rating.models import Rating

```

```

import pandas as pd
import numpy as np

class CollaborativeFilter():

    def __init__(self):
        onto = get_ontology(os.path.join(settings.BASE_DIR,
                                         settings.ONTOLOGY_FILE)).load()

        dataset = {}
        for user, ratings in self.__get_ratings().items():
            experiments = {}

            for exp in onto.Experiment.instances():
                try:
                    idExperiment = exp.idExperiment[0]
                    if idExperiment in ratings.keys():
                        experiments[idExperiment] = ratings[idExperiment]
                    else:
                        experiments[idExperiment] = None
                except IndexError:
                    idExperiment = exp.idExperimentSPL[0]
                    if idExperiment in ratings.keys():
                        experiments[idExperiment] = ratings[idExperiment]
                    else:
                        experiments[idExperiment] = None

            if user in list(dataset.keys()):
                dataset[user].update(experiments)
            else:
                dataset[user] = experiments

        self.df = pd.DataFrame(dataset)
        # print(self.df)

    def __get_ratings(self):
        user_ratings = {}
        resultset = Rating.objects.all()

        for rs in resultset:
            if rs.user in list(user_ratings.keys()):
                user_ratings[rs.user].update({rs.id_experiment: rs.rating})
            else:
                user_ratings[rs.user] = {rs.id_experiment: rs.rating}

        return user_ratings

    def euclidean(self, usr1, usr2):
        self.df['sqr'] = self.df.loc[:, [usr1, usr2]].apply(
            lambda item: np.power(item[0] - item[1], 2), axis=1)

        return 1 / (1 + np.sqrt(self.df['sqr'].sum()))

```

```
def recommendation(self, usr):
    df_usr = self.df.loc[:, [usr]]
    experiments = df_usr[df_usr[usr].isnull()].index

    totals = {}
    sum_similarity = {}
    for other in self.df.drop(columns=usr).columns:
        similarity = self.euclidean(usr, other)

        for item in experiments:
            if np.isnan(self.df[other][item]):
                continue

            totals.setdefault(item, 0)
            totals[item] += self.df[other][item] * similarity

            sum_similarity.setdefault(item, 0)
            sum_similarity[item] += similarity

    rankings = [(total / sum_similarity[item], item)
                for item, total in totals.items()]

    rankings.sort()
    rankings.reverse()

    return rankings
```