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) é fundamental para 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 dos experimentos, visando aumentar a confiabilidade dos seus resultados. Por mais que eles tem abordado a qualidade de experimentos controlados de forma geral, ainda não há evidencias que estão analisando em contextos específicos, como é o caso de Linhas de produto de Software (LPS). Neste caso, ainda existe uma falta de instrumentação e medição especifica da qualidade dos experimentos em LPS. Por isso torna-se necessário fornecer um corpo de conhecimento confiável e replicável no contexto de LPS. Devido essa importância, projetar, executar e analisar os resultados de um experimento em LPS torna-se crucial para garantir a qualidade dos mesmos. Neste sentido propomos uma ontologia (OntoExper-LPS) para experimentos em LPS, pois possui centenas de experimentos publicados. A ontologia é concebida principalmente com base em diretrizes definidas e é projetada usando linguagem OWL (*Ontology Web Language*), suportada pelo ambiente Protégé para verificação de sintaxe e avaliação inicial. A OntoExper-LPS foi preenchida com mais de 200 experimentos em linhas de produtos de software, reunidos em um estudo de mapeamento sistemático. Com este trabalho surge também a oportunidade de investigar a elaboração de um sistema de recomendação para experimentos em LPS se baseando sobre inferencias aplicada OntoExper-LPS. Portanto, este trabalho apresenta conceitos fundamentais para elaboração de uma ontologia voltada para experimentos em LPS, e como prova de conceito, um sistema de recomendação para experimentos em LPS. Acreditamos que essa ontologia bem como o sistema de recomendação pode contribuir para documentar melhor os elementos essenciais de um experimento, promovendo assim a repetição, a replicação e a reprodutibilidade dos experimentos. Que por meio destes, possa levar qualidade para os projetos experimentais e resultados obtidos da modelagem ontológica e do sistema de recomendação. Apresentamos uma avaliação da OntoExper-LPS com relação a 8 critérios de qualidade, onde foi aplicado um questionário e enviado à especialistas da área. Os resultados dessa avaliação são apresentados em um capitulo a parte. Ao final temos um sistema de recomendação implementado como prova de conceito como uso de inferência na OntoExper-LPS, que apresenta bons resultados em recomendações para experimentos controlado utilizando

métodos híbridos de sistemas de recomendação. Espera-se com este projeto, contribuir com a comunidade de LPS no sentido de melhorar os projetos e execução de experimentos, aumentando a confiança do corpo de conhecimento visando a transferência de tecnologia para indústria.

Palavras-chave: Experimento Controlado de Software. Linha de Produto de Software. Ontologia. Ontologia em Engenharia de Software. Qualidade de Experimentos. Sistemas 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: Controlled Software Experiment. Ontology. Quality of Experiments. Software Product Line. Systems of Recommendation in Software Engineering. Systems of Recommendation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

ALM: *Application Lifecycle Management*
API: *Application Programming Interface*
AQ: Avaliação de Qualidade
CBF: *Content-based Filterin*
ES: Engenharia de Software
ESE: Experimentação em Engenharia de Software
ETL: *Extract, Transform, Load*
GRSSE: Grupo de pesquisa em Reuso Sistemático de Software e Experimentação
ID: Identificador Único
LPS: Linha de Produto de Software
PDI: *Pentaho Data Integration*
POO: Programação Orientado a Objetos
RSSE: *Recommendation System in Software Engineering*
OWL: *Ontology Web Language*
SPARQL: *Protocol and RDF Query Language*
MTV: *(Model, Template, View)*
MVC: *(Model, Vew, Controller)*
SGBD: Sistemas de Gestão de Base de Dados
VD: Variável Dependente
DT: Discordo Totalmente
DP: Discordo Parcialmente
N: Nem Concordo nem Discordo (neutro)
CP: Concordo Parcialmente,
CT: Concordo Totalmente

SUMÁRIO

1

Introdução

1.1 Contextualização

Foi discutido que a 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 melhora-se a qualidade dos experimentos [1] que por sua vez vem construir um corpo de conhecimento confiável e referente na área de experimentação em ES.

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.

Ontologias estão entre os métodos mais utilizados para formalizar informações. Ontologias são representações formais de uma abstração contendo definições formais de nomenclatura, conceitos, propriedades e relações entre os conceitos, a fim de definir

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

um vocabulário controlado de termos e relações de conceitos [referência]. Assim, o uso de ontologias para representar informações sobre experimentos, padroniza os dados facilitando a interoperabilidade, a troca de informações e a replicação de experimentos.

Por mais atraente que seja, uma solução de ontologia para toda ES é muito esparsa. Dadas as particularidades de cada área em ES, seria complexo projetar uma ontologia capaz de representar todas as características particular de cada área em um único trabalho. Dessa forma, concentramos nossos esforços no campo da LPS, devido à nossa experiência em grupo, uma vez que um mapeamento sistemático foi realizado em centenas de experimentos publicados em LPS (Furtado, 2018). (Grupo de Pesquisa em Reutilização Sistemática de Software e Experimentação Contínua - GReater).

Uma LPS é determinado por um conjunto e produtos de um determinado segmento de mercado (Northrop et al., 2007), como um sistema para aparelhos celulares, onde há ativos centrais com as principais funcionalidades que o software deve implementar, chamadas semelhanças, e pode haver um conjunto de funcionalidades específicas para determinados dispositivos, chamados variabilidades. 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.

Portanto, a definição de uma ontologia para experimentos específicos em LPS, se destina a auxiliar outros pesquisadores no desenvolvimento e replicação de novos 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 os alunos e professores, 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, se baseando em consultas e recomendações de experimentos correlacionadas a sua área de atuação.

Portanto, o objetivo deste trabalho de mestrado é propor uma ontologia para representar formalmente essa diretriz, a fim de padronizar os experimentos de ES em SPL. Esperamos facilitar a representação dos dados de experimentos para aumentar o uso geral de experimentação em ES, aumentar o número de replicações e o compartilhamento de dados.

Resultados obtidos com a avaliação da qualidade e viabilidade da ontologia indicam a possibilidade de criar inferências aos dados e diretrizes sobre os experimentos, dessa forma é possível criar como prova de conceito, modelos de inferência para um sistema de recomendação sobre esta proposta de ontologia.

1.2 Motivação e Justificativa

Realizar um experimento em LPS exige alguns pontos de atenção específicos para garantir a qualidade do experimento. Estes pontos foram investigados em um trabalho de mestrado do nosso - GReater. Neste trabalho foi elaborado diretrizes para a determinar a qualidade de experimentos em LPS. Esta 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, 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, modelar uma ontologia que representa formalmente o conhecimento adquirido até o estado da arte sobre experimentos de ES em LPS. Chamaremos essa proposta de **OntoExper-LPS**, uma ontologia para apoiar experimentos em LPS.

Os objetivos específicos deste projeto são:

- gerar e representar um conjunto de metadados a partir das informações sobre experimentos em LPS;
- definir um modelo de ontologia para experimento de ES em LPS;

- provar ontologia com experimentos de ES em LPS;
- avaliar a qualidade e viabilidade do modelo proposto;
- elaborar um sistema de recomendação como prova de conceito do modelo de ontologia; e
- avaliar e empacotar o modelo, os resultados da avaliação e o sistema de recomendação.

1.4 Metodologia de Desenvolvimento

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

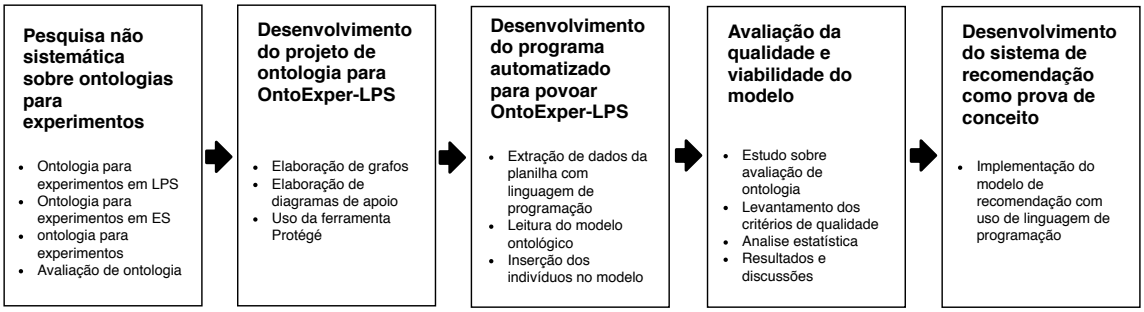

Figura 1.1: Etapas da Metodologia de Desenvolvimento de Pesquisa

- **Pesquisa não sistemática sobre ontologias para experimentos:** nesse momento foi realizado estudos não sistemático para encontrar trabalhos relacionados descrevendo ontologias para experimentos de ES em LPS e avaliação de ontologia, foram encontrado alguns estudos que serviram de apoio a essa pesquisa, eles estão descritos na Seção 2.6 do Capítulo 2;
- **Desenvolvimento do projeto de ontologia para OntoExper-LPS:** trata sobre desenvolvimento da OntoExper-LPS, que incluiu realizar a escolha das tecnologias e ferramenta 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-LPS:** 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 encontrado na literatura, e 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 quantitativo, com objetivo de avaliar a qualidade da OntoExper-LPS, para isso foi elaborado um questionários com oito critérios de qualidade, enviado para os especialistas responderem, e posteriormente realizado uma análise das respostas, bem como pontos de melhorias encontrado durante a avaliação.

- **Desenvolvimento do sistema de recomendação como prova de conceito:** após processar os indivíduos na OntoExper-LPS, foi possível elaborar como prova de conceito, um sistema de recomendação utilizando o 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, experimentais e quasi-experimentos em ES, qualidade de experimentos e quasi-experimentos em ES, Ontologias e Sistemas de Recomendação; o Capítulo 3 apresenta uma Ontologia para Experimentos em LPS - OntoExper-LPS, destacando desde a concepção ao povoamento da ontologia proposta e uma pré-análise de pontos de falha da ontologia; o Capítulo 4 apresenta uma avaliação quantitativa sobre a qualidade da OntoExper-LPS por especialistas da área; o Capítulo 5 apresenta um sistema de recomendação para experimentos em LPS, usando como modelo de dados a OntoExper-LPS; e o Capítulo 6 apresenta as considerações finais acerca deste projeto de dissertação de mestrado, bom como as suas contribuições, limitações e trabalhos futuros.

2

Fundamentação Teórica

2.1 Considerações Iniciais

Este capítulo apresenta conceitos importantes sobre a fundamentação teórica, necessárias para a compreensão desta pesquisa como LPS, experimento e *quasi*-experimento em ES, qualidade de experimentos e *quasi*-experimento em ES, ontologia, sistemas de recomendação e trabalhos relacionados. Dessa forma a estrutura deste capítulo fica, na Seção 2.2 apresenta os principais pontos de LPS e variabilidades; na Seção 2.3 é apresentado uma visão geral de experimento e *quasi*-experimento em ES; na Seção 2.4 são apresentados infromações preliminares sobre qualidade de experimentos e *quasi*-experimento em ES; A Seção 2.5 descreve os fundamentos sobre o que é uma ontologia e sua finalidade; Seção 2.6 são discutidos os trabalhos relacionados; e na Seção 2.7 são apresentadas as Considerações finais deste capítulo.

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 (Northrop et al., 2007). 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 por um conjunto de características comuns chamadas de similaridades, e características variáveis chamadas de variabilidades (Van der Linden

et al., 2007). Este núcleo forma a base da LPS que determina a Arquitetura de uma LPS, que são eles, componentes reusáveis, modelos de domínios, requisitos da LPS, planos de testes e modelos de características de variabilidades.

O modelo de características contém todas as características de uma LPS e os seus inter-relacionamentos. De acordo com Apel, "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 (Apel, 2013).

- Variabilidades são descritas por: ponto de variação que permite a resolução de variabilidades em artefatos genéricos de uma LPS, e;
- Variantes são representadas pelos: possessíveis 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 (Halmans e Pohl, 2003).

O *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 algumas das seguintes características, opções de criar, visualizar, remover e editar a legenda da imagem. As características variantes 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 grafo com o Modelo de Características da LPS *MobileMedia*. As arestas com círculos preenchido representa as características pertencentes ao *core asset*. 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;

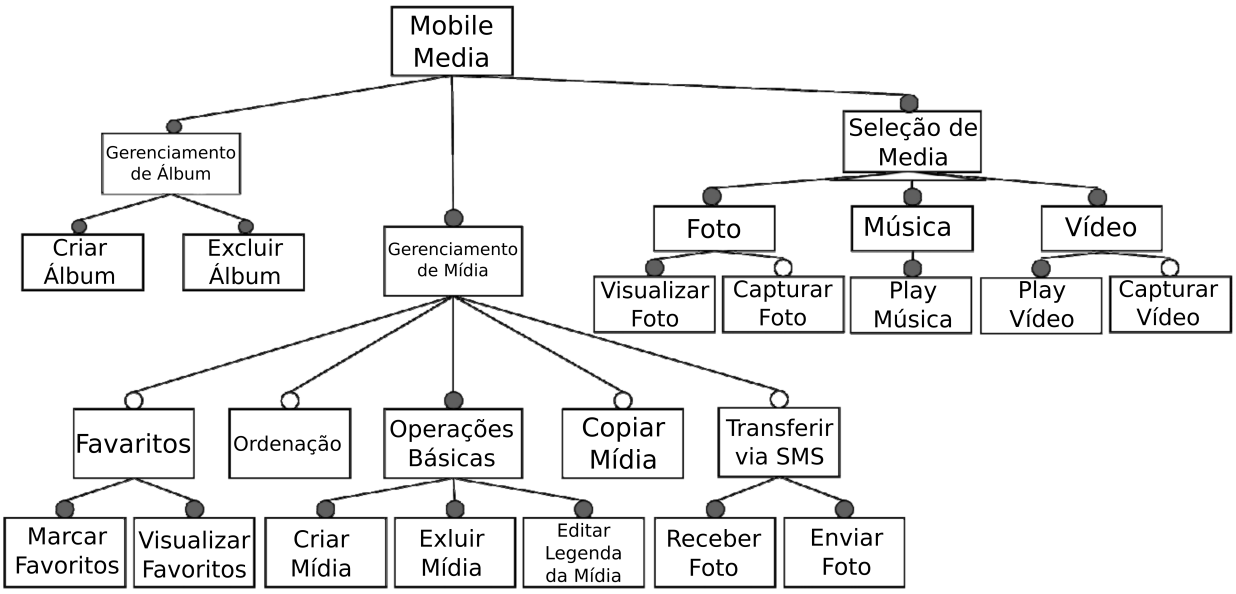

Figura 2.1: Um modelo de Características. Traduzido de Sommerville (2011)

  - Operações Básicas

    Criar Mídia

    Excluir Mídia

    Editar Legenda Mídia

  - Favoritos;

    Marcar Favoritos

    Visualizar Favoritos

- Seleção de Mídia;

  Foto

  Visualizar Foto

  Capturar Foto

Pohl, Böckle e van Der Linden 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. No qual, é composto de cinco subprocessos principais, sendo eles: 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 linha de produto. No qual, é composto pelos subprocessos: Engenharia de Requisitos da Aplicação, Projeto da Aplicação, Realização da Aplicação e Teste da Aplicação.

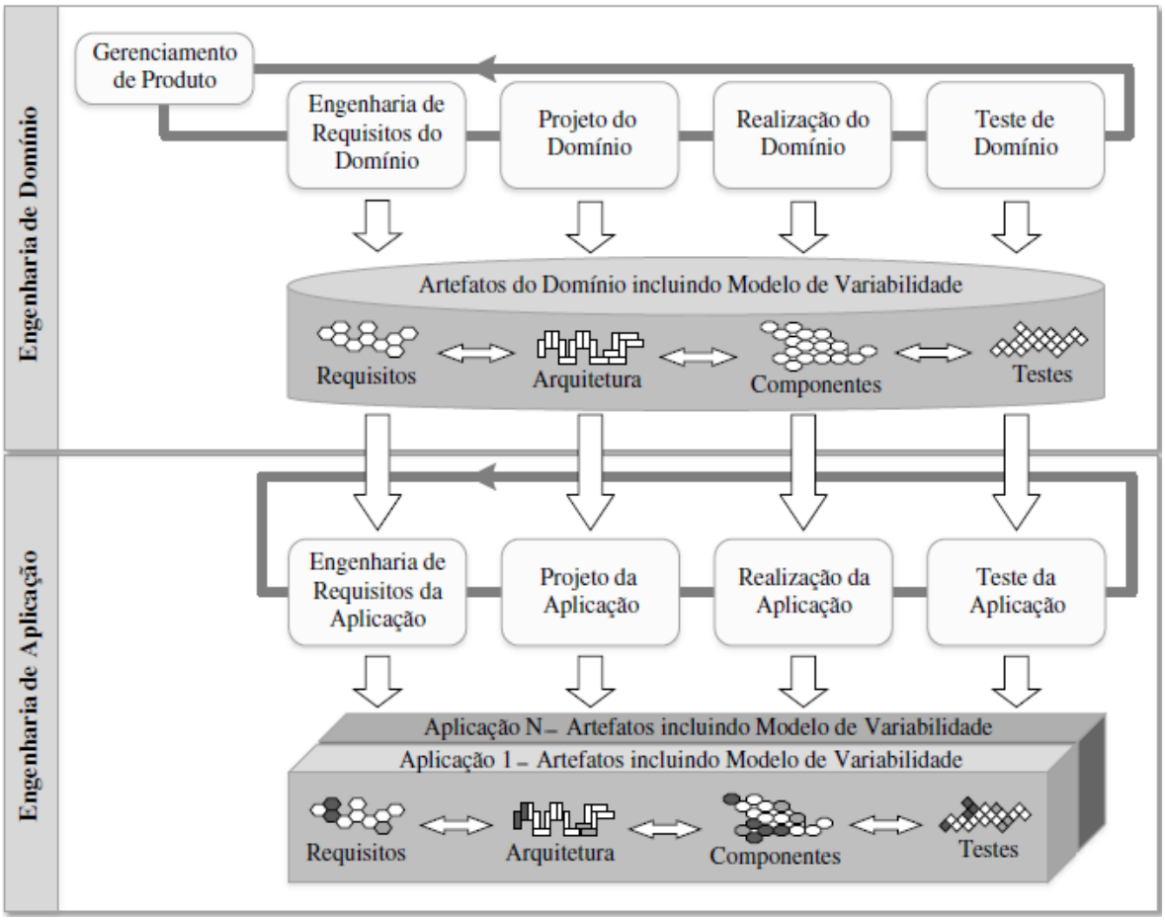

Figura 2.2: *Framework* de Engenharia de LPS (Pohl et al., 2005). Traduzido por Geraldi (2015)

2.3 Experimentos e Quasi-Experimento 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).

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

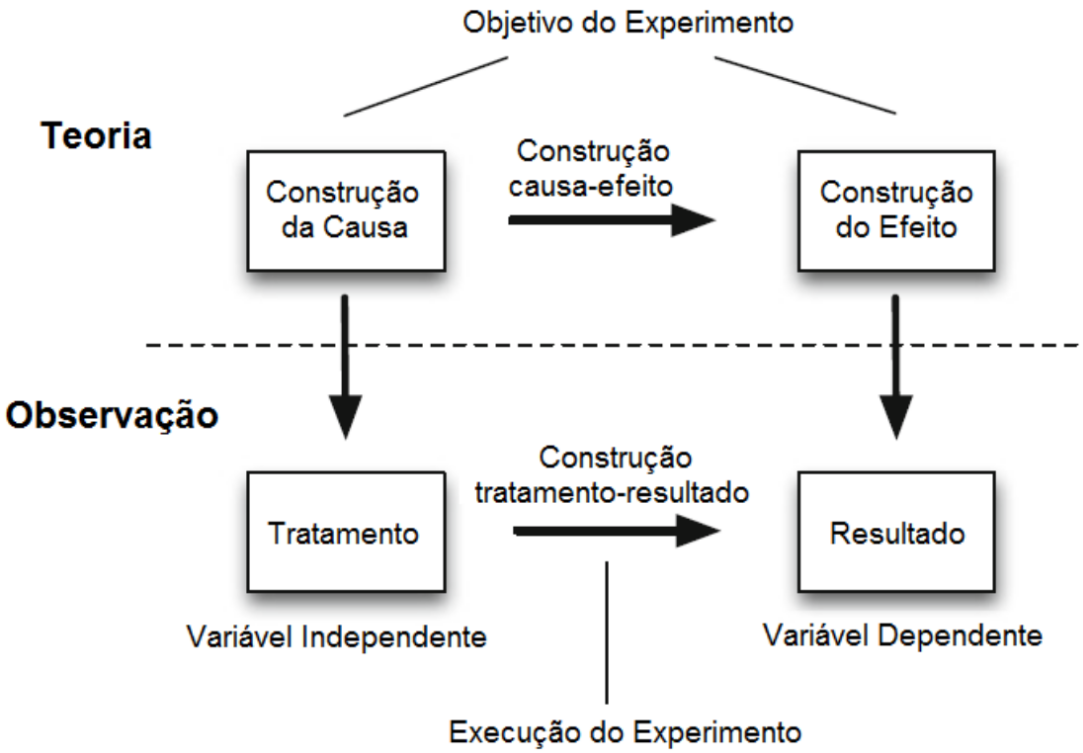

Figura 2.3: Conceitos Essenciais de um Experimento. Traduzido de Wohlin et al. (2012a)

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

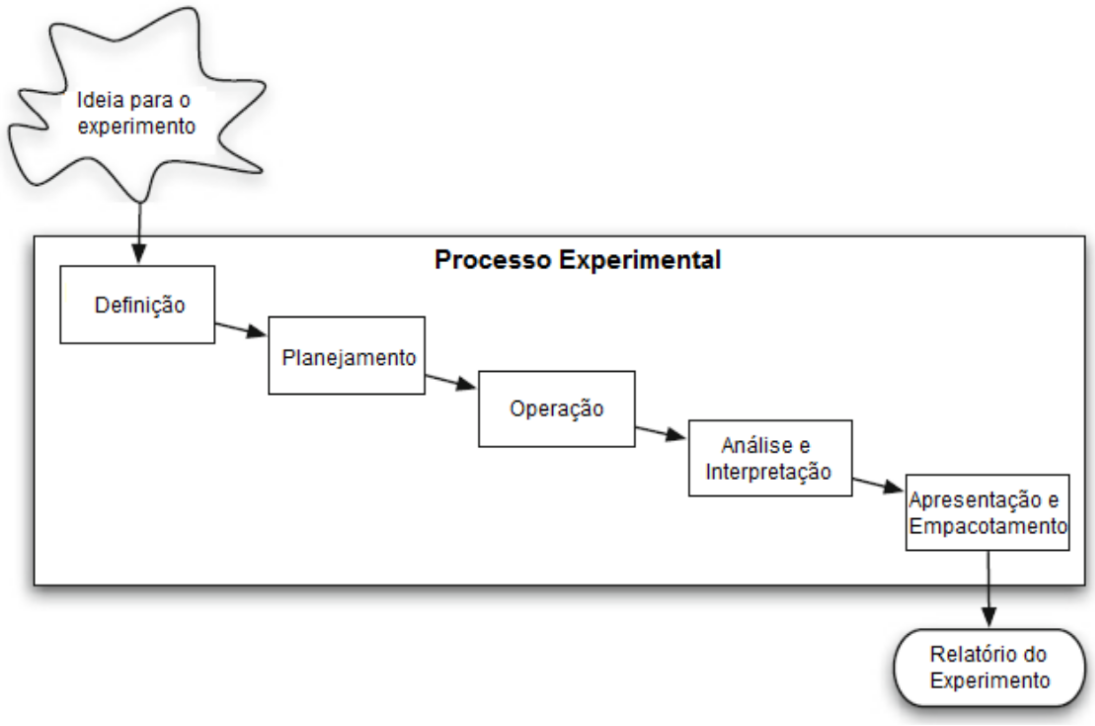

Figura 2.4: Visão Geral do Processo Experimental. Traduzido de Wohlin et al. (2012a)

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; 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 pôde 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 Qualidade de Experimentos em Engenharia de Software

Segundo Dieste e Juristo (2013), o conceito de qualidade de experimentos em ES pode ser visto em dois pontos de vista diferentes, o primeiro é considerar a qualidade como o resultado da validade interna de um bom experimento e o segundo é tornar a qualidade operacional assim como a quantidade de vieses nos resultados experimentais. Outro ponto colocado por Dieste e Juristo, é a validade externa que também tem uma função chave ao analisar se um experimento tem boa qualidade, porém essa função é contrária à validade interna.

Na Tabela - 2.1 são apresentadas as definições dos conceitos de qualidade citados anteriormente.

Tabela 2.1: Definições dos conceitos de qualidade em experimentos em Engenharia de Software. Traduzido de Kitchenham et al. (2007)

Termo	Sinônimo	Definição
Viés	Erro sistemático	Uma tendência para produzir resultados que partem sistematicamente de resultados "verdadeiros". Resultados sem viés são válidos internamente.
Validade Interna	Validade	O alcance em que o projeto e a condução do estudo são possíveis de evitar erro sistemático. A validade interna é um pré-requisito para a validade externa.
Validade Externa	Generabilidade, Aplicabilidade	O alcance em que os efeitos observados no estudo são aplicáveis fora do estudo.

Segundo Dieste e Juristo (2011) os experimentos de boa qualidade são aqueles livres de vieses. O viés está relacionado com a validade interna, por exemplo, quão bem os experimentos são planejados, executados e analisados (Dieste e Juristo, 2013). Para minimizar os vieses existem alguns métodos como, aleatorização para criar grupos experimentais homogêneos, imparcialidade para alocar os indivíduos. Desta forma os resultados podem

ser analisados mesmo depois do experimento ter sido realizado com replicações. Enquanto os experimentos de baixa qualidade seriam os que usam pouco ou nenhum dos métodos citados (Dieste e Juristo, 2011).

Como o viés não pode ser medido, existem algumas abordagens para avaliá-lo. Os instrumentos de Avaliação de Qualidade (AQ), são projetados para avaliar a validade interna e inferir a qualidade de experimentos, tais como, abordagens simples (questionários), *checklists* (contém ou não contém), escalas de qualidade, opinião de especialista (Dieste e Juristo, 2011, 2013; Teixeira, 2014).

Por outro lado a qualidade de um experimento em ES também pode ser avaliada considerando o projeto e análise dos experimentos, em termos de poder estatístico, análise do tamanho de efeito (resultado), *quasi*-experimentais e relatório de experimento (Kampenes, 2007).

Até o momento não se tem conhecimento de trabalhos que tratam a qualidade de experimentos em área específica de ES. Entretanto, foram recuperados alguns trabalhos, por meio de pesquisas não sistemáticas, que estão relacionados com a avaliação de qualidade dos experimentos em Engenharia de Software, descritos a seguir:

- Kitchenham et al. (2007) propõem um *checklist* de avaliação de qualidade de experimentos em ES contendo cinquenta questões para avaliar a qualidade de experimentos, em que sugerem que os pesquisadores selecionam apenas as questões do *checklist* mais adequadas ao contexto de suas próprias questões de pesquisa.

- Kitchenham et al. (2015) apresentam um *checklist* de avaliação de qualidade de experimentos em ES com nove questões, onde cada questão possui subquestões categorizadas em: (i) "Questões sobre objetivo", (ii) "Questões sobre o projeto, coleta de dados e análise do dados"e (iii) "Questões sobre o resultado do estudo".

- Dieste e Juristo (2011) desenvolveram uma escala de qualidade para determinar a qualidade de experimentos, contendo dez questões baseadas nas cinco dimensões de Kitchenham et al. (2007), sendo: contexto experimental, projeto experimental, análise, interpretação dos resultados e apresentação dos resultados. As respostas de cada questão são "sim"ou "não".

2.5 Ontologia

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 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.

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.

2.6 Trabalhos Relacionados

Durante uma revisão não sistemática (*ad hoc*) da literatura foram encontrados alguns estudos que propuseram abordagens para representar formalmente dados sobre experimentos

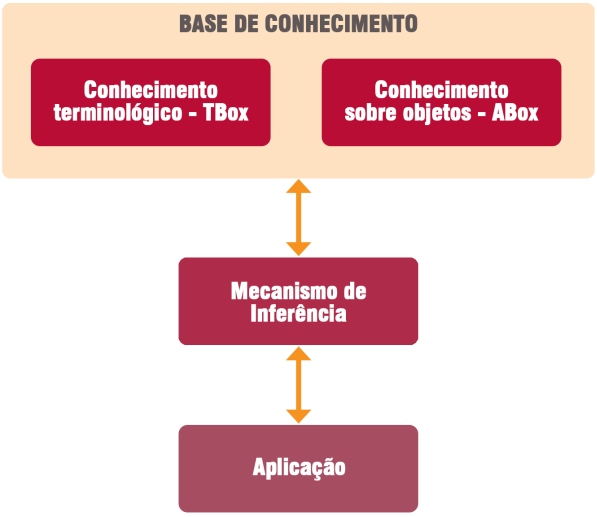

Figura 2.5: Estrutura de uma Ontologia. Autor

em ES. A revisão da literatura mostrou que a maioria dos estudos focou e representar todo o domínio do ES, ou experimentos em geral, esse fator é um agravante para elaboração dos modelos, pois, à quantidade de detalhes pode variar significativamente. Estes trabalhos estão descritos a seguir. A contribuição deles é discutida ao final desta sessão.

Durante nossa revisão de literatura encontramos: (Garcia et al., 2008) Ontologia para Experimentos Controlados em Engenharia de Software, (Scatalon et al., 2011) Empacotando Experimentos Controlados Usando uma Abordagem Evolutiva Baseada em Ontologia (S), (da Cruz et al., 2012) Uma Ontologia Fundamental para Apoiar Experimentos Científicos, (Blondet et al., 2016) ODE: uma Ontologia para Projeto Numérico de Experimentos, (Soldatova e King, 2006) Uma Ontologia de Experimentos Científicos, (Gelernter e Jha, 2016) Desafios na Avaliação de Ontologia, (Cruzes et al., 2007) Extraindo Informações da Engenharia de Software Experimental papéis. No entanto, todos esses trabalhos não tratam experimentos de LPS.

O trabalho de Garcia et al. (Garcia et al., 2008), Poveda-Villalón et al. (Scatalon et al., 2011) e Cruz et al. (da Cruz et al., 2012) se destaca em nosso contexto por propor e modelar ontologias específicas para experimentos em engenharia de software. O trabalho de (Garcia et al., 2008) propõe, através 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 auxiliares de pesquisadores e revisores, além de propor meta-análises, conduzir e avaliar experimentos controlados. O trabalho de Poveda-Villalón et al. (Scatalon et al., 2011) é uma evolução do trabalho de Garcia et al. (Garcia et al., 2008), mas focado na evolução desta ontologia

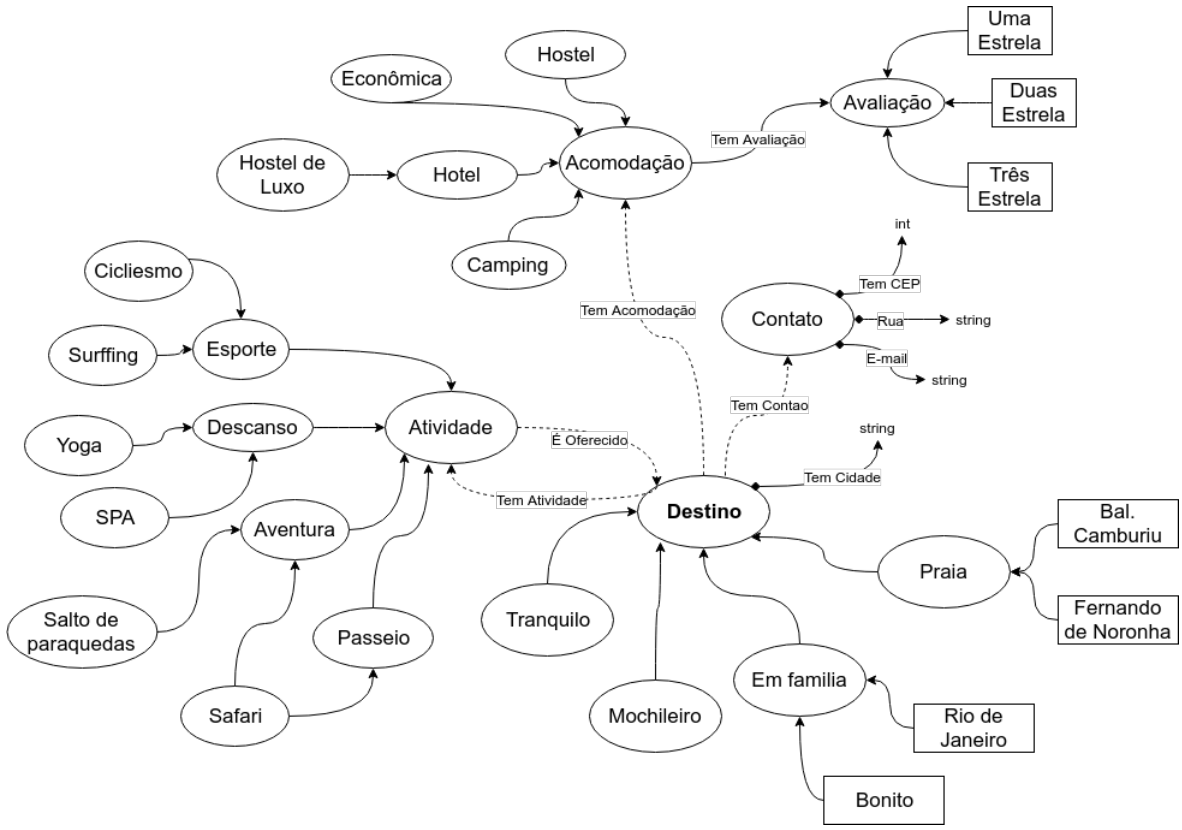

Figura 2.6: Exemplo de uma Ontologia para o domínio: Destino de Viagem. Autor

proposta. O trabalho de Cruz et al. (da Cruz et al., 2012) apresenta uma ontologia chamada OVO (Open onVence 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. (Blondet et al., 2016) e Soldatova e King (Soldatova e King, 2006) tratam ontologias no contexto geral de experimentos. O trabalho de Blondet et al. (Blondet et al., 2016) traz uma proposta de ontologia para DoE (Designs of Experiments) para apoiar as decisões de processo sobre o DoE. O trabalho de Soldatova e King (Soldatova e King, 2006) propõe a ontologia da EXPO que é uma mediana 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 (Gelernter e Jha, 2016) dá uma visão geral sobre os desafios de avaliar uma ontologia, mas não trata da ontologia para experimentos.

Finalmente, o trabalho de Cruzes et al. (Cruzes et al., 2007) trata de uma técnica para extrair meta-informação de experimentos em engenharia de software, entendemos que este assunto está relacionado a este artigo, pois estaremos gerando através da ontologia proposta diversos 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 trablahos de (Garcia et al., 2008) e (Scatalon et al., 2011). Desse modo a modelagem de otologia de para este trabalho será aplicado para experimento em LPS, de tal modo que, será possível fazer inferência de LPS na ontologia e extrair informações. Assim tais trabalhos relacionados são fundamentais para realização deste trabalho de mestrado.

Até o momento não há trabalhos de sistema de recomendações para voltados a experimentos em ES, e nem em LPS. Outros trabalhos relacionados mais próximos já foram apresentado na Seção 2.4 e não são objetos de evolução de experimento gerais em ES.

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 /citePohl 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. Nessa temos que a ontologia é uma modelagem formal de dados 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 SMartOntology.

3

Uma Ontologia para Experimentos em LPS - OntoExper-LPS

3.1 Considerações Iniciais

Este capítulo apresenta a elaboração da proposta de ontologia OntoExper-LPS, sua concepção, a construção do projeto, exemplos de aplicação, uma breve avaliação empírica e considerações finais. Dessa forma a estrutura deste capítulo fica, na Seção 3.2 apresenta o processo de concepção do modelo seguindo 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; na Seção 3.3 foi desenvolvido o projeto da OntoExper-LPS para se obter modelo ontológico, foi utilizado como tecnologia base OWL (*Ontology Web Language*) juntamente com a Ferramenta Protégé, o resultado final foi um artefato do tipo OWL contendo toda modelagem de classes, subclasses, propriedades de objeto e propriedades dos dados, além da modelagem, foi criado nesta seção, um programa automatizado para povoar a ontologia utilizando os metadados levantados pelo GReater, este programa conta com a utilização da linguagem de programação Python e a biblioteca OwlReady2; na Seção 3.4 foi elaborado uma consulta SPARQL para demonstrar como interagir com a ontologia e uma breve explicação deste procedimento; na Seção 3.5 foi realizado uma avaliação empírica do modelo, por meio da ferramenta OOPS! que avalia pontos de falha do modelo, nesta seção discutimos como estes ponto de falham impacta no modelo proposto por este trabalho; e na Seção 2.7 são apresentadas as Considerações finais deste capítulo..

3.2 Concepção

Com base na tipologia proposta por (Almeida e Bax, 2003) definimos o tipo da ontologia proposto neste trabalho, descrito na Tabela 3.2. Todas as tipologias para ontologias podem ser encontradas no Apêndice A

Ou seja, podemos tipificar o modelo proposto 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 tanto uma ontologia para modelagem de conhecimento quanto para aplicação de domínio.

Também foi aplicado o seguinte processo de elaboração da ontologia, seguindo os processos 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;
- Elaboração de sentenças para restringir inferências de conhecimento baseadas na estrutura.

Tabela 3.1: Tipagem da ontologia proposta

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.
Quanto ao grau de formalismo Uschold; Gruninger (1996)	Ontologias semi informais	Expressa em linguagem natural de forma restrita e estruturada.
Quanto à aplicação Jasper; Uschold (1999)	Ontologias como especificação	Cria-se uma ontologia para um domínio, a qual é usada para documentação e manutenção no desenvolvimento de softwares.
Quanto à estrutura Haav; Lubi (2001)	Ontologias de domínio	Descrevem o vocabulário relacionado a um domínio, como, por exemplo, medicina ou automóveis
Quanto ao conteúdo Van-Heijist; Schreiber; Wielinga (2002)	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.

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

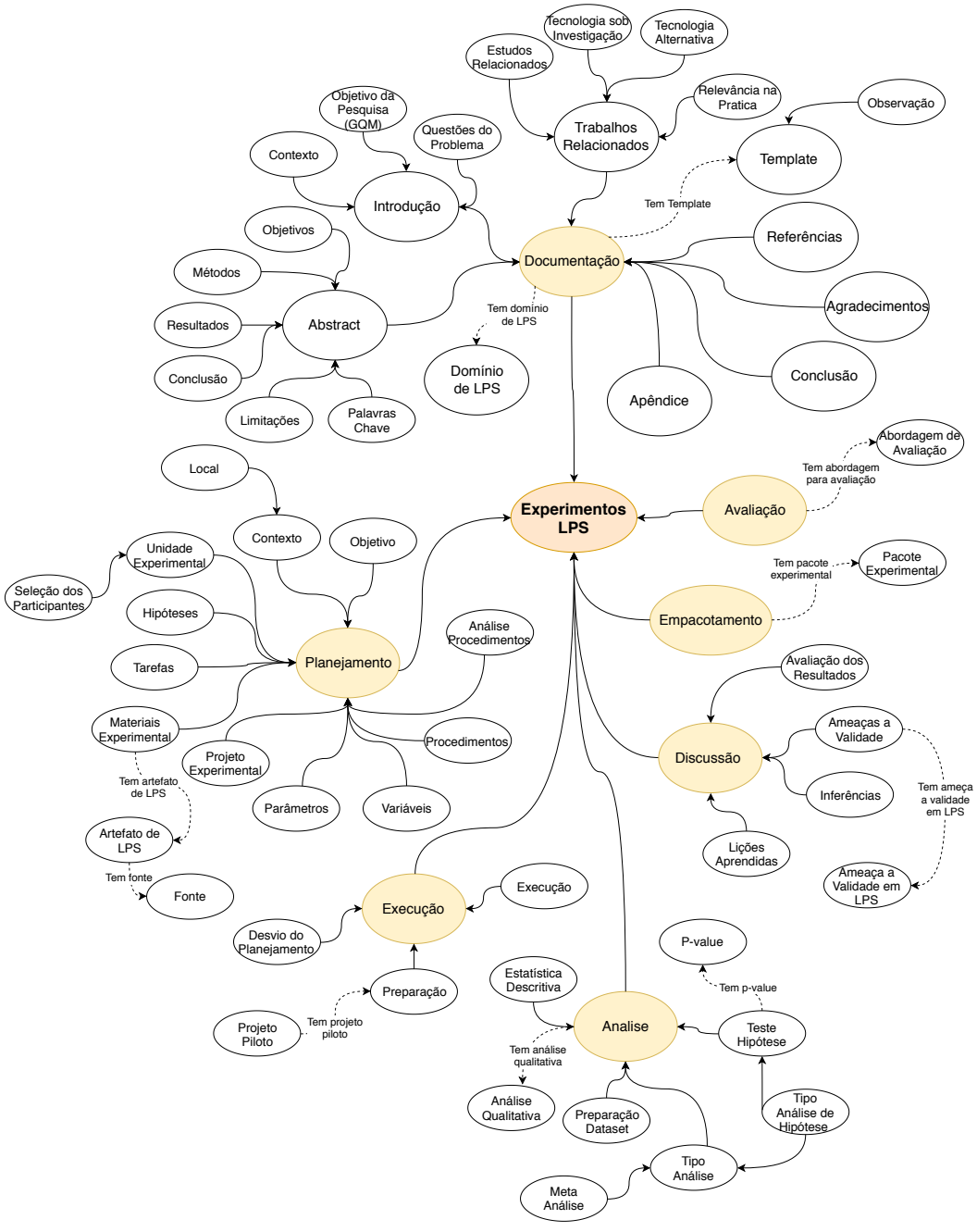

Figura 3.1: Grafo inicial da proposta de ontologia.

A Figura - 3.1 representa este modelo inicial contendo a definição principal dos termos para o experimento, Experimento SPL, Documentação, Template, Avaliação, Discussão,

Análise, Execução e Planejamento. Em seguida evoluímos para possíveis termos de mais baixa hierarquia, que estão 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 SPL;
      - Fonte
  - Projeto Experimental;
  - Parâmetros;
  - Variáveis;
  - Procedimentos;
  - Objetivo;
  - Hipóteses;
  - Tarefas;
  - Análises de procedimentos;
- Execução:
  - Preparação:
    - Projeto Piloto;
  - Execução; e
  - Desvio do Planejamento;
- Análise:
  - Tipo de Análise:
    - Meta Análise
    - Tipo de Análise de Hipótese;
  - Teste de Hipótese:
    - P-value
  - Estatística Descritiva;
  - Análise qualitativa;
  - Preparação do *Dataset*;
- Discussão:

  - Ameaças a Validade:
    - Ameaça a Validade em LPS.
  - Avaliação dos Resultados;
  - Inferências;
  - Lições Aprendidas;

- Empacotamento:
  - Pacote Experimental.

- Avaliação:
  - Abordagem de Avaliação.

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 usa-lo como guia de informações e metadados de ESE em LPS.

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).

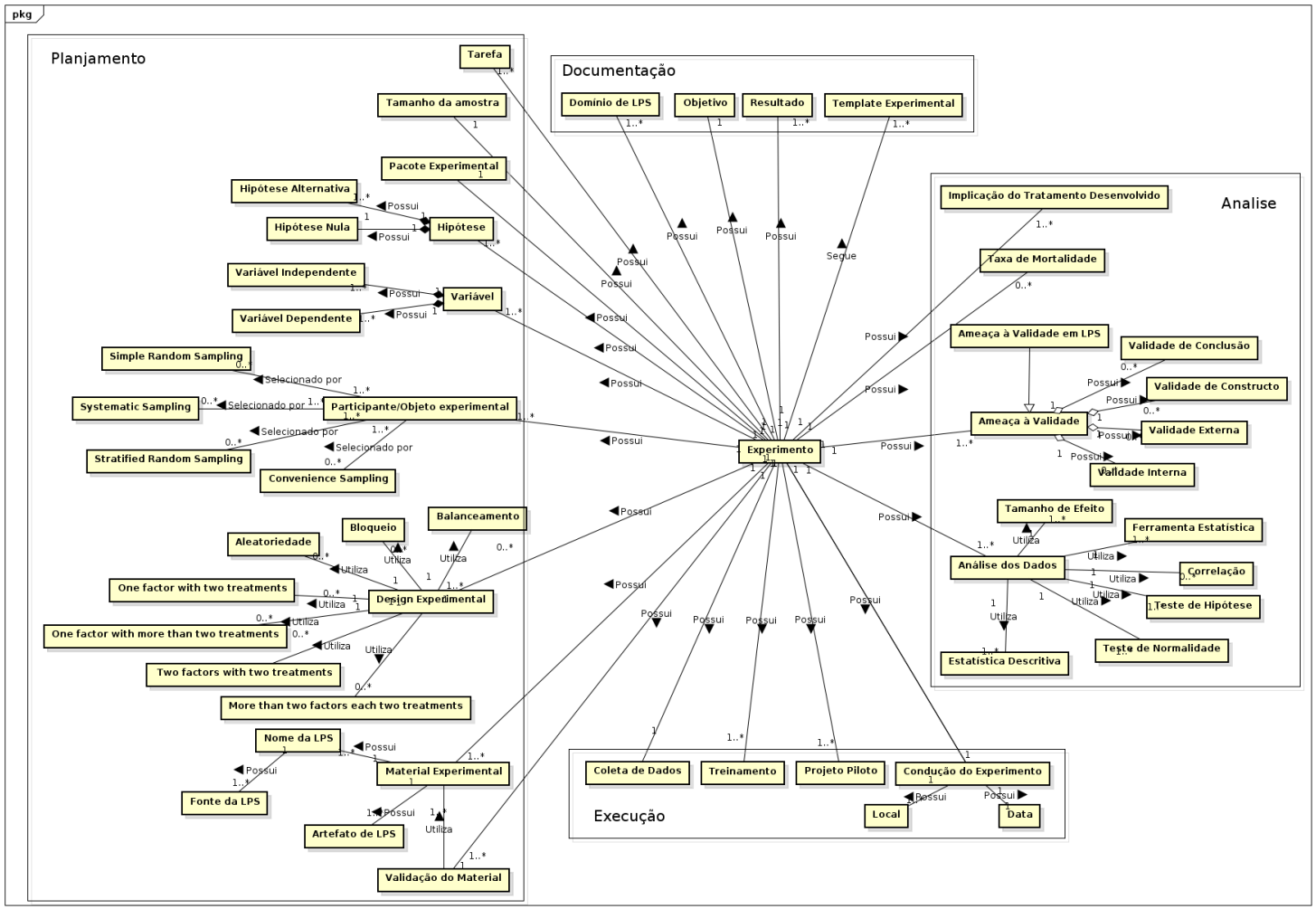

Figura 3.2: Modelo Conceitual Clusterizado.

A Figura - 3.2, apresenta uma modificação do modelo conceitual original, elaborada no trabalho de (Furtado, 2018), onde foi aplicado a clusterização separando cada cluster para cada fase do template experimental de Wohlin. Esta clusterização foi necessária para compreensão mais abstrata das relações entre os termos do domínio levantados 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 inicial, realizada com grafos. O objetivo é transformar cada termos do grafo em uma classe do conceito de Orientação a Objetos. Nessa representação, a modelagem da *TBox*, ou seja, a relação entre os termos (classes) e suas propriedades (atributos) ficou mais clara. Essa forma de representação destacou a relação principal quando definimos a composição da classe Experiment e ExperimentSPL em quase todas as outras subclasses. Nesta representação também ficou explícita a tipificação das propriedades, para posteriormente executar a inserção da *ABox* na modelagem.

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

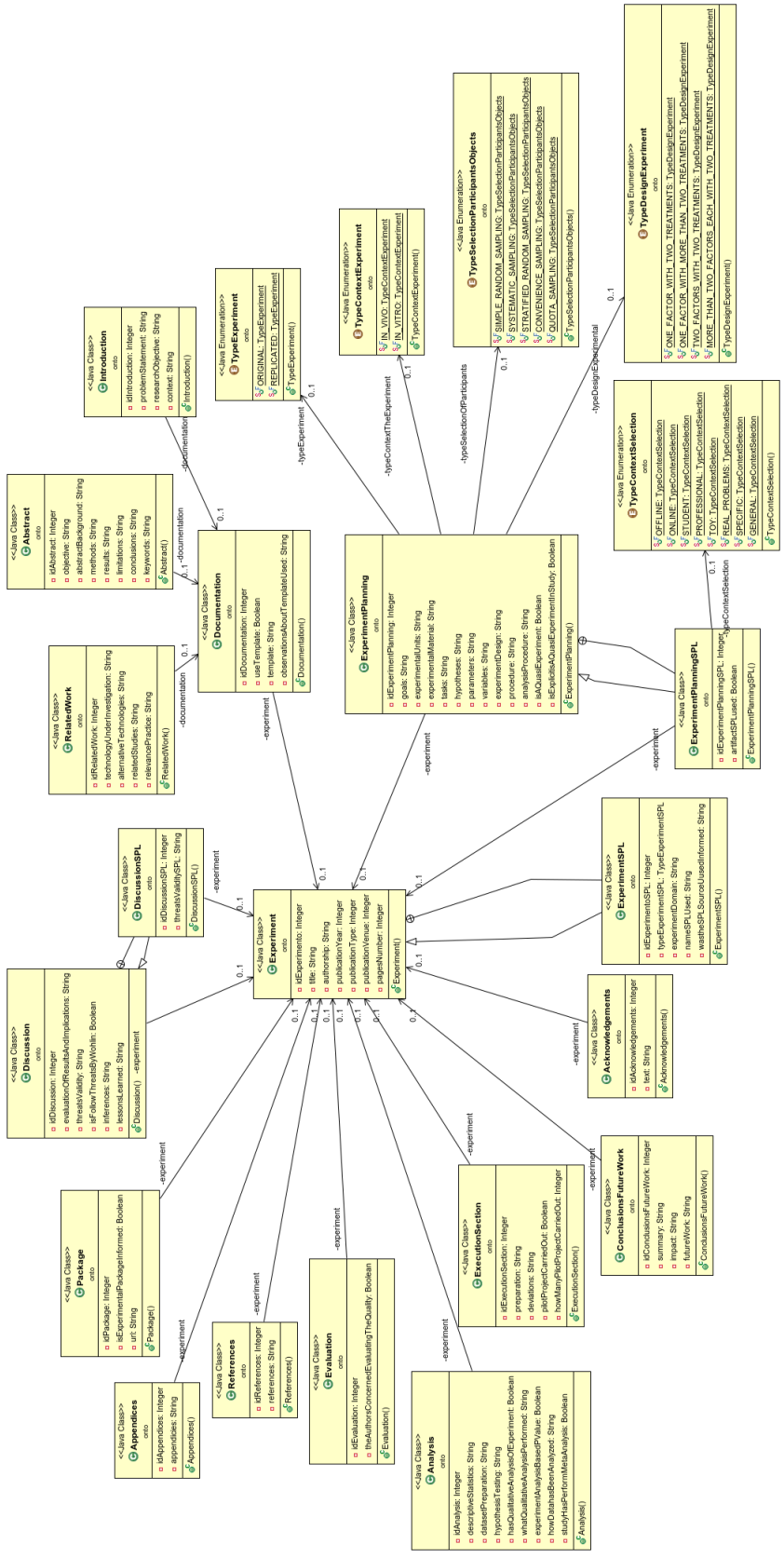

Figura 3.3: Diagrama de Classe para proposta da modelagem da ontologia.

Posteriormente, foi utilizada a ferramenta Protégé para construção oficial da ontologia, seguindo os padrões OWL.

3.3 Projeto

Definimos a criação da ontologia usando a tecnologia OWL, 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 Protégé 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 tanto por, desenvolvedores de sistema, como por especialistas em domínio, para criar bases de conhecimento, permitindo representar facilmente o conhecimento de uma área. Este editor é capaz de tratar classes, com sua definição e exemplos, simultaneamente propriedades de objetos e de dados (Musen, 2015).

Fazendo uma analogia ao diagrama de classe, no Protégé, as classes, os atributos de classes e seus relacionamentos estão em um contexto de entidades. As classes e hierarquias são definidos na aba **Classes**, os relacionamentos são definidos na aba **Propriedades de Objetos** e os atributos são definidos na aba **Propriedade dos Dados**.

Definimos nossas entidades com base do diagrama de classe construído na fase de concepção, com a seguinte estrutura (i) definição de classes (ii) definição das propriedades de objetos, (iii) definição das propriedades dos dados.

Definição de classes

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

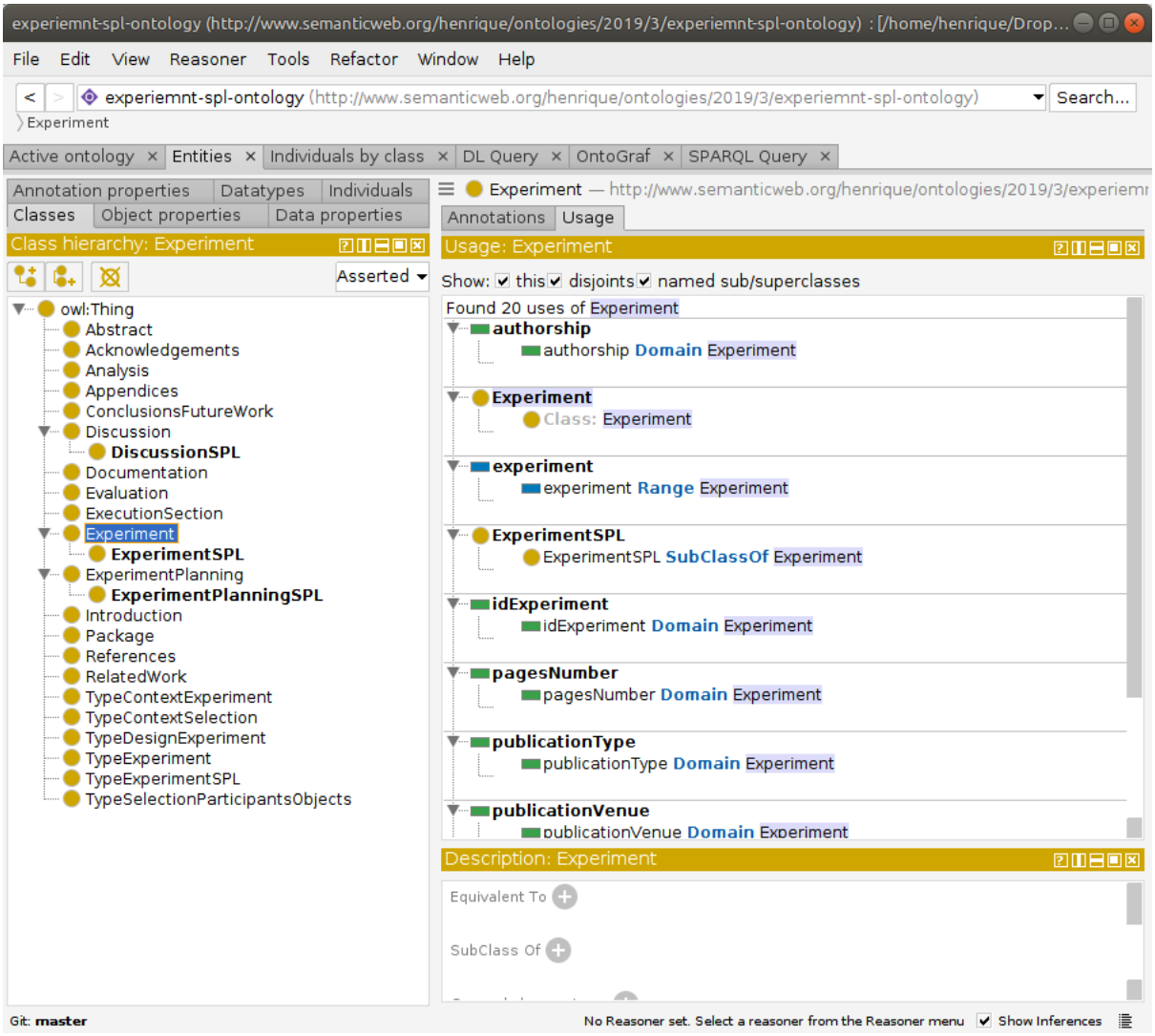

Figura 3.4: Definição da classe Experiment no Protégé.

Definição das propriedades de objetos

No Protégé, definimos uma propriedade de objeto raiz chamada *topObjectProperty* na qual todas as outras propriedades são filhas dela. A Figura - 3.5 apresenta a definição de propriedade de objeto para propriedade *documentation*, neste momento a definição

é composta pelo nome da propriedade e seus relacionamentos com classes, Domínio *(Domains)* e Alcance *(Ranges)*.

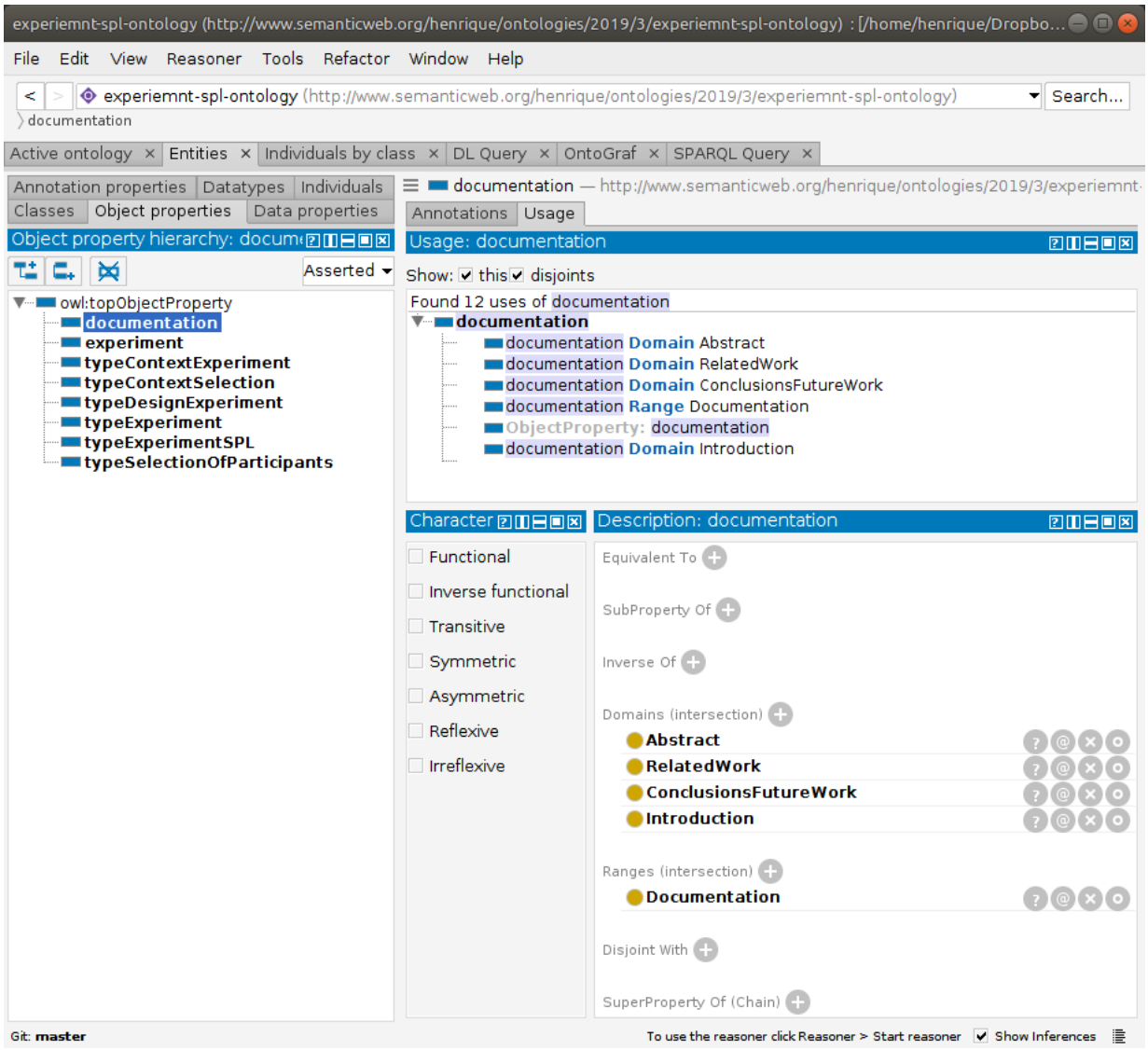

Figura 3.5: Definição da propriedade de objeto documentation no Protégé.

Definição das propriedades dos dados

No Protégé, definimos uma propriedade de dados raiz chamada *topDataProperty* na qual todas as outras propriedades são filhas dela. Para cada Propriedade de dado definimos 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 para chamada *nameSPLUsed*, neste momento a definição é composta pelo nome da propriedade e seus relacionamentos, Domínio (uma classe) e Alcance (uma tipagem).

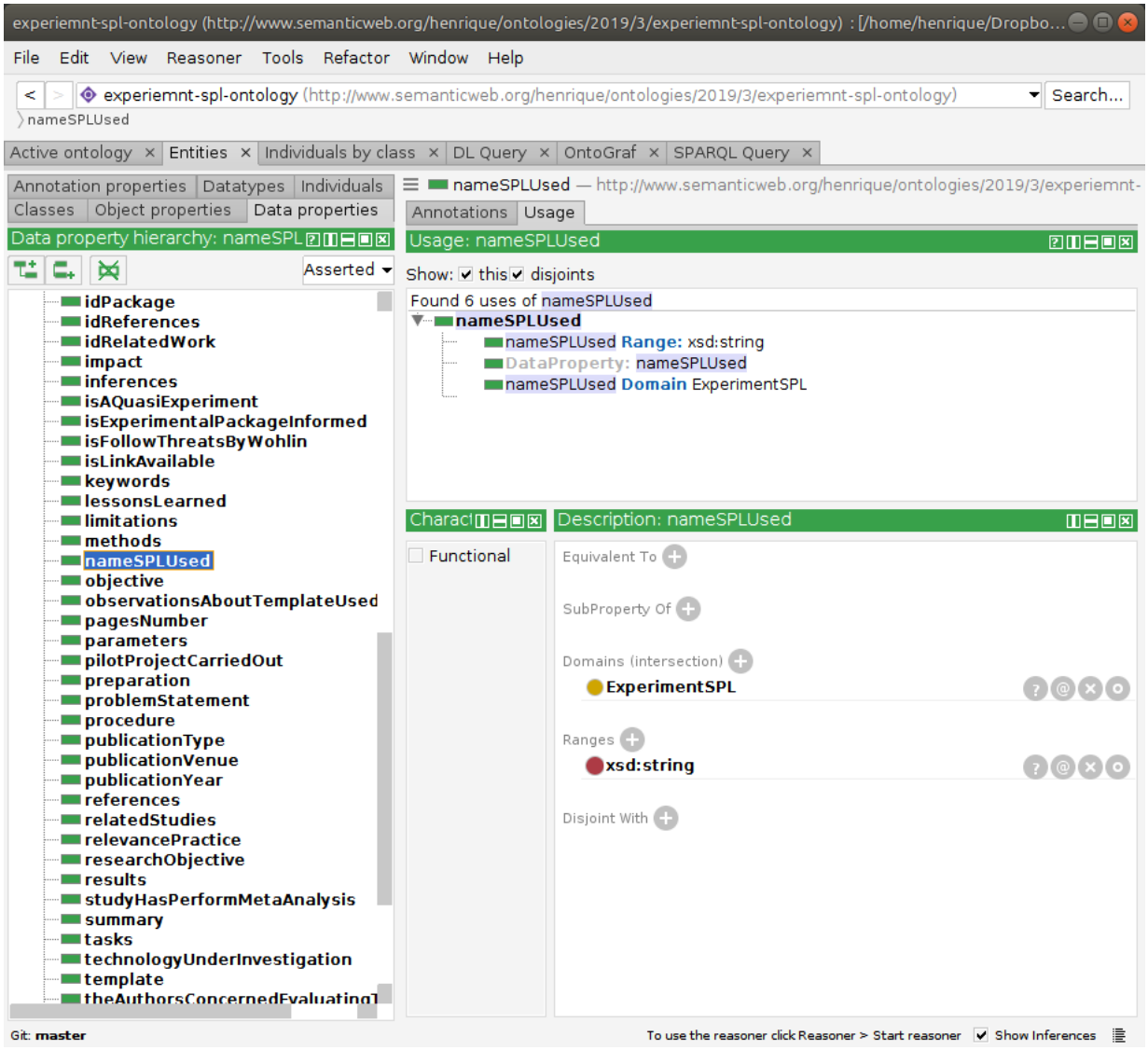

Figura 3.6: Definição da propriedade de dado nameSPLUsed no Protégé.

Artefato gerado pelo Protégé

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

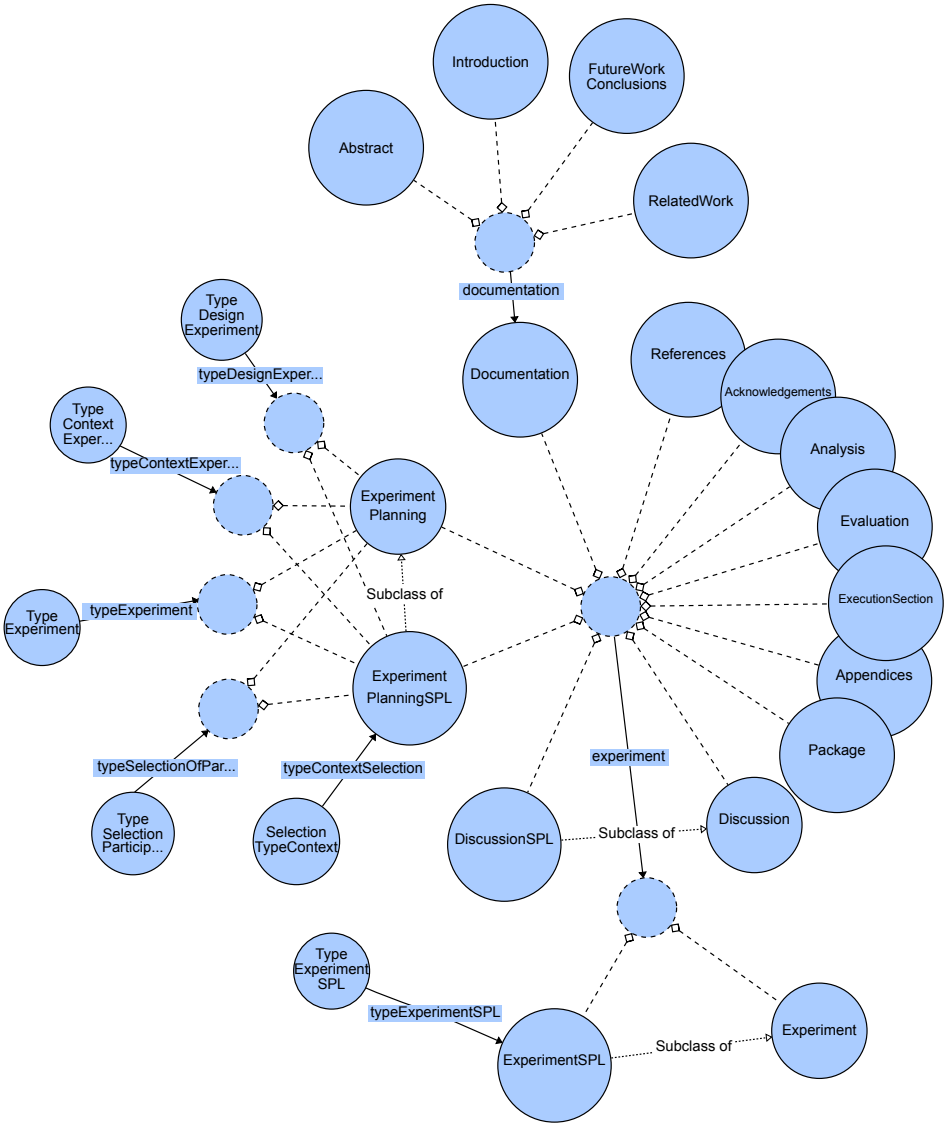

Figura 3.7: Grafo da modelagem da ontologia - gerado pelo WbVOWL.

3.3.2 Povoamento com Python

A próxima etapa, após a modelagem da ontologia, foi inserir os indivíduos na mesma, ou seja, popular a ontologia para realização de inferências futuras. Apesar do Protégé ter capacidade para realizar essa operação, optamos por utilizar um script para realizar tal tarefa, visto que, no Protégé o processo de inserir indivíduos na ontologia é realizado de modo manual, por meio do menu *Individuals by Class*, onde é preciso selecionar propriedade por propriedade para cada indivíduo. Sabendo que para cada indivíduo temos 84 propriedades de dados mais 8 propriedade de objetos, somando um total de 92 relacionamentos para cada indivíduo. Portanto, sabendo que temos aproximadamente

duzentos indivíduos a serem inseridos em uma carga inicial, seriam aproximadamente mais de 184.000 iterações manuais no Protégé. Este cálculo está descrito na Equação 3.1.

$$n^{\circ}\, iteracoes\, manuais = n^{\circ}\, relacionamentos * n^{\circ}\, individuos \tag{3.1}$$

Dado essa estimativa de operações manuais optamos pela utilização de uma ferramenta script, a fim de facilitar e automatizar a inserção dos indivíduos e posteriormente a inserção de novos indivíduos. Escolhemos a linguagem de programação Python por fornecer bibliotecas prontas, tanto para manipulação de dados em planilhas (Pandas), como para ontologia em arquivos .owl (OwlReady2), para tal finalidade.

Script

Para o ambiente de desenvolvimento do script, utilizamos 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)*, onde extraímos os dados originais da planilha, desenvolvida no trabalho de revisão sistemática sobre experimentos em LPS, e manipulamos estes dados para inserir na ontologia, seguindo a modelagem inicial construída no Protégé.

Para leitura dos dados da planilha usamos a biblioteca Pandas (McKinney, 2010), ela nos retorna um objeto do tipo *Dataframe*, no qual é a representação da própria planilha no ambiente de programação Python.

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

Com relação ao *Experiment* temos: *ID, Title, Authorship, Publication year, Publication type, Publication venue, Pages number, dados de Experimentos em SPL temos: 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* temos: *Does it use template? (Y/N)?, If yes, what template?, Observations about the template used. Para o Abstract temos: 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. Para o Introduction Section temos: 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)?). Para o Related Work Section temos: 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?). Para o Conclusions and Future Work Section temos: 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* temos: *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, Context Selection (Off-line vs. on-line, Student vs. professional, Toy vs. real problems, Specific vs. general).*

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

Com relação a *Package* temos: *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-LPS, 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. Foi desenvolvido tres funções de conversão, (i) convertToBoolean, (ii) convertStringEmpty e (iii) convertToNumber. Este caso ocorreu para os seguintes dados:

- *Does it use template? (Y/N)?* para useTemplate aplicando o método convertToBoolean;
- *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 convertToBoolean;
- *Did the study perform meta-analysis? (Y/N)* para hasMetaAnalysis aplicando o método convertToBoolean;
- _informedSPL para informedSPL aplicando o método convertToBoolean;
- _hasQuasiExperiment para hasAQuasiExperiment aplicando o método convertToBoolean;
- _hasPilot para hasPilot aplicando o método convertToBoolean;
- _howManyPilot para howManyPilot aplicando o método convertToNumber;
- _hasThreatsValidityByWolin para hasThreatsValidityByWolin aplicando o método convertToBoolean;
- _hasPackage para hasExperimentalPackage aplicando o método convertToBoolean;

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

Quarta operação: padronização de dados constantes e faltantes. Foi preciso padronizar os dados 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] sendoo 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 temos um objeto na programação que representa o modelo, onde a 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.

Foi criado dois laços para percorre 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 sequencia 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 instância 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 temos o objeto que representa a ontologia populado com com os indivíduos, e por meio da biblioteca Owlready2 geramos 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 conferimos a quantidade de linhas da planilha com a quantidade de indivíduos inserido na ontologia, pode ser visto em 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 usados pelos indivíduos da ontologia. Foi utilizado a tecnologia SPARQL, para executar a mesma.

SPARQL é uma linguagem de consulta e um protocolo para acesso a RDF elaborado pelo W3C RDF Data Access Working Group. 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 /citeApache Jena

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 onde são definidos quais prefixos RDF será usada, bem como a OntoExper-LPS. A primeira linha realiza contagem do termo *template*. O bloco *WHERE* aplica a restrição da consulta, nesta restrição só buscamos 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.

Este exemplo percorre uma classes (*Documentation*) das 24 classes existentes no modelo, uma propriedades de dados (*template*) de 87 propriedades de dados do modelo proposto. O cálculo da descrito em 3.2, estima que neste exemplo estamos usando 0,0004% da capacidade de resposta que o modelo possa responder.

$$\%\,uso\,da\,ontologia = \%\,classe * \%\,prop\,de\,objeto * \%\,prop\,de\,dados \tag{3.2}$$

Este simples exemplo de consulta apresenta como podemos criar mecanismos de inferência no modelo de ontologia proposto neste trabalho. Dessa forma, foi possível validar empiricamente que é possível extrair informações sobre experimentos em SPL usando a OntoExper-LPS.

3.5 Avaliação Empírica

Para esta avaliação empírica da proposta de ontologia, seguimos duas estratégias, (i) uma para avaliar as possíveis armadilhas e (ii) outra para levantar falhas no modelo.

Foi utilizada a ferramenta OOPS! para gerar avaliação do modelo de ontologia proposto. A ferramenta ajuda a detectar algumas das armadilhas mais comuns que aparecem ao desenvolver ontologias [referência]. Segue abaixo os pontos de falha que a ferramenta avalia.

- P01. Criando elementos polissêmicos;
- 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 a qualidade da ontologia. No entanto, nem todas as armadilhas identificadas devem ser interpretadas como erro, mas sim como sugestões que devem ser revisadas manualmente em alguns casos.

Esta avaliação pode ajudar a descobrir erros 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:

- ***Critical:*** 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;
- ***Minor:*** Não é realmente um problema, mas ao consertá-lo, tornaremos a ontologia mais agradável.

A Tabela - 3.2 apresenta o resumo do resultado ao rodar nosso proposta de modelo de ontologia na ferramenta OOPS!.

Tabela 3.2: Resumo dos resultado da avaliação executada por OOPS!

Pitfall	**Description**	**Critical Level**
P08	Anotações ausentes em 119 casos	*Minor*
P10	Falta de disjunção na ontologia *	*Important*
P13	Relações inversas não declaradas explicitamente em 8 casos	*Minor*
P19	Definindo vários domínios ou intervalos nas propriedades em 6 casos	*Critical*
P41	Nenhuma licença declarada na ontologia *	*Important*

*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 [referência 5 do OOPS].

No caso P10, a ontologia não possui axiomas desarticulados entre classes ou entre propriedades que devem ser definidas como disjuntas. Esta armadilha está relacionada com as orientações fornecidas em [6], [2] e [7].

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: 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 [7]. 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. Estes pontos de melhorias serão tratados na seção 4.10.

3.6 Considerações Finais

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

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 OntoExper-LPS. 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, temos uma breve avaliação empírica do modelo proposto, onde foi aplicado um ferramenta (OOPS!) que avalia 41 pontos de possíveis pontos falhas na ontologia, os pontos de falhas relatado pela ferramenta, serão tratados na seção 4.10.

No próximo capítulo, será apresentado uma avaliação da OntoExer-LPS como um estudo qualitativo.

4

Avaliação da OntoExer-LPS: Um estudo Quantitativo

4.1 Considerações Iniciais

Este capítulo apresenta um estudo quantitativo realizado para avaliar a OntoExer-LPS proposta no Capítulo 3, com base em um questionário contendo onze questões, oito com critérios de qualidade no estilo escala Likert /citeLikert, uma avaliação de frequência da escala Likert, teste estatístico e sua correlação entre os oito critérios.

Os oito critérios de qualidade foram identificados e definidos na Parte I da tese *Ontology Evaluation* de /citeDenny Vrandecic, são eles:

- **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? definidos, sejam outras fontes de dados, conceituações ou mesmo realidade. /cite(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:** agrega 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.

4.2 Escala Likert

A tese de /citeDenny Vrandecic não expoem uma forma clara para mensurar os oito critérios. Neste ponto foi definido que uma escala Likert poderia auxiliar. Segundo /citeLikert 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.

4.3 Definição do Estudo

O propósito principal deste estudo foi avaliar a OntoExer-LPS proposta no Capítulo 3. Baseado no modelo Goal-Question-Metric (GQM) (Basili e Rombach, 1988), este estudo tem por objetivo:

Avaliar a OntoExer-LPS proposta
Com o propósito de definir valores quantitativo
Referente os critérios de avaliação
Do ponto de vista de especialistas de ES, LPS e Ontologia
No contexto de pesquisadores das seguintes instituições: UEL, ICMC-USP, PUC-PR, UEM, UTFPR, USP, SIDIA, PUC-RS e Unipar.

4.4 Planejamento do Estudo

O planejamento do estudo é composto das seguintes etapas:

- **Projeto Piloto:** com a intenção de avaliar a instrumentação do estudo, um projeto piloto foi conduzido em Outubro de 2019, com um mestre e Doutor da Universidade Estadual de Maringá (UEM). Os dados obtidos foram descartados, mas, as considerações sobre erros e melhorias nos questionários foram consideradas.
- **Treinamento:** por causa do nível de conhecimento dos participantes não foi necessária a realização desta etapa.
- **Especialistas:** os especialistas convidados foram pesquisadores da área de ES, LPS e Ontologia. Dos participantes, X são pós-doutor (X%), X são doutores (X%), X são mestres (X%), X é mestrando (X%).

- **Instrumentação:** todos os especialistas receberam os seguintes documentos: i) um e-mail contendo um breve descritivo da avaliação, com os link para ferramentas e tecnologias necessárias; (ii) uma cópia do Instrumento de Avaliação Quantitativo caracterizado por um questionário - Google Forms; e (iii) uma cópia dos metadados da ontologia, os metadados são compostos por: um arquivo do tipo OWL do modelo da OntoExer-LPS, um arquivo do tipo OWL da OntoExer-LPS povoada com mais de 170 indivíduos, um arquivo do tipo planilha do Excel contendo os dados originais dos experimentos, uma cópia do artigo *An Ontology for Software Product Line Experiments* não publicado até o momento da escrita desta dissertação, uma cópia do capítulo 3 desta dissertação, uma cópia da tese de *Ontology Evaluation* de /cite-Denny Vrandecic, uma cópia do resumo da tese, uma cópia da Figura(XX)/diagrama de classe, uma cópia da Figura(XX)/modelo conceitual clusterizado, uma cópia da Figura(XX)/grafo inicial da ontologia, uma cópia da Figura(XX)/grafo gerado pela OWLweb.

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

4.5 Execução do Estudo

Esta seção apresenta as etapas seguidas durante a execução deste estudo.

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 4.4);
2. o especialista faz um estudo prévio dos metadados, esclarece possíveis dúvidas;
3. o especialista faz a leitura e preenche o formulário de Avaliação Quantitativo - 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.

4.6 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 estatístico aplicado aos resultados;
- apresentação e análise da correlação entre os critérios de qualidade; e

4.6.1 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 anos, destacamos 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.

Tabela 4.1: Perfil dos Participantes que Realizaram o Estudo Quantitativo

Especialista (E)	**Nível de Formação**	**Experiência na Área (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	Ensino Superior - Completo	3
E14	Mestrado	5
E15	Doutorado	12
E16	Mestrado	6
E17	Doutorado	6

4.6.2 Frequência Likert

Os dados coletados dos especialistas foram analisados por meio de uma análise exploratória de dados usando Jupyter-Lab, como linguagem de programação Python juntamente com as bibliotecas Pandas, Matplotib, SeaBorn, Scipy.

Foram oito critérios de qualidades avaliados pelos especialistas no estilo Likert. A Tabela - 4.2 apresenta os a frequência das respostas dada pelos especialistas de cada critério em relação a 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
- **%CT:** Concordo Totalmente, valor: 5.

Tabela 4.2: Frequência das Respostas Likert

	%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

A moda representa o valor mais frequente respondido pelos especialista. Diante disso, foi possível identificar que o critério **Precisão** teve a moda 5, com uma frequência de 52,94% para CT; o critério **Adaptabilidade** teve a moda 5 também, com uma frequência de 64,71% para CT; o critério **Clareza** teve a moda 4, porém com uma frequência iguais (47,06%) para CT e CP; o critério **Completude** teve a moda 4, com uma frequência de 35,29% para CP; o critério **Eficiência Computacional** teve a moda 3, com uma frequência de 47,06% para N; o critério **Concisão** teve a moda 5, com uma frequência de

58,82% para CT; o critério **Consistência** teve a moda 5, com uma frequência de 64,71% para CT; e o critério **Capacidade Organizacional** teve a moda 5, com uma frequência de 52,94% para CT.

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 desempenho 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 desempenho 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 N neste critério.

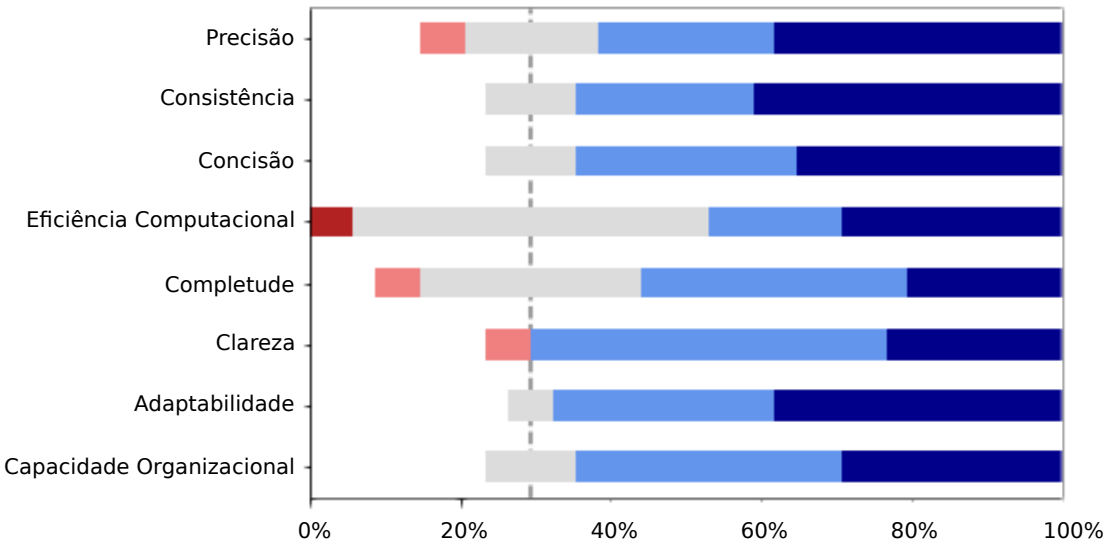

Figura 4.1: Frequência das Respostas em %

4.6.3 Teste de Normalidade

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

Tabela 4.3: Somatório das Resposta dos Especialistas

Especialista (E)	Somatório
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

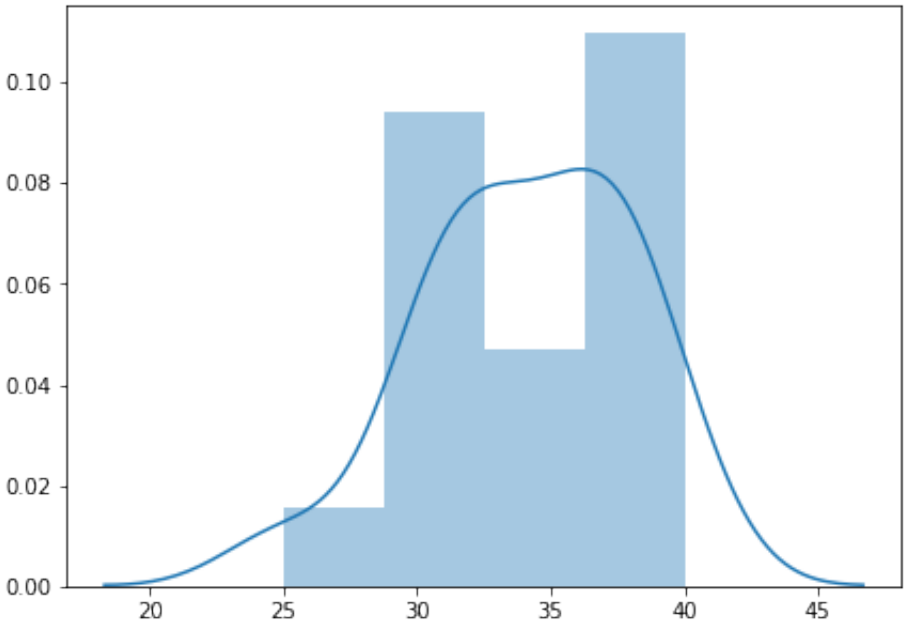

Figura 4.2: Histograma da Somatória

Colocando estes valores no gráfico de histograma apresentado na Figura - 4.2. O próximo passo foi executar o teste de normalidade, aqui foi aplicado o teste de Shapiro-Wilk.

O teste Shapiro-Wilk testa a hipótese nula de que os dados foram extraídos de uma distribuição normal, considerando que *p-value* > 0,05.

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 **normal**.

4.6.4 Teste Estatístico

Sendo a distribuição da VD uma normal, podemos aplicar o teste estatístico de Mann-Whitney. Para aplicar este teste foi preciso elaborar hipóteses para este questionário. Segundo /cite ao aplicar teste, quando o *p-value:* > 0,05 poder rejeitar H0 e quando p-value < 0,05 não rejeita H0.

- **Hipótese 1:** Existe diferença no resultado dado o Nível de Formação?
- **Hipótese 2:** Existe diferença no resultado dado a A Experiência na Área?

O resultado do teste de Mann-Whitney para Hipótese 1, foi u = 0.000, p-value = 0.00000029, logo não rejeitamos H0 e podemos afirmar que, **existe diferenças nos resultado dado o Nível de Formação**.

O resultado do teste de Mann-Whitney para Hipótese 2, foi u = 0.000, p-value = 0.00000027, logo não rejeitamos H0 e podemos afirmar que, **existe sim diferenças nos resultado dado o Nível de Formação**. 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 5 anos de experiências.

4.6.5 Correlação dos Critérios

A Figura - 4.3 apresenta um mapa de calor indicando o grau de correção entre os critérios de qualidade. Foi possível observar os critérios com maior grau de correção foram **Completude** com **Precisão** com o valor de correlação mais significativo (0,74), **Capacidade Organizacional** com **Clareza** com o segundo valor de correlação mais significativo (0,73) e **Capacidade Organizacional** com **Precisão** com o terceiro maior valor de correlação mais significativo.

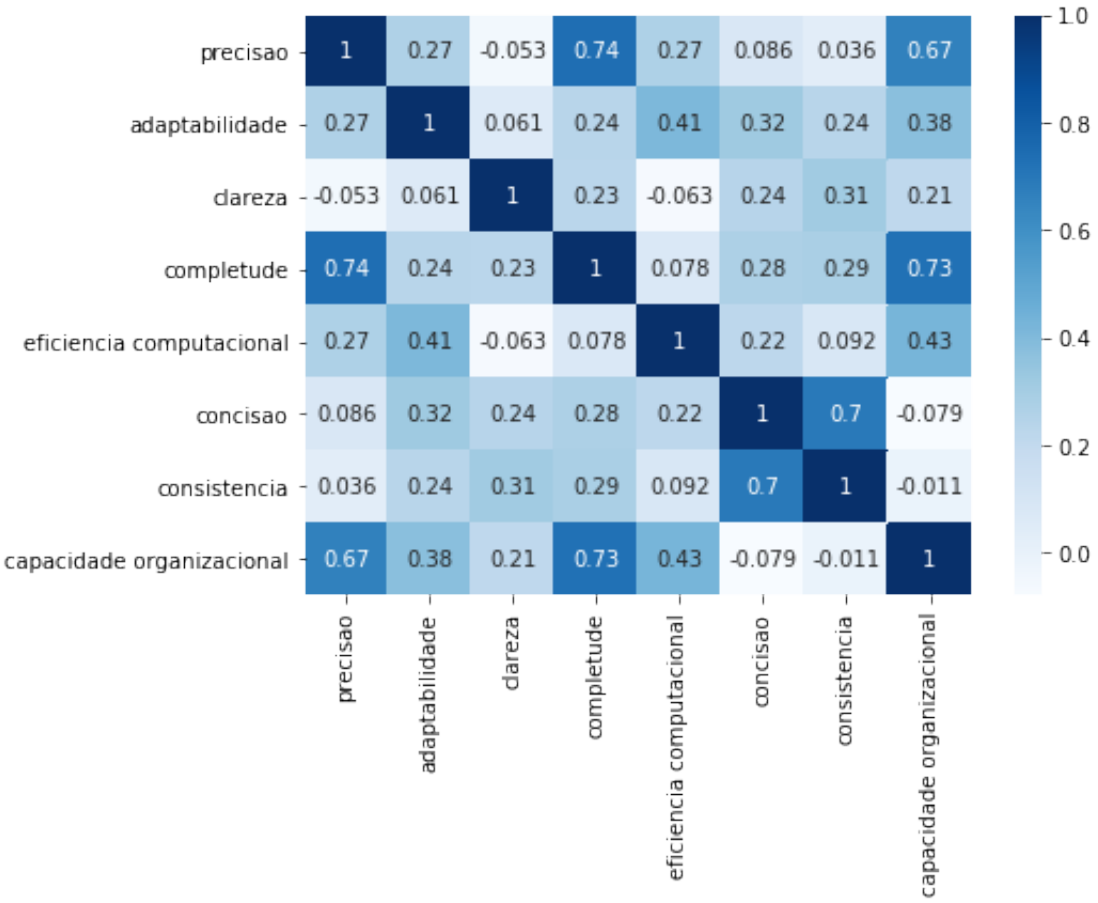

Figura 4.3: Correlação do s critérios

4.7 Discussão dos Resultados

A análise dos dados apresentada na seção anterior, 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 - 4.4

Feito essa análise é possível afirmar que a OntoExper-LPS 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 a eficiência computacional, ou seja, ela não possui um bom desempenho em sua execução (execução de inferências).

Tabela 4.4: Total de Respostas por Elemento da Escala Likert

	DT	**DP**	**N**	**CP**	**CT**
Total de respostas	1	3	23	41	68
%	0,74%	2,21%	16,91%	30,15%	50,00%

4.8 Ameaças à Validade

4.8.1 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 OntoExer-LPS 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 OntoExer-LPS (modelagem e povoados com indivíduos) são de difícil leitura pelos seres humanos, logo se faz necessária a utilização de ferramentas para exibição da OntoExer-LPS, 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:** 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.

4.8.2 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.

4.8.3 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.

4.8.4 Ameaças à Validade de Conclusão

A principal ameaça para este estudo está relacionada ao número de especialistas (??). 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.

4.9 Empacotamento e Compartilhamento

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

4.10 Propostas de Melhorias

4.10.1 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. Consegui navegar e me entender melhor 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 OntoExer-LPS 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 pequisa. 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 OntoExer-LPS precisa ser melhorada quanto a sua eficiência computacional. Os pontos de melhoria são, corrigir os termos *Keyworld* para *Keywords*, associar *SPL artifacts* a *SPL domain*, incluir os axiomas *Results*, *Analyze*, "Ameaças a Validade", *Subjects*e modificar o axioma *Package* para *Replication-Package*.

4.10.2 Melhorias do OOPS! - Armadilhas

Na Tabela - 3.2 foi relatado as armadilhas que a ferramenta encontrou na OntoExer-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 OntoExer-LPS.

4.11 Considerações Finais

Este capítulo apresentou a avaliação quantitativa sobre oito critérios de qualidade aplicados a OntoExer-LPS com especialistas em ES, LPS e Ontologia. Os resultados obtidos permitiram identificar a qualidade da OntoExer-LPS. Assim, os resultados dos dados analisados forneceu evidências que as OntoExer-LPS possui um grau de qualidade relevante.

Permitindo assim a evolução de novos estudos sobre a OntoExer-LPS, expandindo-a e melhorando seus pontos de baixa qualidade.

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

5

Um Sistema de Recomendação para Experimentos em LPS (Prova de Conceito) - RecSysExper-SPL

5.1 Considerações Iniciais

Este capítulo apresenta uma prova de conceito sobre a realização de inferência no modelo de ontologia proposto neste trabalho de mestrado, a OntoExper-LPS, por meio de uma implementação de um sistema de recomendação (SR). Dessa forma a estrutura deste capítulo fica, na Seção 5.2 apresenta um breve contexto sobre SR e sua origem, seus principais modelos de implementação e como os SR são visto pela ES; na Seção 5.3apresenta como as consultas iniciais na OntoExper-LPS levou a implementação do SR e como ele encaixa como motor de inferência na OntoExper-LPS, bem como sua interação com o usuários; na Seção 5.4 apresenta as quatro fase de construção SR, (i) a **Fase 1** apresenta a criação do projeto de sistema web usando Django Framework e a implementação de um prototipação da tela de iteração do usuário com SR, (ii) a **Fase 2** apresenta como a OntoExper-LPS é carregada no SR e como foi implementado os mecanismos de consultas na OntoExper-LPS, a **Fase 3** apresenta implementação da avaliação dos usuários no SR e como esta é armazenado no SR, esta avaliação é denominada *Ratings*, por fim a **Fase 4** apresenta com foi implementado do modelo de recomendação, na qual, foi usado o conceito de filtragem colaborativa aplicando cálculo da distância Euclidiana, bem como também apresenta como foi implementação a questão da

da partida fria do modelo de recomendação para novos usuários; na Seção 5.6 apresenta quais foram as tecnologias e ferramentas usadas para o desenvolvimento do SR; e na Seção 5.8 são apresentadas as Considerações finais deste capítulo.

5.2 Sistema de Recomendação

Os sistemas de recomendação são aplicativos de software que visam dar suporte para usuário na tomada de decisões ao interagir com grandes espaços de informação. Estes 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 usados em sistemas de recomendação, onde a recomendação de conteúdo é auxiliada pela colaboração de um grupo de pessoas, batizada como "filtragem colaborativa". Neste trabalho, iniciou o desafio de casar corretamente os dados recomendados com os usuários que o recebem, analisando o real relacionamento de interesse (Kwong et al., 1992; Resnick e Varian, 1997).

Podemos apresentar uma definição formal para sistema de recomendação da seguinte forma:

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 quão útil é um determinado item s para um determinado usuário c,* $\boldsymbol{u{:}C\ x\ S \rightarrow R}$*, onde R é um conjunto totalmente ordenado segundo a função utilidade. Então, para cada usuário* $\boldsymbol{c \in C}$*, procura-se um item* $\boldsymbol{s' \in S}$ *que maximiza a utilidade do usuário. Isto pode ser expressado pela equação 5.1:*

$$\forall c \in C, s'_c = argmax_{s \in Su(c,s)} \tag{5.1}$$

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*), e 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). 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).

Dado que este trabalho trata o sistema de recomendação como uma prova de conceito para realização de inferência sobre a OntoExper-LPS, utilizaremos a abordagem Filtragem Colaborativa, na próxima seção estará descrito o como este modelo de SR foi implementado e incorporado no projeto.

5.2.2 Sistema 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) comenta que, em um ambiente de desenvolvimento de software aplicando ES existe um *landscape* 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 as RSSEs ajudam a descobrir novas informações e (ii) traz familiaridade e reforço, pois as RSSEs suportam a confirmação do conhecimento existente. Finalmente, referenciar uma tarefa e um contexto específicos, 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).

RSSE compreende 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 - 5.1 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 encontrado uma revisão sistemática (trabalho da Maki, 2016), que aborda métodos e modelos de implementação de um RSSE apresentando vários aspectos de SR em ES,

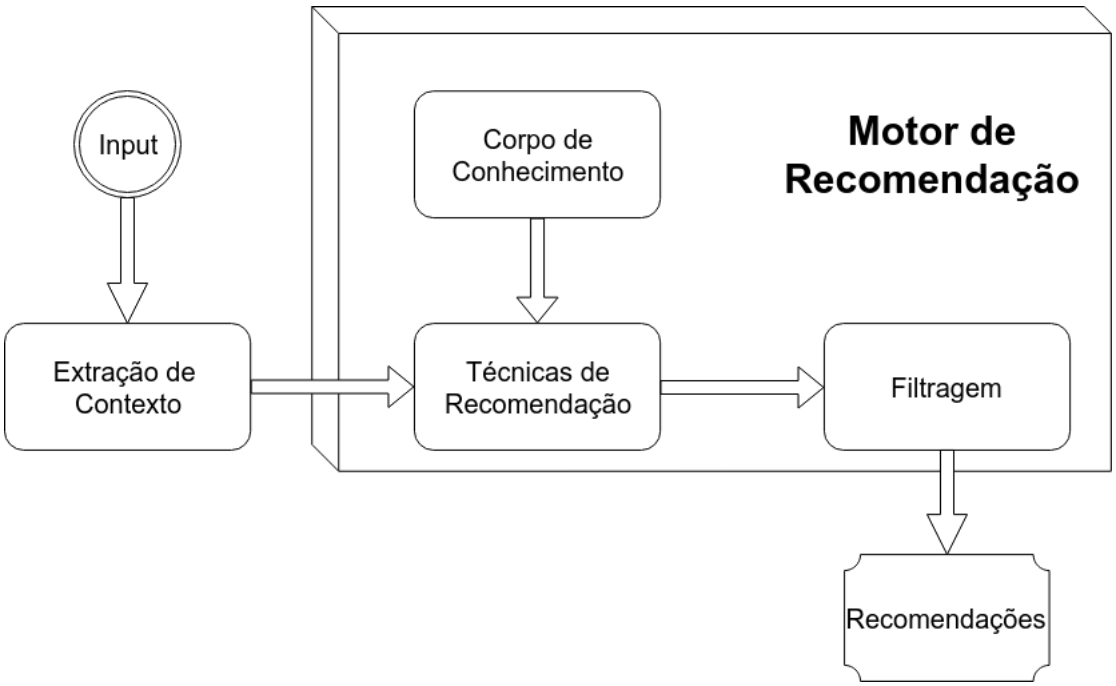

Figura 5.1: Passos de Construção para um RSSE. Traduzido e Adaptado de Maki (2016)

principalmente no tipo de corpo de conhecimento aplicado a RSSE. Nessa revisão foi possível identificar algumas áreas da ES que utiliza SR, apresentadas a seguir.

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

Por meio deste estudo, foi possível identificar em qual domínio de aplicação da industria de ES estão aplicando qual técnica de SR, apresentados na Tabela - 5.1 a seguir.

Tabela 5.1: Sumário de técnicas de recomendação em cada domínio, Traduzido e Adaptado de Maki (2016)

Domínios	**Técnicas**								**Número de Referências**
	CBF	CF	KBF	Hibrido	IA	Redes Sociais	Info. de Contexto	Grupo de agregação	
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			2	21
Total	**31**	**39**	**36**	**41**	**27**	**5**	**12**	**2**	**104**

5.3 Concepção

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

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

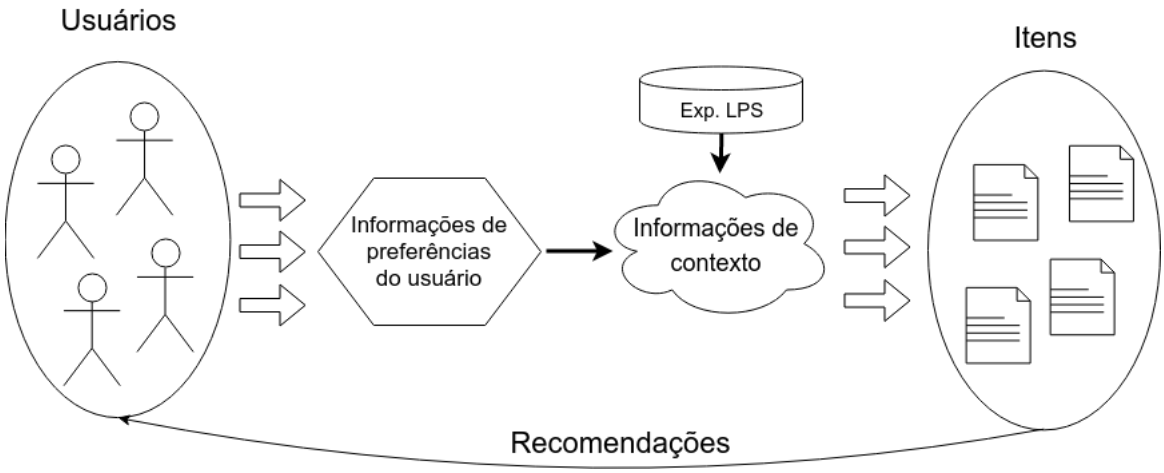

Figura 5.2: Modelagem geral do SR como prova de conceito para OntoExer-LPS

A Figura - 5.2 apresenta o conceito geral da metodologia deste projeto. Iniciando pela entrada dos usuários, depois extraímos as preferências dele (por meio de ratings), e em seguida realizada a extração de informações de contexto utilizando a base de metadados em LPS (OntoExper-LPS), 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 sendo definido da seguinte forma:

- Artefatos de LPS:
- Tipo do experimento LPS

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

5.4 Projeto

Para o desenvolvimento o SR foi necessário encontrar tecnologias que atendesse os requisitos, tanto de SR como de ontologias. A linguagem de programação Python foi escolhida novamente por atender estes dois requisitos. Quanto a ontologia, porque ela 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 Framework Django 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 prototipação da tela de iteração do usuário com SR;
- **Fase 2:** carregamento da OntoExper-LPS no SR e implementação dos mecanismos de consultas na OntoExper-LPS;
- **Fase 3:** *ratings* dos usuários no SR e armazenado da avaliação no SR; e
- **Fase 4:** implementação do modelo de recomendação, implementação da o conceito de filtragem colaborativa.

5.4.1 Fase 1: O Projeto de SR Usando Django Framework

O 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)* , porém com outra nomenclatura. Abaixo está descrito 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.

- **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 possibilidades 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 parte 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 - 5.3

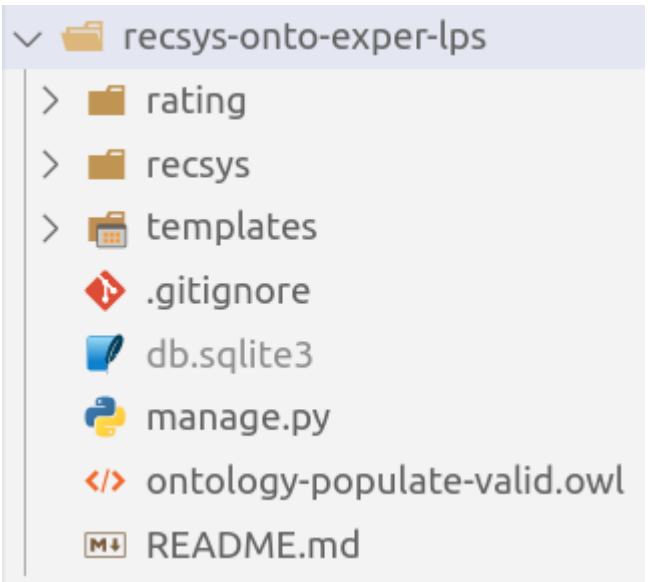

Figura 5.3: Árvore de diretórios da aplicação SR com Django Framework.

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* (confiuguração global do projeto) e *views.py* que é o principal aquivo deste projeto. Nele contém toda regra e controle das funcionalidades implementada 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-LPS 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-LPS no SR

Como foi apresentado na seção anterior, o arquivo ontology−populate−valid.owl está localizado na raiz do projeto, para nosso 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-LPS

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

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

Listing 5.2: Carregando a OntoExper-LPS

```
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-LPS. Foi criado um métodos *_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.

A Figura - 5.4 apresenta a tela construída para SR. Nesta 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. Neste método é realizar a inferência na ontologia, segundo os valores que o usuário selecionou na tela. Na listagem 5.3 apresenta o código da consulta realizada.

Listing 5.3: Inferência na OntoExper-LPS

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

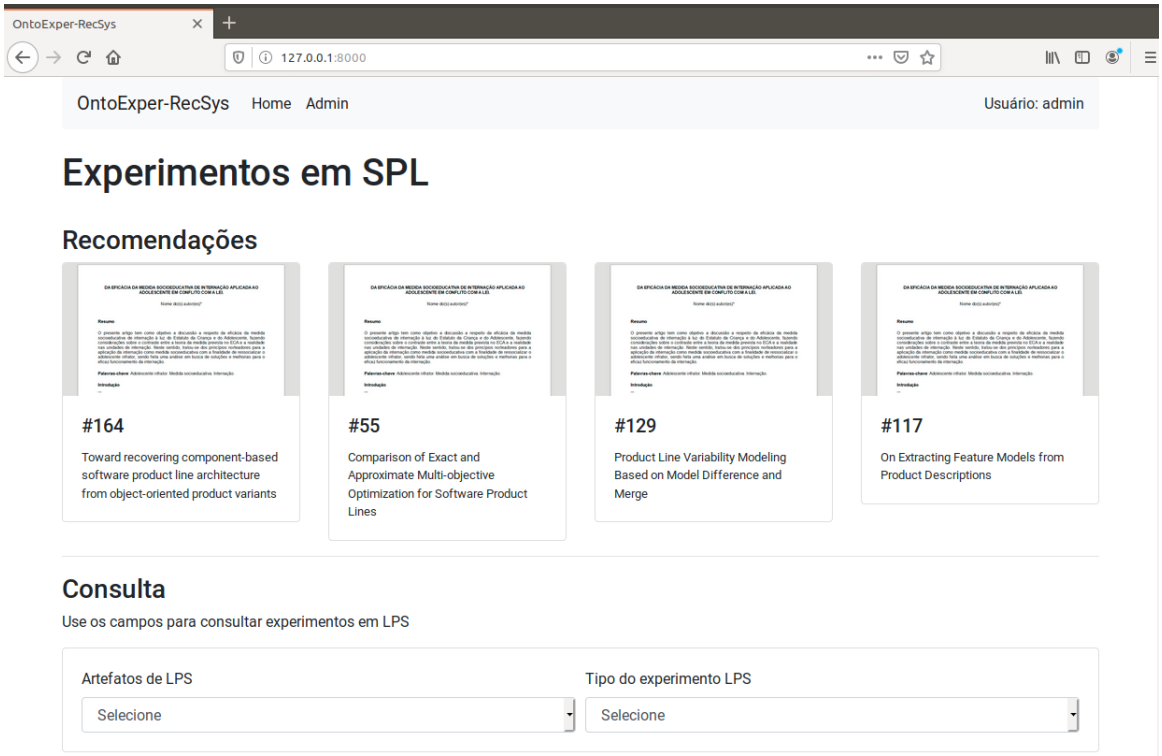

Figura 5.4: Tela do SR.

```
        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* explícitos e o *rating* implícito.

- No *rating* explícitos 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. Nesta 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 - 5.5 apresenta a tela que com as opções de *rating* explícito.

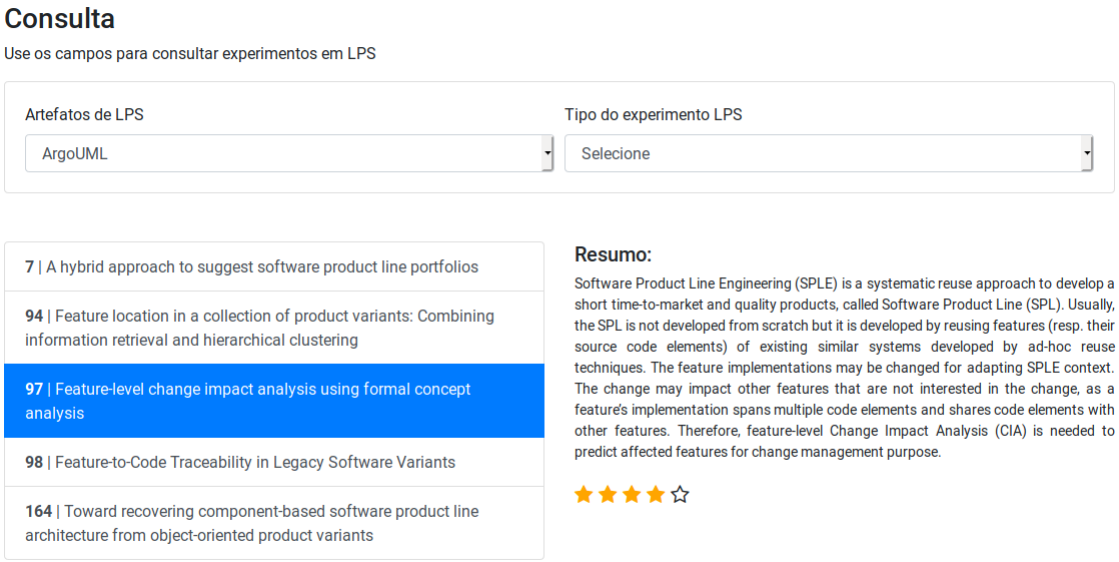

Figura 5.5: *Rating* Explícito no SR.

Quando o usuário dá 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 detalhado a modelagem destes dados.

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

Neste projeto foi utilizado 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 ranques-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.

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], neste caso podemos observar que o usuário-1 avaliou seis experimentos e usuário-2 sete sendo eles #102, #117, #125, #13, #9, #129 e o #170 (estes 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 descrita na Equação 5.2.

$$DE(x, y) = \sqrt{\sum_{i}^{p} (x_i - y_i)^2} \tag{5.2}$$

  Usamos 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. Este cálculo fica mais interessante quando aplicamos mais dimensões, cada dimensão no no nosso cenários pode ser associada a um experimento, por exemplo, se incluirmos no cálculo o experimento #129, temos 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, simplesmente aplicando a Equação 5.3. Dessa forma podemos afirmar que neste exemplo a similaridade entre o usuário-1 e o usuário-2 é 0.25

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

  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 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 SR recomendaria o experimento que #170 por ter maior nota.

$$NOTA_p() = \frac{\sum_i^{usuarios} (sim_i * nota_i)}{\sum_i^{usuariosTemNota} (sim_i)} \tag{5.4}$$

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**	

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 melhor solução seria implementar a filtragem baseada em conteúdo, pois este 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 SQLite, ele foi utilizado por ser o banco de dados padrão para Django Framework, dessa forma não foi preciso realizar nenhuma

outra configuração a mais. O SQLite é um dos bandos de dados mais simples, todos os registros de metadados são armazenados em um único arquivo.

As tabelas 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;
- rating_rating; e
- sqlite_sequence.

As tabelas com prefixo *auth* e *django* são geradas e mantidas pelo próprio Django Framework. A tabela *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: Tabela *rating_rating* do Banco de Dados

Coluna	Tipo
id	integer
user	varchar(50)
rating	real
id_experiment	integer
title_experiment	varchar(255)

5.6 Ambiente de Desenvolvimento

Para desenvimento do SR, foi necessária algumas ferramentas de apoio, além do Django Framework e do SQLite já citado, foi utilizado outras tecnologias como HTML CSS e JS, com o apoio dos *frameworks* Bootstrap e jQuery.

Para ambiente de desenvolvimento de código fonte foi utilizado o editor de texto VSCode 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 Browser. Todas estas ferramentas são *open source* e estão disponivel para *download* até a data de escrita deste trabalho.

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

5.7 Empacotamento e Compartilhamento

Os documentos disponibilizados aos participantes e resultados deste estudo qualitativoest ?ao dispon ??veis em: https://doi.org/..., visando dissemina ?c ?ao 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-LPS. 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-LPS 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 cap??tulo, será apresentado a conclusão final deste trabalho de mestrado.

6

Conclusão

A realização de experimentos de ES em LPS traz informações relevantes para fornecer evidências de uma teoria para o mundo real. A capacidade de compreender, estudar, e replicar experimentos tornam este 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 em 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 organizada, estrutura, personalizadas e úteis, principalmente pela oportunidade da criação de aplicações auxiliares neste sentido, como por exemplo sistemas de recomendações, bem como objetivo, incentivar a cultura de experimentos pela indústria de software.

Neste contexto, foi desenvolvido uma proposta a um modelo de ontologia, a OntoExper-LPS, com o objetivo de organizar e estruturar o conhecimento adquirido sobre experimentos de ES em LPS. Este corpo de conhecimento foi previamente levantado em um trabalho de dissertação de mestrado do grupo de pesquisa GReater (Furtado, 2018), onde foram levantados 211 artigos que relatam experimentos em LPS. Com este levantamento foi possível elaborar um modelo de ontologia a fim de representar o conhecimento sobre experimentos de ES, bem como entendê-lo para LPS, em seguida inserimos os dados dos artigos neste modelo, chamamos esta fase de inserção dos indivíduos na ontologia. Desta forma estruturamos a *TBox* e *Abox* da ontologia proposta.

Com a composição da OntoExper-LPS foi possível responder a questão de pesquisa desta dissertação: **Como formalizar o conhecimento de experimentação em LPS?**. A OntoExper-LPS foi avaliada por um estudo quantitativo, na qual foram avaliados 8 critérios de qualidade desenvolvidos por /citeDenny Vrandecic em sua tese *Ontology Evaluation*, por meio de uma escala Likert, onde participaram especialista de ES experimental, LPS e Ontologias. Também foi possível realizar uma breve avaliação em relação a algumas armadilhas que o modelo da OntoExper-LPS poderia possuir, usamos a ferramenta OOPS!.

Os resultados obtidos do estudo quantitativo forneceram evidências preliminares que a OntoExper-LPS é viável para formalizar o conhecimento de experimentação em LPS com um nível de qualidade satisfatório, tornando a OntoExper-LPS possível de ser estendida para outras áreas de experimentos em ES.

Foi elaborado um protótipo de sistema de recomendação como uma prova de conceito de que é possível realizar inferência e extrair informações relevantes da OntoExper-LPS. O resultado foi um SR capaz de interagir com usuários e eles podem realizar consultas e receber recomendações a partir de inferências na OntoExper-LPS.

6.1 Contribuições

A principal contribuição desta pesquisa foi a formalização do conhecimento sobre experimentação em LPS. LPS são construída por meio de um domínio de aplicação, similaridades, o core assessments, e variabilidades, o que distingue um produto do outro dentro da família de produtos, todas estas características estão presente na OntoExper-LPS. Entretanto este modelo proposto pode ser facilmente extensível 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 de um formato de estudo qualitativo aplicando uma escala Likert para avaliação dos critérios de qualidade a ontologias;
- um sistema de recomendação com a finalidade de recomendar experimentos em LPS de maneira mais relevante para o usuário que está interagindo como 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.

Os seguintes artigos serão submetidos:

- Vignando, H.; OliveiraJr, E. An Ontology for Software Product Line Experiments - SAC '20: ACM/SiGAPP Symposium on Applied Computing. (Em Preparação)

6.2 Limitações

As limitações deste trabalho são discutidas nesta seção.

A OntoExper-LPS proposta consideram elementos experimentais gerais e da abordagem de LPS, portanto, não é poss??vel generalizar para outros dom??nios.

O número de especialistas que participaram do estudo quantitativo para avaliar os critérios de qualidade da OntoExper-LPS. Apesar do conhecimento avançado dos participantes, acredita-se que novas avaliações são necessárias, bem como a melhoria do questionários e replicação de tal estudo com um maior número 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:

Identificamos que no estudo qualitativo existem vários pontos de melhoria na modelagem da OntoExper-LPS, estes ajustes devem ser realizados para padronizar o modelo com o objetivo de tornar possível a extensão e divulgação da mesma, principalmente em relação a eficiência computacional da mesma, dessa forma viabilizando a implementação de novas aplicações de inferência.

Estender a OntoExper-LPS para incluir metadados de qualidades já definidos no trabalho de (Furtado, 2018).

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

REFERÊNCIAS

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.

Blondet, G.; Le Duigou, J.; Boudaoud, 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.

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.

Dieste, O.; Juristo, N. Systematic review and aggregation of empirical studies on elicitation techniques. *IEEE Transactions on Software Engineering*, v. 37, n. 2, p. 283–304, 2011.

Dieste, O.; Juristo, N. Challenges of evaluating the quality of software engineering experiments. In: *Perspectives on the Future of Software Engineering*, Springer, p. 159–177, 2013.

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, 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.

Gruber, T. What is an ontology. *WWW Site http://www-ksl. stanford. edu/kst/whatis-an-ontology. html (accessed on 07-09-2004)*, 1993.

Halmans, G.; Pohl, K. Communicating the variability of a software-product family to customers. *Software and Systems Modeling*, v. 2, n. 1, p. 15–36, 2003.

Kampenes, V. Quality of design, analysis and reporting of software engineering experiments: A systematic review. 2007.

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.

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.

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.

MCKINNEY, W. Data structures for statistical computing in python. In: *Proceedings of the 9th Python in Science Conference*, USA, 2010, p. 51 – 56.

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`

NORTHROP, L.; CLEMENTS, P.; BACHMANN, F.; BERGEY, J.; CHASTEK, G.; COHEN, S.; DONOHOE, P.; JONES, L.; KRUT, R.; LITTLE, R.; ET AL. A framework for software product line practice, version 5.0. *SEI.–2007–http://www. sei. cmu. edu/productlines/index. html*, 2007.

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.

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.

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.

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.

Teixeira, E. O. Análise da qualidade de experimentos controlados no contexto da engenharia de software empírica. 2014.

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: Tipologias para Ontologias

A.1 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, casualidade, comportamento, funções etc.

A.2 Quanto ao grau de formalismo Uschold; Gruninger (1996)

- **Ontologias altamente informais:** expressa livremente em linguagem natural;
- **Ontologias semiinformais:** expressa em linguagem natural de forma restrita e estruturada;
- **Ontologias semiformais:** expressa em uma linguagem artificial definida formalment;
- **Ontologias rigorosamente formais:** os termos são definidos com semântica formal, teoremas e provas.

A.3 Quanto à aplicação Jasper; Uschold (1999)

- **Ontologias de autoria neutra:** uma ontologia escrita em uma única língua e depois convertida para uso em diversos sistemas, reutilizando-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 termo.

A.4 Quanto à estrutura Haav; Lubi (2001)

- **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.

A.5 Quanto ao conteúdo Van-Heijist; Schreiber; Wielinga (2002)

- **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 descreve;
- **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 campo;
- **Ontologias de representação:** Explicam as conceituações que estão por trás dos formalismos de representação do conhecimento.

Apêndice A: Código Fonte do Modelo da SMartOntology

B.1 Saída Da Opção time-flags com os Breakpoints `.dijkstra e .main`

```
-------------------------------------------------------------------------------------------
-     TOOLCHAIN  UNB-RISC16                  -
-     Energy Profile Version 1.00            -
-                                            -
-     Trip Time Flags                        -
-     Options: time-flags & BreakPoints: .dijkstra and .main
-                                            -         Sumary
-     Program   dijkstra.o
-                                            -
-                                            -         1-Times
-                                            -            1.1    Real time
-                                            -            1.2    Processor Time
-                                            -         2-Flags Reached
-------------------------------------------------------------------------------------------
- Section 1 [Times]
-
-      1.1  Time Consumption - Real Time (in seconds):                0
-      1.2  Time Consumption - Processor UNB-RISC16 (in seconds):     0,000000688
-------------------------------------------------------------------------------------------

- Function .d Reached After 0 Cycles From the Last Begin
- Function .main Reached After 8 Cycles From the Last Point
-------------------------------------------------------------------------------------------
```

Apêndice B: Código Fonte do Povoamento de Dados no Modelo SMartOntology

Apêndice C: Questionário de Avaliação e Respostas Aplicado à SMartOntology

Apêndice D: Código Fonte dos Exemplos de Aplicação de Recomendação Usando SMartOntology