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

Projeto de Dissertação de Mestrado

HENRIQUE VIGNANDO

Uma Ontologia para Experimentos e Quasi-Experimentos Controlados para Linha de Produto de Software - SMartOntology

Maringá
2020

HENRIQUE VIGNANDO

Uma Ontologia para Experimentos e Quasi-Experimentos Controlados para Linha de Produto de Software - SMartOntology

Projeto de dissertação apresentado 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

Uma Ontologia para Experimentos e Quasi-Experimentos Controlados para Linha de Produto de Software - SMartOntology

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 (SMartyOntology) 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, suportada pelo ambiente Protégé para verificação de sintaxe e avaliação inicial. A ontologia foi preenchida com mais de 150 experimentos em linhas de produtos de software, reunidos em um estudo de mapeamento sistemático. Surge também a oportunidade de investigar a elaboração de um sistema de recomendação para experimentos em LPS se baseando em na modelagem de informação estruturada pela ontologia proposta. Portanto, este trabalho apresenta conceitos fundamentais para elaboração de uma ontologia voltada para experimentos em LPS, e para criação de 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. Na qual, possa levar qualidade para os projetos experimentais e resultados obtidos por meio dos experimento recomendados. Ontologias e Sistemas de recomendação são bem conhecidos na ES, acredita-se ser possível aplicar essas teorias para recomendar experimentos em LPS. Apresentamos também uma avaliação de viabilidade da ontologia proposta. Ao final temos um sistema de recomendação que apresenta bons resultados em recomendações para experimentos controlado. Espera-se também 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. Qualidade de Experimentos. Sistemas de Recomendação em Engenharia de Software. Sistemas de Recomendação.

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
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*

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 (**?**). Essa discussão permite que o conhecimento seja gerado de forma sistemática, disciplinada, quantificável e controlada (**?**). 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.

Por outro lado, os sistemas de recomendação em ES vêm ganhado espaço por causa de própria evolução dos sistemas de recomendação tradicionais no mercado, como por exemplo Netflix, Spotify, Amazon, etc. Os sistemas de recomendação têm como objetivo recomendar algo mais atrativo para seu usuário, no caso de sistema de recomendação em ES o espaço de informação para gerar recomendação são os próprios recursos do ambiente de projeto e desenvolvimento de software, como código fonte, revisões do gerenciador de revisões de código e *issues tracker* (**?**).

Dessa forma, acredita-se haver uma oportunidade de poder juntar essas duas áreas de pesquisa, na qual, será possível investigar o estudo de sistemas de recomendação para apresentar recomendações no contexto de experimentos e *quasi*-experimentos controlados em LPS.

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

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

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.

Realizar um experimento em LPS com qualidade, exige-se uma experiência considerável em ES e o mínimo de experiência em LPS. Com isso, a curva de aprendizado torna-se longa para extrair e apresentar de forma satisfatória, e com qualidade os resultados de experimentos em LPS. Além disso, a indústria tem adotado de forma crescente o conceito de LPS, desta forma, cada vez mais exigente por um corpo de conhecimento na área. Portanto, é um desafio para os estudantes e profissionais de ES poder realizar um experimento em LPS com qualidade.

Este trabalho propõe a especificação e implementação de um sistema de recomendação a fim de gerar recomendações de experimentos em LPS. Dessa forma, o usuário do sistema de recomendação terá processos e diretrizes confiáveis para a realização de seus experimentos em LPS. Espera-se que com estes processos e diretrizes os usuários possam planejar, executar, analisar e reportar experimentos em LPS, sendo assim, não comprometendo a replicação e auditoria dos mesmos.

1.2 Motivação e Justificativa

Realizar um experimento em LPS exige alguns pontos de atenção específicos para garantir a qualidade do experimento. Estes pontos tem sido investigados em um trabalho de mestrado em andamento do nosso Grupo de pesquisa em Reuso Sistemático de Software e Experimentação (GRSSE), neste trabalho vem sendo 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 influenciem 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 um sistema de recomendação que recomende métodos, processos, diretrizes, entre outros, para realizar um experimento em LPS,

pode proporcionar facilidade ao desenvolvimento dos mesmos, incentivando a cultura e desenvolvimento de experimentos na academia e industria.

Por meio do GRSSE, foi encontrada uma lacuna nas pesquisas de qualidade em experimentos de ES em LPS que proporciona esta pesquisa, apresentando um campo aberto à pesquisa para determinar qualidade e recomendação para experimentos em LPS.

1.3 Objetivos

Esta pesquisa tem como objetivo geral especificar e implementar um sistema de recomendação para experimentos em LPS caracterizados por sua qualidade

Os objetivos específicos deste projeto são:

- gerar e representar um conjunto de meta dados a partir das informações sobre experimentos em LPS;
- definir técnicas de recomendação com base nos meta dados;
- projeto e desenvolvimento do sistema de recomendação e;
- avaliar e empacotar o sistema de recomendação.

1.4 Metodologia de Desenvolvimento

O processo de execução deste trabalho para chegar ao objetivo será, pesquisar de maneira exploratória um modelo de sistema de recomendação em experimentos de LPS. Para tal resultado será desenvolvido um projeto de software e em seguida executado o mesmo. A **??** apresenta as etapas de desenvolvimento desta pesquisa.

Neste projeto de pesquisa vamos levantar uma base de dados sobre a qualidade de experimentos em LPS. Por meio desta base será possível gerar um modelo de dados baseado em ontologias, **TBox** e **ABox**, com o objetivo de extrair informações preditivas desta base. Após processar essas informações baseada nos meta dados de qualidade de experimentos em LPS, será possível criar um sistema de recomendação, utilizando as ferramentas apropriadas de recomendação em ES.

- **O Papel da Ontologia:** será de estruturar e modelar a base de informações extraída do Mapeamento Sistemático de experimentos em LPS que está sendo desenvolvido pelo GRSSE. Pode se dizer que este modelo será o conjunto de meta dados;

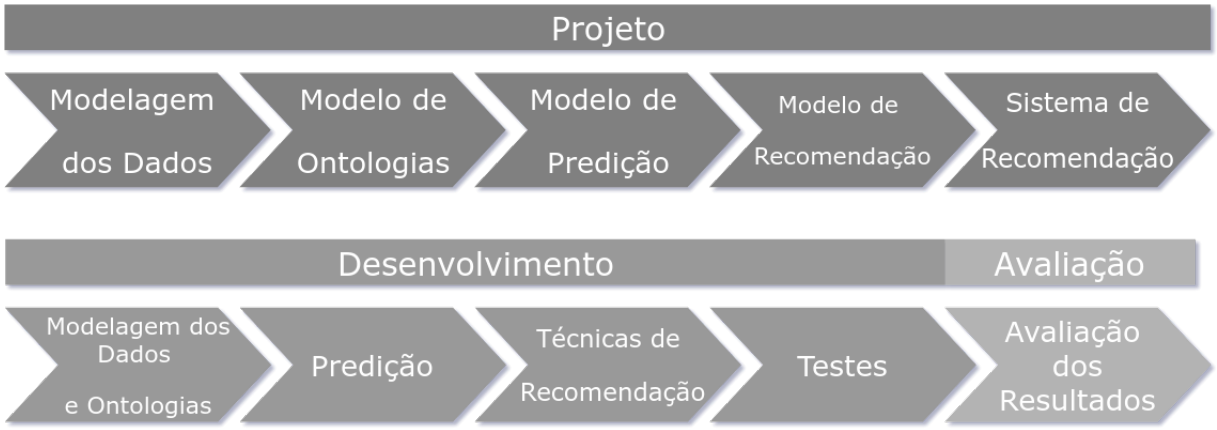

Figura 1.1: Etapas da Metodologia de Desenvolvimento de Pesquisa

- **O Papel do Sistema de Recomendação:** será de interagir com o usuário afim de extrair informações relevantes, de modo que se possa determinando o deste perfil do usuário, para que possa ser realizada inferências no modelo ontológico de diretrizes de qualidade gerando recomendações de métodos, processos, diretrizes para o experimento do usuário.

O desenvolvimento do projeto, inclui realizar a escolha das tecnologias a serem usadas como ferramenta de construção do software, como por exemplo, as linguagens de programação, o ambiente de desenvolvimento, a diagramação do projeto, os *stakeholders* envolvidos no projeto, ferramentas de *Application Lifecycle Management* (ALM) aplicadas ao escopo do projeto, escolha das abordagens de sistemas de recomendação, definição do modelo de ontologias, definição da base de dados para representação tanto, dos dados de origem (itens, usuários), quanto, para representação dos dados para apresentação e armazenamento dos resultados obtidos da recomendação.

Após a definição do projeto de software, inicia-se o processo de desenvolvimento do sistema de recomendação. Com o auxílio de ferramentas de ALM será possível acompanhar por todos os *stakeholders* envolvidos o desenvolvimento online da ferramenta, desta forma se tornando um processo mais colaborativo entre eles. Inicialmente será realizado a modelagem dos dados extraído da avaliação de qualidade dos experimentos em LPS realizado pelo trabalho do GRSSE, que são 174 experimentos encontrado na literatura nesse ramo de pesquisa. Em seguida será aplicado um modelo de ontologia definido no projeto de software, nesta base de informações de experimentos, tem como propósito, realizar predições para um modelo mais abstrato sobre qualidade de experimentos em LPS. Na sequência será desenvolvido o modelo de recomendação, este desenvolvimento consiste na modelagem dos dados encontrado na predição da ontologia para extrair as informações

necessárias para o modelo de recomendação, posteriormente aplicar os algoritmos neste modelo. Em seguida será desenvolvido um *front-end* de interação com o usuário poder dar entrada na informações iniciais para gerar as recomendações. O último passo será realizado um estudo para avaliação deste sistema de recomendação.

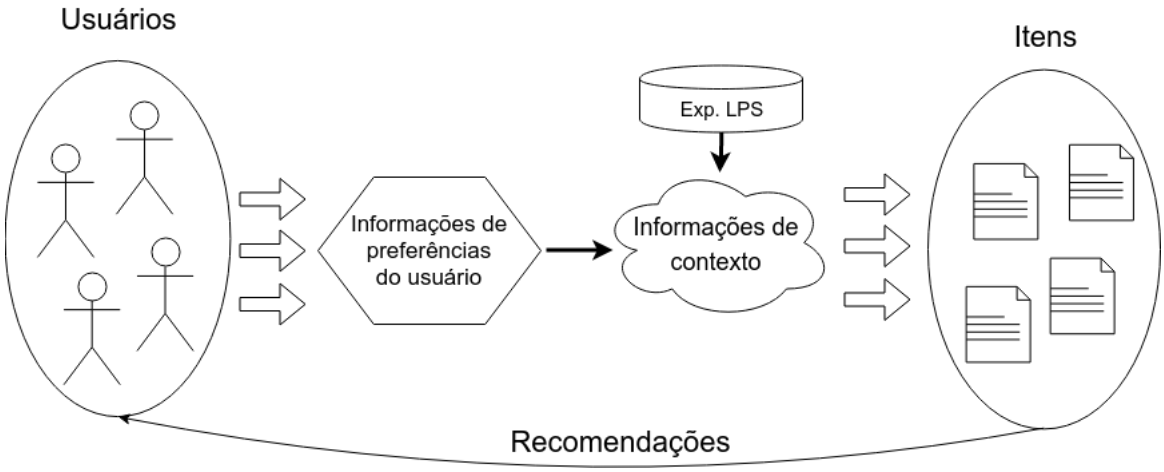

Figura 1.2: Modelagem geral da RSSE proposta

A **??** apresenta o conceito geral da metodologia deste projeto. Iniciando pela entrada dos usuários, depois extraímos as preferencias dele, e em seguida será feita a extração de informações de contexto utilizando a base de meta dados em LPS, para então fazer inferência nos itens (que são experimentos em LPS), por meio dessa inferência será obtido as recomendações.

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

- Informações de LPS:

  Dominio;

  Sub-dominio;

  Tipos de artefatos e;

  Feature module.

- Experimentos

  filtrar por qualidade.

1.4.1 Empacotamento

Todo projeto está sendo versionado no Github pelo link: https://github.com/rickvig/pcc-pesquisa.

1.5 Organização do texto

Este documento está estruturado da seguinte forma: o Capítulo 2 apresenta a Fundamentação Teórica sobre Linha de Produto de Software, Experimentos e Quasi-Experimentos em Engenharia de Software, qualidade de experimentos e quasi-experimentos em Engenharia de Software, Ontologias e Sistemas de Recomendação tradicionais e Sistemas de Recomendação em ES; o Capítulo 3 apresenta uma Ontologia para Experimentos de LPS; e o Capítulo 4 apresenta as considerações finais acerca deste projeto de dissertação de mestrado.

2

Fundamentação Teórica

Este tópico apresenta conceitos fundamentais sobre Linha de Produto de Software, Experimento e *Quasi*-Experimento em Engenharia de Software, Qualidade de Experimentos e *Quasi*-Experimento em Engenharia de Software, Ontologia e Sistemas de Recomendação tradicional e voltado para Engenharia de Software.

2.1 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 (**?**). 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 (**?**). 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 **?**, "uma característica é um comportamento característico ou visível ao usuário final de um sistema de software". Uma característica pode ser obrigatória, opcional ou alternativa. O modelo de características representa as variabilidades e as variantes de uma LPS (**?**).

- Variabilidades são descritas por: Ponto de variação que permite a resolução de variabilidades em artefatos genéricos de uma LPS, e;

- Variante é representa pelos: os 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 (**?**).

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.

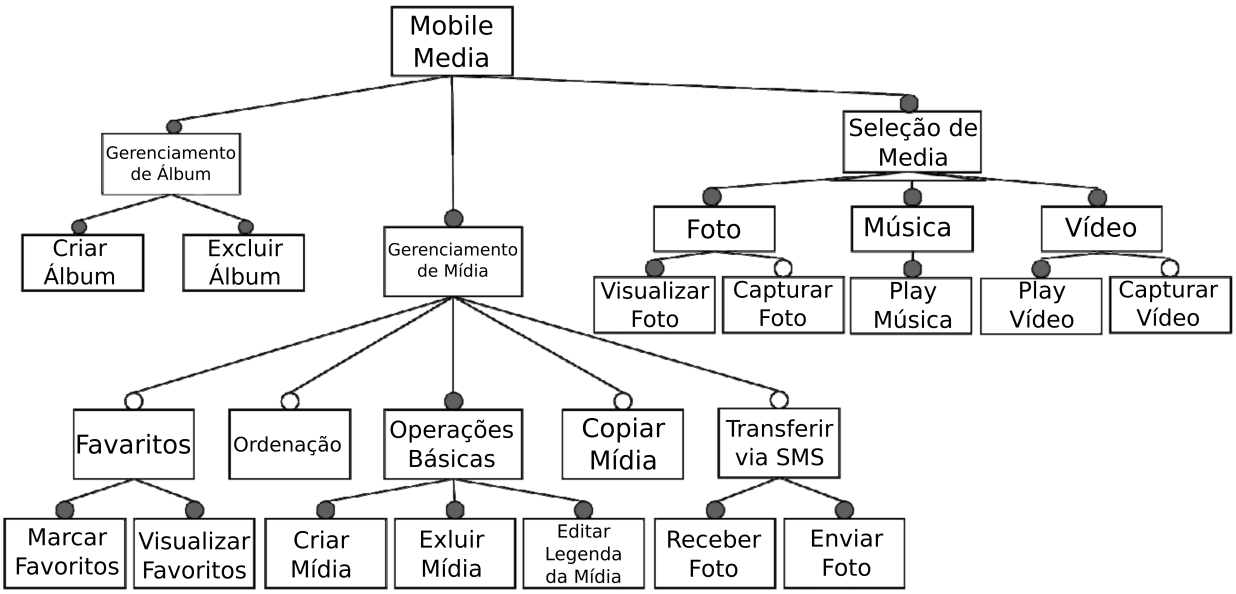

Figura 2.1: Um modelo de Características. Traduzido de **?**

A **??** 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 representa características opcionais. As arestas ligadas por um triangulo, 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 instancia deste exemplo teria as seguintes características no seu produto de software final:

- Gerenciamento de Álbum;

  Criar Álbum;

  Excluir Álbum;

- Gerenciamento de Mídia;

  – Operações Básicas

    Criar Mídia

    Excluir Mídia

    Editar Legenda Mídia

  – Favoritos;

    Marcar Favoritos

    Visualizar Favoritos

- Seleção de Mídia;

  Foto

  Visualizar Foto

  Capturar Foto

? 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 **??**.

- **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.

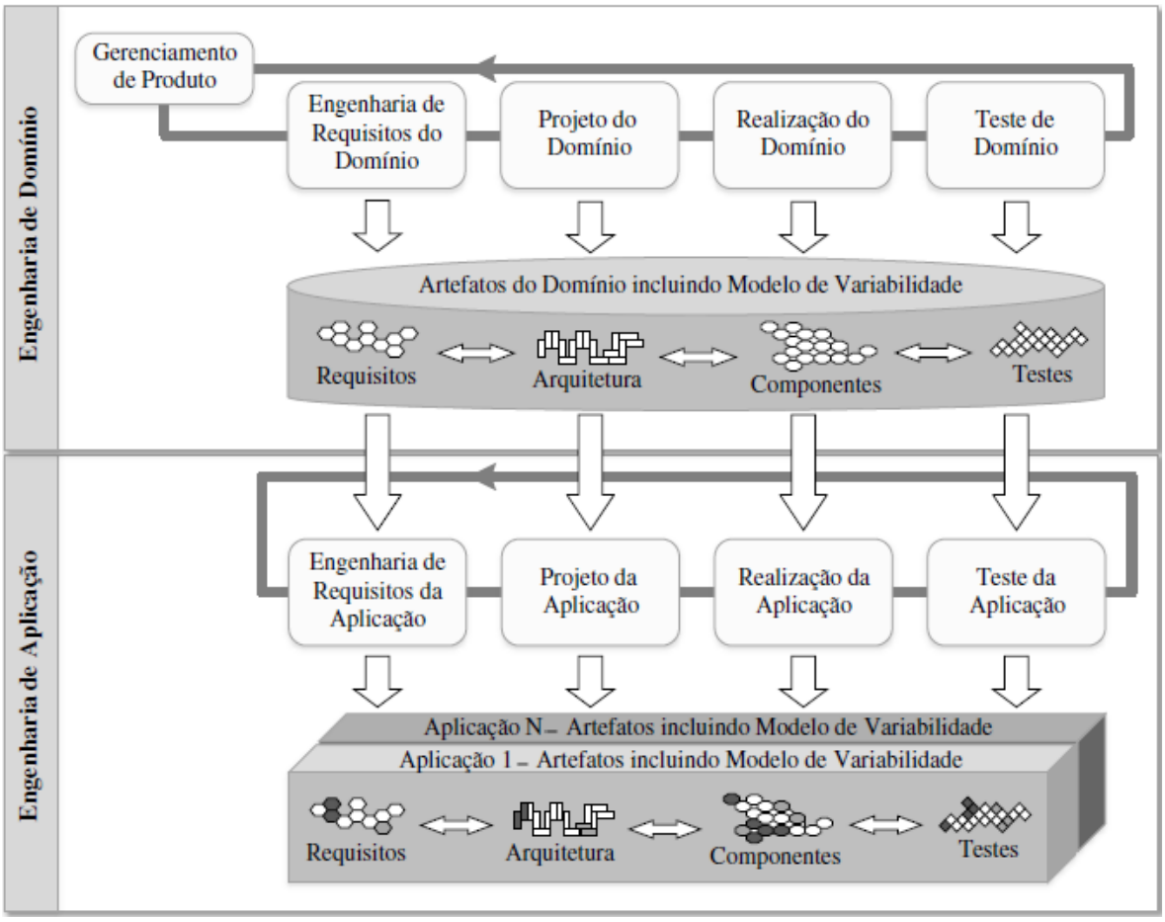

Figura 2.2: *Framework* de Engenharia de LPS (Pohl et al., 2005). Traduzido por Geraldi (2015)

2.2 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 (**?**).

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 (**?**). A **??** apresenta à ideia de uma relação causa e efeito em teoria, na qual a parte superior à linha trastejada se encontra a teoria e, na parte inferior a observação (**?**).

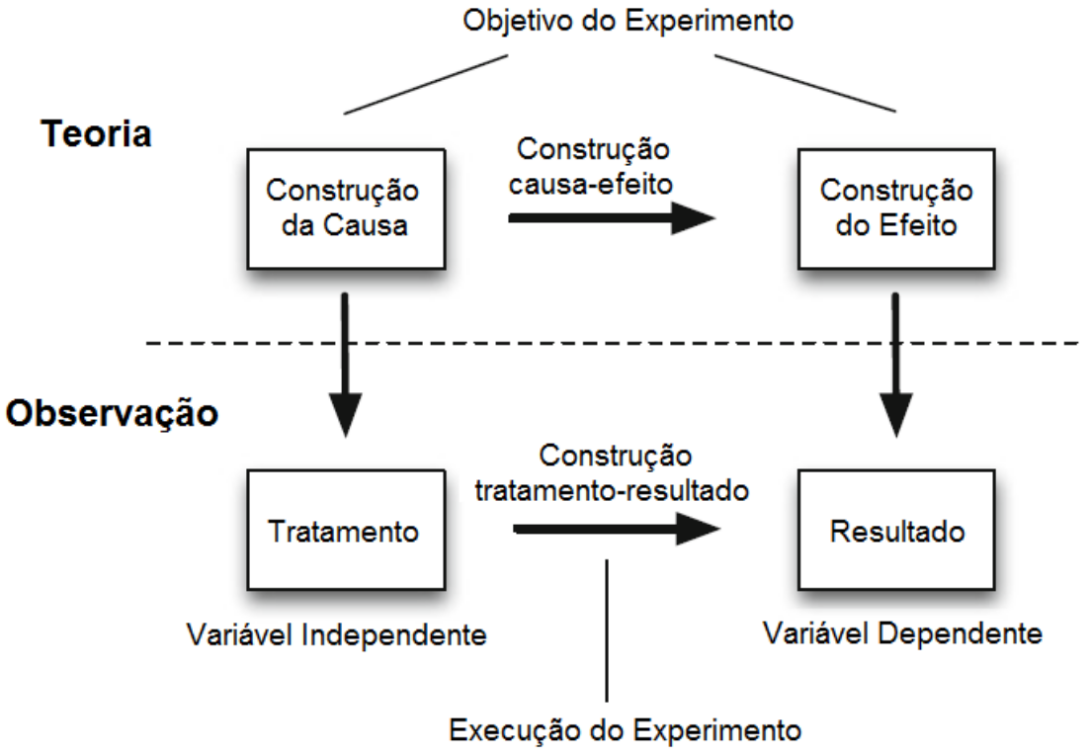

Figura 2.3: Conceitos Essenciais de um Experimento. Traduzido de **?**

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 (**?**).

Segundo **?**, existe uma característica dita fator de confusão em experimentos de ES que envolvem seres humanos. Esse fator pode ser representado pela presença de algum elemento indesejável no estudo que dificulta distinguir entre duas ou mais causas possíveis

de um efeito que foi medido pela variável dependente como, por exemplo, os níveis de habilidade dos participantes e a extensão de suas experiências anteriores com o objeto experimental.

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

Segundo **?** a realização de um experimento pode ser dividido em um processo contendo cinco atividades, conforme apresentadas na **??** e descritas a seguir:

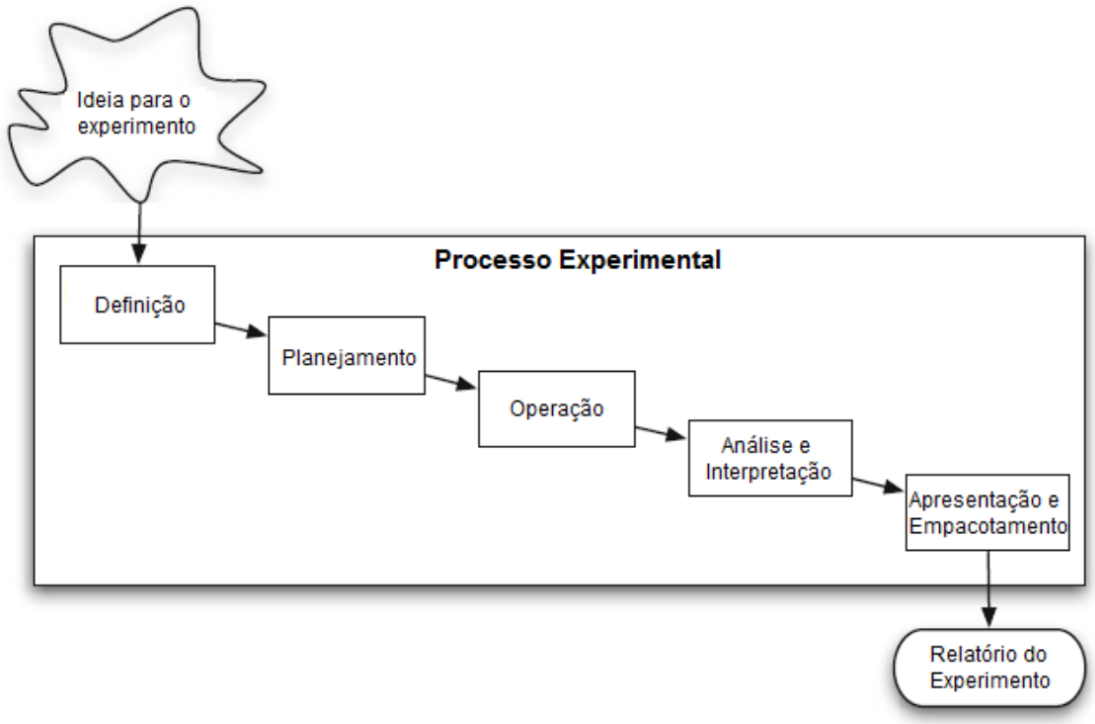

Figura 2.4: Visão Geral do Processo Experimental. Traduzido de **?**

- **Definição:** é a primeira atividade, onde define-se o problema, objetivo e metas do experimento. Caso não seja devidamente estabelecida, pode ocorrer retrabalho ou o experimento não pode ser utilizado para se estudar o que era almejado;

- **Planejamento:** é uma preparação de como o experimento será conduzido, em que ocorre a determinação do contexto do experimento, a formulação das hipóteses, sendo **hipótese nula** que o experimentador espera rejeitar com a maior confiança possível e a **hipótese alternativa** que se espera aceitar, a seleção de variáveis (dependentes e independentes), a seleção dos participantes, o projeto do experimento, a instrumentação e a avaliação da validade, dividida em quatro tipos, sendo **validade interna** refere-se ao relacionamento tratamento-resultado; **validade externa** apresenta a generalização dos resultados a uma população maior; **validade de *constructo*** demonstra a relação entre a teoria e observação; 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 (**?**).

- **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.

2.3 Qualidade de Experimentos em Engenharia de Software

Segundo **?**, 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 **?**, 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 **??** 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 **?**

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 **?** 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 (**?**). 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 (**?**).

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* (contem ou não contem), escalas de qualidade, opinião de especialista (**???**).

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 (**?**).

Até o momento não se tem conhecimento de trabalhos que tratam a qualidade de experimentos em área especifica 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:

- **?** 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 selecionem apenas as questões do *checklist* mais adequadas ao contexto de suas próprias questões de pesquisa.

- **?** 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".

- **?** desenvolveram uma escala de qualidade para determinar a qualidade de experimentos, contendo dez questões baseadas nas cinco dimensões de **?**, sendo: contexto experimental, projeto experimental, análise, interpretação dos resultados e apresentação dos resultados. As respostas de cada questão são "sim"ou "não".

2.4 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 **?**, 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 **?**, 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*) (**?**). 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 **??** apresenta essa estrutura.

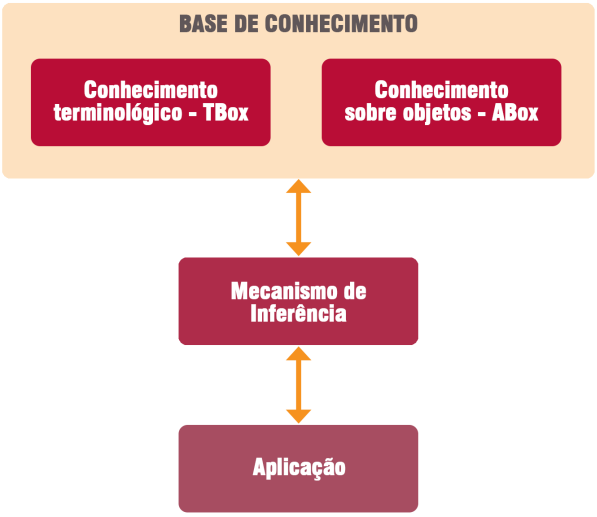

Figura 2.5: Estrutura de uma Ontologia. Autor

A **??** 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 trastejadas representam um relacionamento de propriedade, já as arestas que começam com um losango indica a definição de uma propriedade, especificando sua tipagem.

2.5 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 (**?**). Segundo **?** 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.

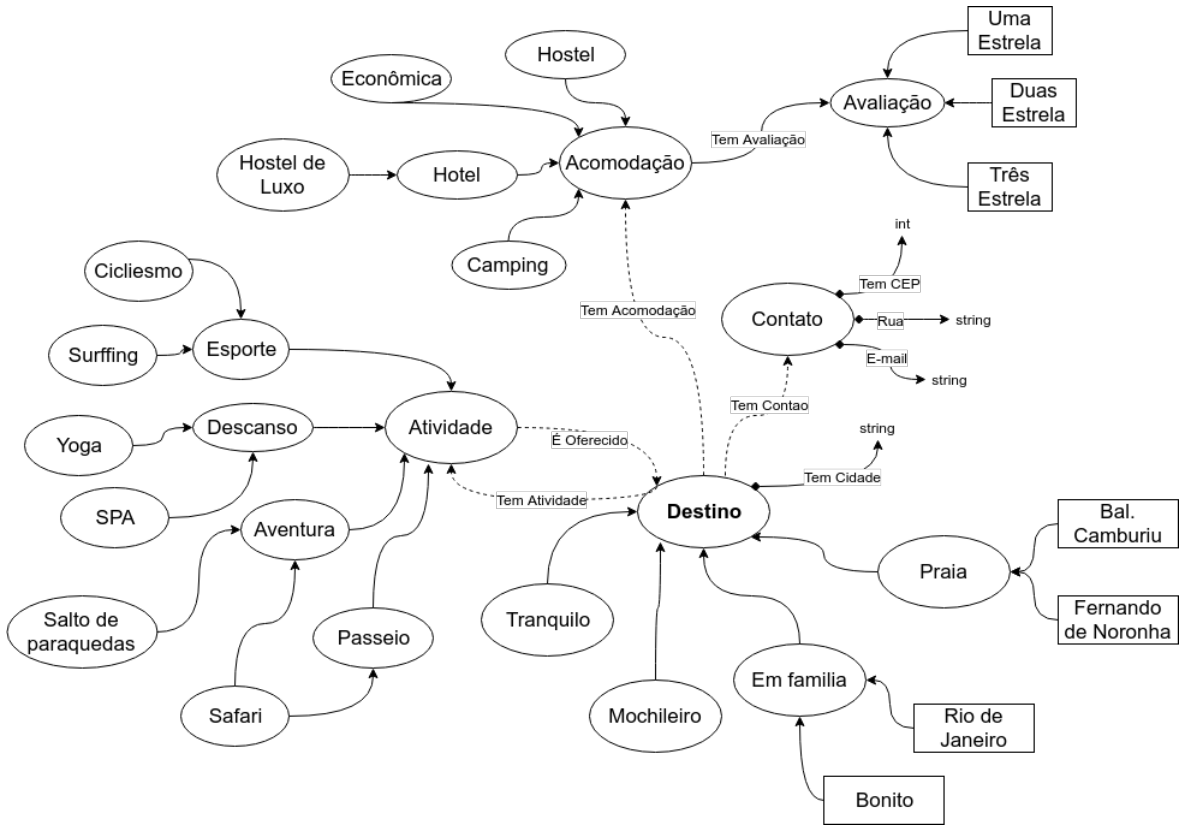

Figura 2.6: Exemplo de uma Ontologia para o domínio: Destino de Viagem. Autor

2.5.1 Sistemas de Recomendação Tradicional

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 (**??**). O primeiro sistema de recomendação proposto foi o *Tapestry*, nesse sistema criou-se um 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". Nesse trabalho, iniciou o desafio de casar corretamente os que dados recomendados com os usuários que o recebem, analisando o real relacionamento de interesse (**??**).

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

Definicão 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 $\boldsymbol{c} \in \boldsymbol{C}$*, procura-se um item* $\boldsymbol{s'} \in \boldsymbol{S}$ *que maximiza a utilidade do usuário. Isto pode ser expressado pela equação* **??***:*

$$\forall c \in C, s'_c = argmax_{s \in Su(c,s)} \tag{2.1}$$

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.

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

- ***Collaborative Filtering*:** A Filtragem Colaborativa baseia-se na ideia 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. (**?**). 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 (**?**).

- ***Content-based Filtering*:** A Filtragem Baseada em Conteúdo tem como característica principal o pressuposto de interesses pessoais, por exemplo, os usuários interessados no tópico de qualidade de experimentos em LPS normalmente não alteram

seu interesse de um dia para outro, mas também estarão interessados em um tópico próximo, como por exemplo experimentos em Sistema de Sistemas. Abstraindo, as abordagens de recomendação baseadas em conteúdo são aplicadas, por exemplo, quando se trata da recomendação de sites (notícias com conteúdo semelhante em comparação com o conjunto de notícias já consumidas) (**?**). Filtragem Baseada em Conteúdo depende de dois tipos diferentes de dados: (i) um conjunto de usuários e (ii) um conjunto de categorias (ou palavras-chave) atribuídas ou extraídas dos itens (descrições de itens). Os sistemas de recomendação de filtragem baseados em conteúdo calculam um conjunto de itens que são mais parecidos com itens já conhecidos pelo usuário atual (**?**).

- ***Knowledge-Based Recommendation*:** A recomendação baseada no conhecimento, baseia-se nos seguintes dados básicos: (i) um conjunto de regras (restrições) ou métricas de similaridade e (ii) um conjunto de itens. Dependendo dos requisitos do usuário, regras (restrições) que descrevam quais itens devem ser recomendados. O usuário atual articula suas necessidades (preferências) em termos de especificações e propriedades de itens que são internamente bem representados em termos de regras (restrições).

- ***Hybrid Recommender Systemson*:** São algoritmos que combinam *Collaborative Filtering* com *Content-based Filtering* e podem ser feitos de diversas formas diferentes, por exemplo, aplicando os dois separados e juntando os resultados depois, adicionando a saída de um ao outro ou unificando as duas abordagens em um único modelo. Alguns exemplos são as abordagens baseada em pesos, misturadas a cascatas (**?**).

- ***Context-aware Recommender Systems*:** Existem casos de recomendações que não podem levar em consideração somente os dados do item ou do usuário, como conteúdo personalizado de um site de filmes, sites de viagens e até sites de notícias. A incorporação do contexto permite personalizar ainda mais a recomendação e criar experiências realmente válidas ao usuário. Segundo **?** *Context-aware Recommender Systems*, segue as abordagens anteriores assumindo a existência de certos fatores contextuais como, por exemplo, o tempo e a localização, que identificam o contexto no qual as recomendações são fornecidas. Eles assumem que cada um desses fatores contextuais podem ter uma estrutura. O fator Tempo, por exemplo, pode ser definido em termos de segundos, minutos, horas, dias, meses e anos. **?** cita como classificar o contexto baseando-se nos seguintes aspectos, (i) o que um sistema de

recomendação pode saber sobre esses fatores contextuais e (ii) como os fatores contextuais mudam ao longo do tempo. Desta forma podemos definir este tipo de sistema de recomendação pela formula **??**

$$f : User \times Item \times Context \rightarrow Rating \tag{2.2}$$

2.5.2 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 (**?**).

? 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 (**?**).

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 (**?**).

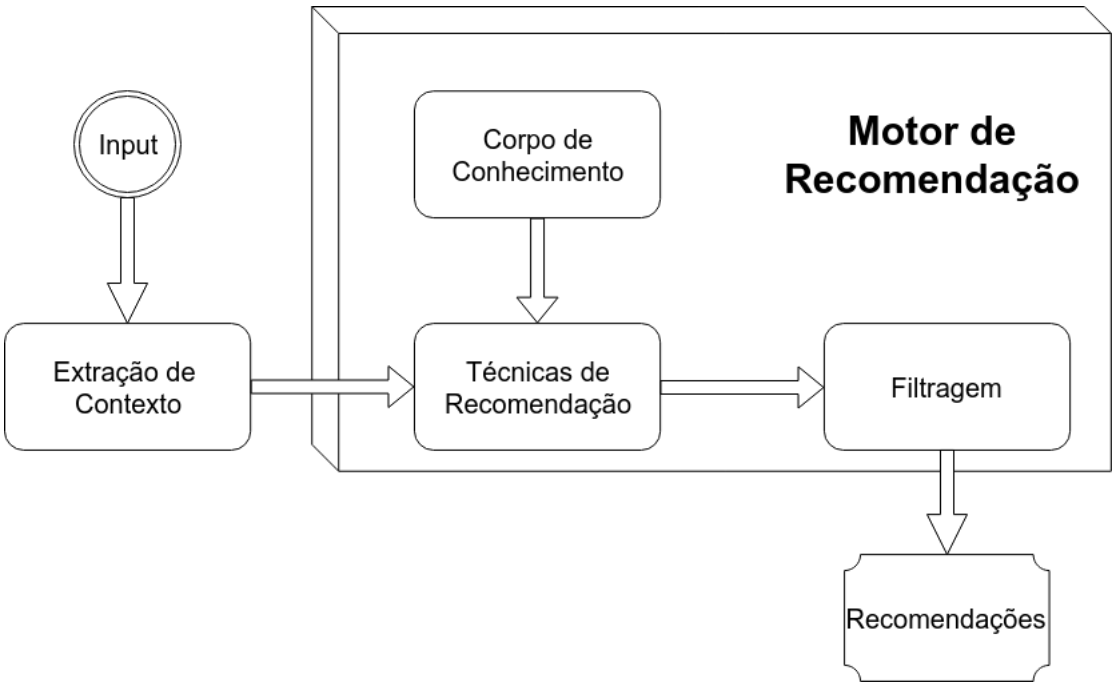

Figura 2.7: Passos de Construção para um RSSE. Traduzido e Adaptado de **?**

A **??** 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 especifico para cada área de ES), depois segue pra um processo de filtragem dos resultados, e como saída a recomendação em si.

Foi encontrado uma revisão sistemáticas (trabalho da **?**), que aborda métodos e modelos de implementação de um RSSE apresentando vários aspectos de SR em ES, principalmente no tipo de corpo de conhecimento aplicado a RSSE. Nessa revisão foi possível identificar algumas áreas da ES que 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 **??** a seguir.

Tabela 2.2: Sumário de técnicas de recomendação em cada domínio, Traduzido e Adaptado de **?**

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**

2.6 Trabalhos Relacionados

Até o momento não há trabalhos de sistema de recomendações para recomendar experimentos em ES, nem em LPS. Outros trabalhos relacionados mais próximos já foram apresentado na Seção 2.3 e não são objetos de evolução de experimento gerais em ES.

Encontramos alguns estudos que propuseram abordagens para representar formalmente dados sobre experimentos de SE.

A revisão da literatura mostrou que a maioria dos estudos focalizou a representação de todo o domínio do SE. Isso é muito complexo devido à quantidade de detalhes de cada campo. Em vez disso, nos concentramos primeiro em um campo de SE devido à nossa experiência em grupo.

Durante nossa revisão de literatura encontramos: (**?**) Ontologia para Experimentos Controlados em Engenharia de Software, (**?**) Empacotando Experimentos Controlados Usando uma Abordagem Evolutiva Baseada em Ontologia (S), (**?**) Uma Ontologia Fundamental para Apoiar Experimentos Científicos, (**?**) ODE: uma Ontologia para Projeto Numérico de Experimentos, (**?**) Uma Ontologia de Experimentos Científicos, (**?**) Desafios na Avaliação de Ontologia, (**?**) Extraindo Informações da Engenharia de Software Experimental papéis. No entanto, todos esses trabalhos não tratam experimentos de SPL.

O trabalho de Garcia et al. (**?**), Poveda-Villalón et al. (**?**) e Cruz et al. (**?**) se destaca em nosso contexto por propor e modelar ontologias específicas para experimentos em engenharia de software. O trabalho de (**?**) 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. (**?**) é uma evolução do trabalho de Garcia et al. (**?**), mas focado na evolução desta ontologia proposta. O trabalho de Cruz et al. (**?**) 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. (**?**) e Soldatova e King (**?**) tratam ontologias no contexto geral de experimentos. O trabalho de Blondet et al. (**?**) 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 (**?**) 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 (**?**) 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. (**?**) 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.

A revisão da literatura mostrou que a maioria dos estudos focalizou a representação de todo o domínio do SE. Isso é muito complexo devido à quantidade de detalhes de cada campo. Em vez disso, nos concentramos primeiro em um campo de SE devido à nossa experiência em grupo.

3

Uma Ontologia para Experimentos em LPS (SMartOntology)

Este tópico apresenta conceitos fundamentais sobre a proposta de ontologia SMartyOntology. Inicialmente foi realizado um processo de concepção do modelo em seguida foi desenvolvido o projeto desenvolvido se obter modelo. Com a finalidade de avaliação foi desenvolvido um exemplo de aplicação por meio de uma predição de recomendação qual artefato de LPS pode ser usado por um dado templete de experimento. Ao final desse capitulo foi realizado uma avaliação empírica do modelo, por meio de uma ferramenta que avalia pontos de falha do modelo.

3.1 Concepção

Com base na tipologia proposta por (**?**) definimos o tipo de ontologia proposto na Tabela **??**:

Ou seja, 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 e de domínio.

Foi aplicado o seguinte processo de elaboração da ontologia (**?**):

- Definição e estruturação dos termos por meio de classes;
- Estabelecimento de propriedades (atributos) inerentes ao conceito representado por um termo;

Tabela 3.1: Type of the Proposed Ontology

Approach	Rating	Description
As for the Mizoguchi function; Vanwelkenbuyse; Ikeda (1995)	Domain ontologies	Reusable in the domain, about concepts and thei activities and rules that
As for the degree of Uschold formalism; Gruninger (1996)	Semi-formal	Expressed in natural lan and structured way
As for the Jasper application; Uschold (1999)	Ontologies as specification	An ontology is created f for documentation and r development of software
As for the Haav structure; Lubi (2001)	Domain ontologies	Describe vocabulary rela such as medicine or auto
As for Van-Heijist content; Schreiber; Wielinga (2002)	Knowledge modeling ontologies	Specify knowledge conce semantically rich interna for use in the domain of
	Application ontologies	Contains the definitions in an application
	Domain ontologies	Express concepts that a given domain of knowle

- 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.

Inicialmente foi desenvolvido um grafo para o modelo inicial da ontologia SMartyOntology, principalmente para validar os termos por meio de classes, subclasses e os relacionamento entre elas.

A **??** representa este modelo inicial contendo a definição principal das classe de mais alto nível para o experimento, SPL Experiment, Documentation, Template, Evaluation, Discussion, Analysis, Execution e Planning. Em seguida evoluímos para possíveis subclasses delas, onde definimos as sub-classes em **??**:

A criação deste modelo inicial de ontologia se baseada no trabalho de mapeamento sistemático de experimentos em SPL (**?**), por meio de uma análise exploratória dos dados levantados nesse mapeamento. O mapeamento servir como guia de informações e metadados de experimentos.

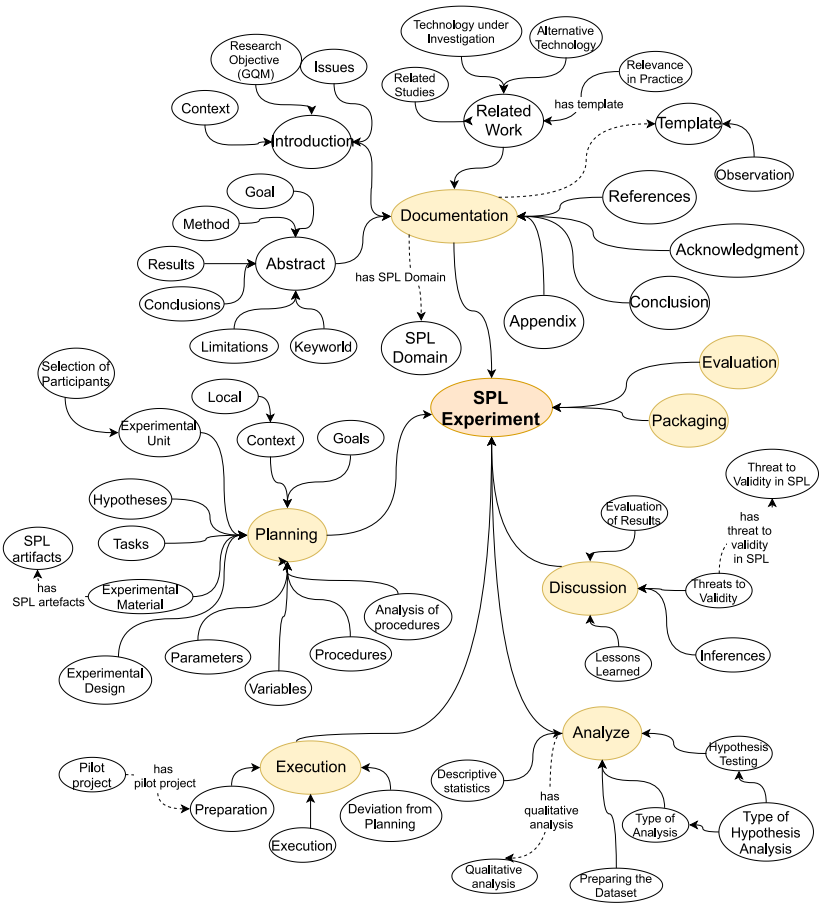

Figura 3.1: Initial graph of ontology.

O mapeamento sistemático se baseou no temple experimental de Wohlin, que traz 5 pilares para elaboração de ESE, são eles: Definição, Planejamento, Operação, Análise e Interpretação (**?**).

A **??**, apresenta uma modificação do modelo conceitual original, aplicando uma clusterização baseada nos pilares do Wohlin. Esta clusterização foi necessária para compreensão mais abstrata das relações entre os termos do domínio levantados no modelo conceitual original. Dessa forma foi possível validar as classes pai do grafo inicialmente proposto.

Em seguida, um diagrama de classes foi criado para uma representação mais formal da modelagem inicial. Nessa representação, a relação entre os termos (classes) e suas propriedades (atributos) ficou mais clara, o modelo TBox. 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 foi explícita a tipificação das propriedades, modelo ABox.

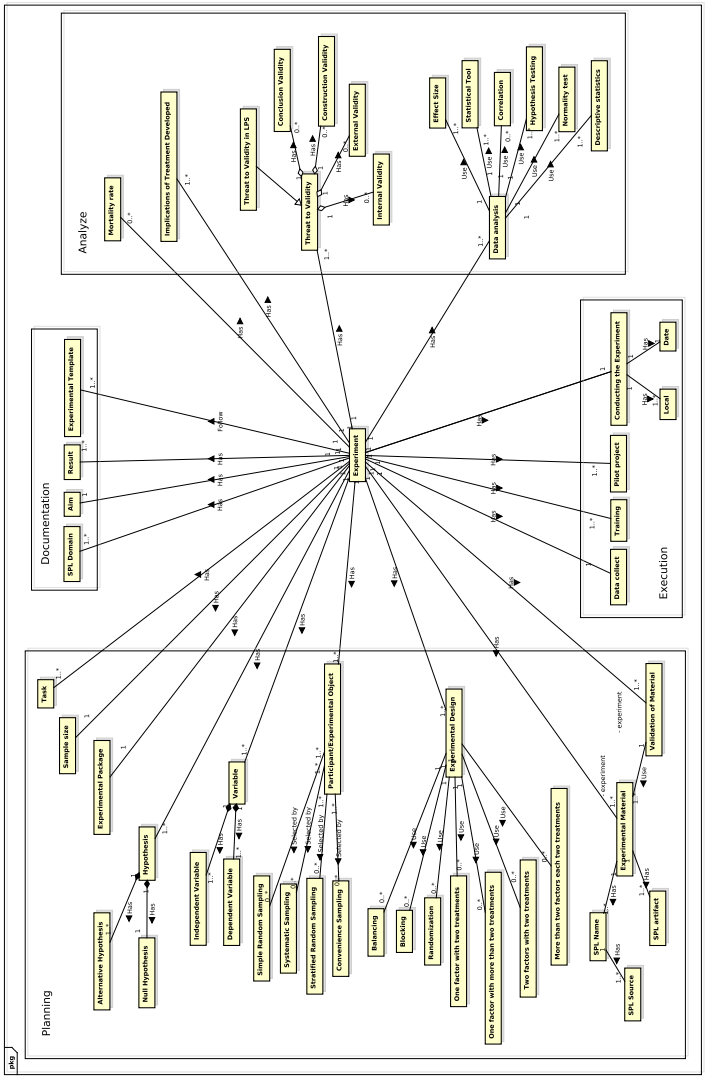

Figura 3.2: Conceptual Model Clustering.

A **??** apresenta diagrama de classe gerado para SMartyOntology.

Posteriormente, definimos usar a ferramenta Protégé para a construção oficial da ontologia. Esta ferramenta 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 (**?**).

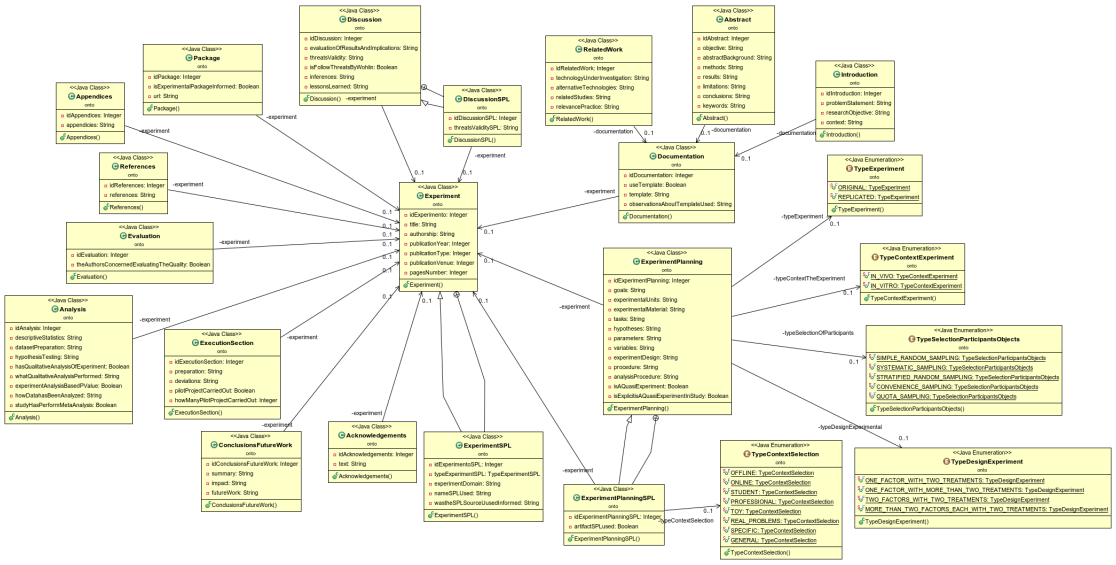

Figura 3.3: Class Diagram for proposal of the ontology model.

3.2 Projeto

Definimos a criação da ontologia usando OWL construindo as classes e subclasses para representar os elementos da ontologia.

3.2.1 Protegé

O padrão OWL foi usado no SMartyOntology para definir todos os elementos, classes e subclasses.

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

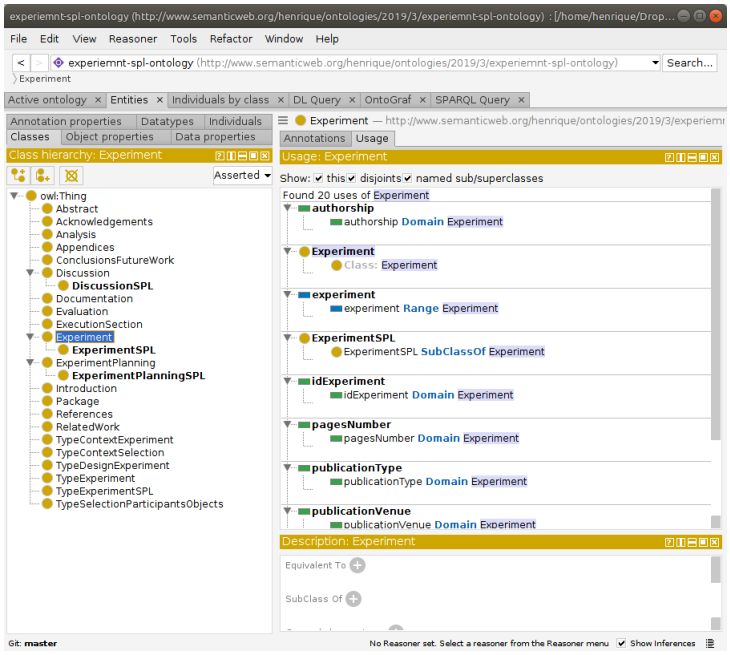

Figura 3.4: Definição da classe Experiment no Protégé.

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 apresenta a definição de propriedade de objeto para chamada ?documentation? dentro do modelo, neste momento a definição é composta pelo nome da propriedade e seus relacionamentos com classes, Domains e Ranges.

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 **??** apresenta a definição de propriedade de dado para chamada ?nameSPLUsed? dentro do modelo, neste momento a definição é composta pelo nome da propriedade e seus relacionamentos, Domains (uma classe) e Ranges (uma tipagem).

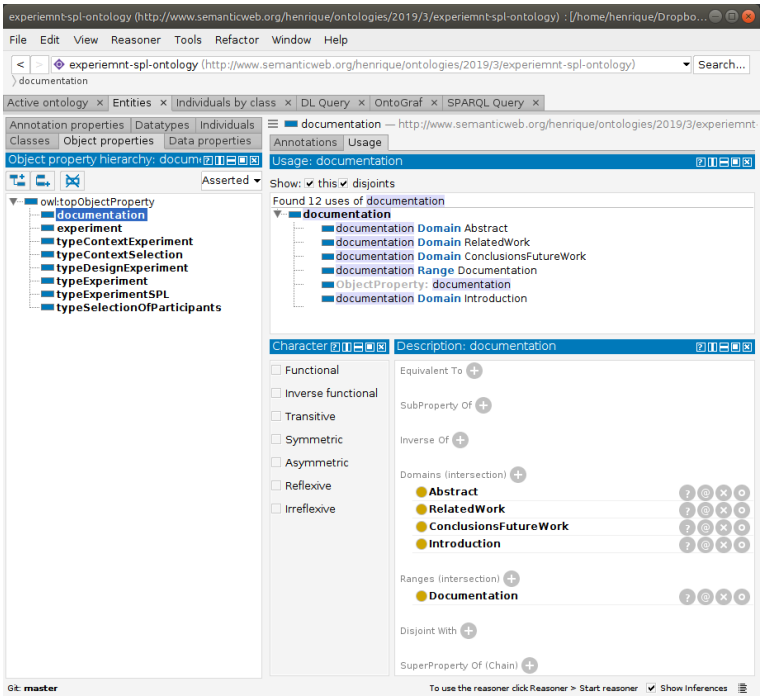

Figura 3.5: Definição da propriedade de objeto documentation no Protégé.

Artefato gerado pelo Protégé

Ao final, o Protégé gera um arquivo .owl contendo a definição do modelo. A **??** fornece uma visão geral do formato do grafo do projeto, contendo todas as classes e suas propriedades de objeto e propriedade de dados. Usamos a ferramenta WebVOWL (**?**) para gerar essa visão.

3.2.2 Pyton

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 174 indivíduos a serem inseridos em uma carga inicial, seriam mais de 16000 iterações manuais no Protégé.

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

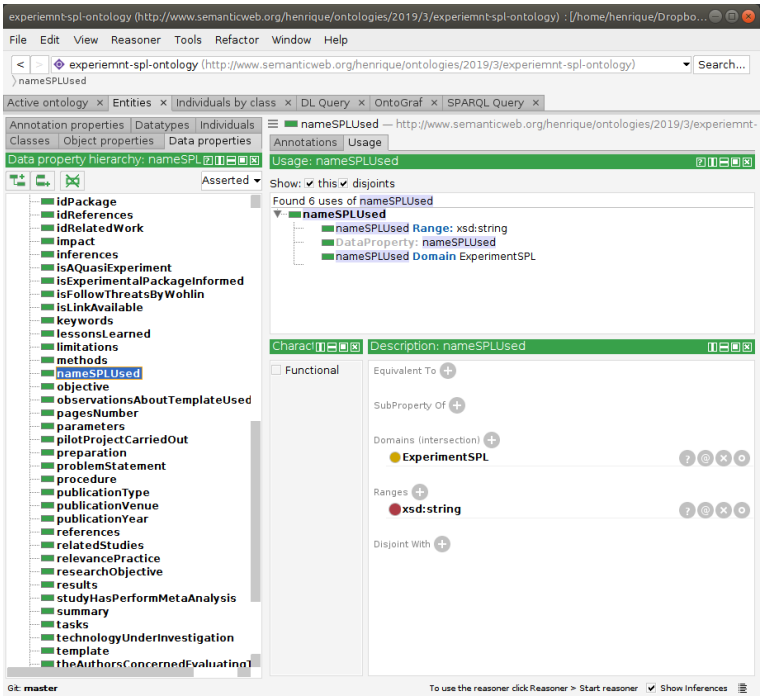

Figura 3.6: Definição da propriedade de dado nameSPLUsed no Protégé.

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).

Script

Para criação do script utilizamos outra ferramenta do Python chamada Jupyter Notebook (**?**) 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 (**?**), ela nos retorna um objeto do tipo Dataframe que chamamos de ?df?, 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 dado.

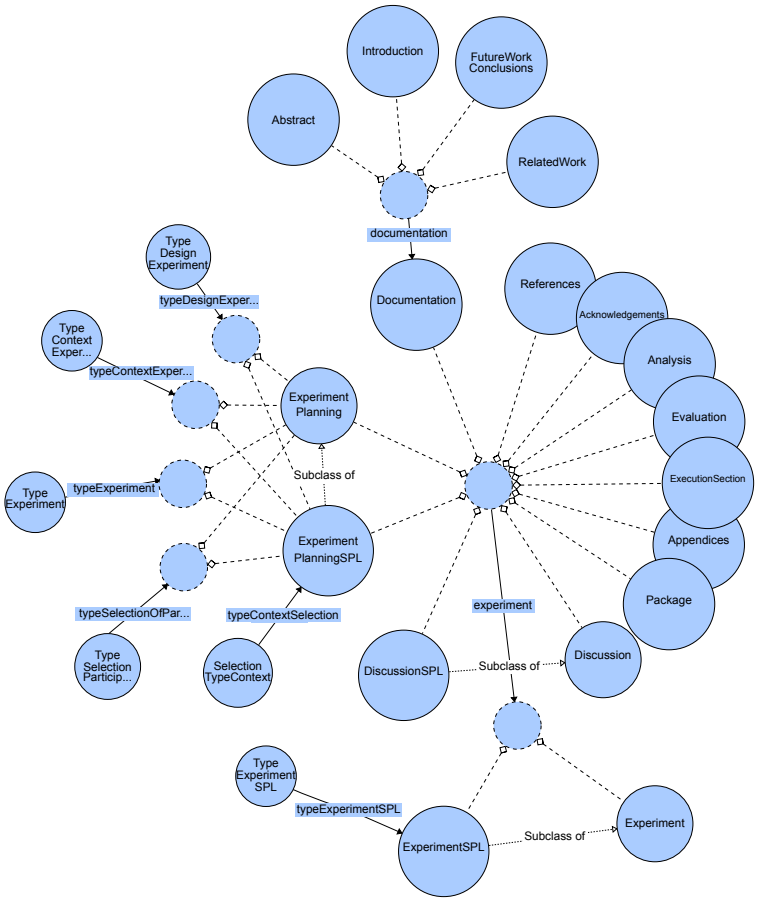

Figura 3.7: Grafo da modelagem da ontologia - gerado pelo WbVOWL.

Com relação ao Experimento 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 tecnology 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 approarch 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 expimental 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 fullfit 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 SMartyOntology, foi preciso manipular os dados da planilha com as seguintes operações.

Primeira operação, 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 SPL. Separamos o conjunto total de 174 artigos em artigos de experimentos que contêm informações explícitas de SPL (152) e artigos que não contêm informação explícitas de SPL (22), 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_SAMPI 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_ TWO_FACTORS_WITH_TWO_TREATMENTS, MORE_THAN_TWO_FACTORS_EACH_WI 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 fazemos a leitura do modelo da ontologia gerada pelo Protégé (arquivo .owl) por meio da biblioteca Owlready2, com isso temos um objeto chamado ?onto? 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 SPL chamado de df_spl e outro chamado df_exp. O primeiro laço sobre df_spl invoca a seguinte sequencia de métodos: registreExperimentSPL ¿ registreExperiment-PlanningSPL ¿ registreDiscussionSPL ¿ registerCommons. O segundo laço sobre df_exp invoca a seguinte sequencia de métodos: registreExperiment ¿ registreExperimentPlanning ¿ registreDiscussion ¿ registerCommons.

Listing 3.1: The first loop on `df_spl`

```
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: The second loop on `df_exp`

```
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.ExperimentPlanningSl

O método registreExperimentPlanning executa o método registreExperimentPlanning-Commons e retorna uma instância de indivíduo da ontologia da classe onto.ExperimentPlanning.

O método registreExperimentPlanningCommons recebe uma instância tanto de onto.ExperimentP 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 ?onto? 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 66,3 kB, e o arquivo populado contém 4,8 MB.

No final do script tem 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 **??**.

Listing 3.3: Stap validation

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

3.3 Exemplo de Aplicação

Foi criado também um exemplo de recomendação com a seguinte questão de recomendação, ?Qual artefato de SPL usar dos experimentos que usam a mesma template experimental mais recorrente?. Com base nesta questão foi criado um script Python utilizando as bibliotecas Pandas e OwlReady2, para responder a mesma.

O script começa carregando a ontologia preenchida e, em seguida, executamos a consulta SPARQL **??** para retornar os agrupados pela template experimental, estes indivíduos são do tipo Documentation. Na sequência, filtramos da lista inicial de indivíduos que usam modelos, aqueles que usam o modelo mais recorrente. Com esses indivíduos em mãos, buscamos seu relacionamento com indivíduos do tipo ExperimentPlanningSPL para extrair informações de artefatos de SPL. Assim, podemos recomendar qual artefato de SPL, com base no artefato mais usado desses indivíduos filtrados.

O script começa carregando a ontologia povoada, e em seguida extrai os indivíduos que usam template experimental, estes indivíduos são do tipo Documentation. Atribuímos este resultado a um objeto do tipo Pandas para executar a função .values_counts() que retorna o agrupamento somado das templates mais usadas. Na sequência, filtramos da lista inicial dos indivíduos que usam templates, aqueles que usam a template mais recorrente. Com estes indivíduos em mãos, buscamos o relacionamento deles com os indivíduos do tipo ExperimentPlanningSPL para extrair a informação dos artefatos de SPL. Então podemos recomendar qual artefato de SPL, com base no artefato mais usado destes individuos filtrados.

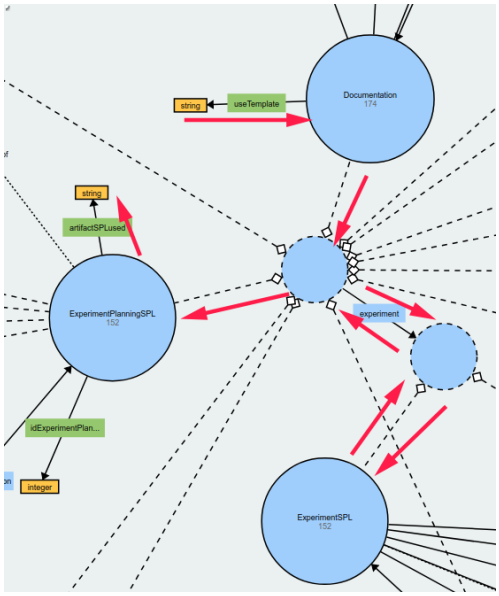

Figura 3.8: Graph Path to Recommendation Example.

O grafo da **??** mostra o caminho percorrido entre as classes e propriedades de objetos e dados para chegar no artefato de SPL a ser recomendado.

O exemplo de recomendação apresenta como podemos criar mecanismos de inferência no modelo de ontologia proposto neste trabalho. Foi possível avaliar que com algumas consultas e verificações é possível extrair informações e meta informações sobre experimentos em SPL.

Este exemplo percorre três classes (Documentation, ExperimentSPL e Experiment-PlanningSPL) de 24 classes, duas propriedades de objetos (documentation e experimet) de 8 propriedades e três propriedades de dados (useTemplate, template e artifactSPLused) de 87 propriedades de dados do modelo proposto. O cálculo . estima que neste exemplo estamos usando 0,1

O exemplo apresenta como podemos criar mecanismos de inferência no modelo de ontologia proposto neste trabalho. Foi possível avaliar que com algumas consultas e verificações é possível extrair informações e meta informações sobre experimentos em SPL.

Este exemplo percorre três classes (Documentation, ExperimentSPL e Experiment-PlanningSPL) de 24 classes, duas propriedades de objetos (documentation e experimet) de 8 propriedades e três propriedades de dados (useTemplate, template e artifactSPLused) de 87 propriedades de dados do modelo proposto. O cálculo . estima que neste exemplo estamos usando 0,1% da capacidade de resposta que o modelo possa responder.

No caso do exemplo de recomendação a primeira parte do retorna a template mais recorrente. O resultado é a saída do script: ?Template more frequency is Wohlin? seguida da Fig. que representa os dados da Tabela.

E segunda parte encontra qual artefato de SPL dos indivíduos que usam a template encontrada na primeira etapa. o resultado é a saída do script: ?Artifact more used is the "test cases"to template: Wohlin? seguido da Fig.

3.4 Avaliação Empírica

Para avaliação da ontologia seguimos duas estratégias, uma para avaliar possíveis armadilhas e falhas no modelo de ontologia proposto.

Usamos 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].

- 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;
- 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 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. 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 para encontrar outras armadilhas, por exemplo, ?Definir relações simétricas que não possuem o mesmo domínio e alcance?.

A ferramenta elenca 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. apresenta o resumo do resultado ao rodar nosso modelo de ontologia proposto na ferramenta OOPS!.

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 [referencia 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 P12, a ontologia carece de informações sobre propriedades equivalentes (owl: equivalentProperty) nos casos de relacionamentos e / ou atributos duplicados. Os seguintes atributos podem ser definidos como equivalentes: wasTheSPLSourceUsedInformed e wastheSPLSourceUsedInformed.

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.

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

3.4.1 Perspectivas de Melhorias

Dada a análise da ferramenta OOPS! o próximo passo será corrigir as armadilhas apresentada na avaliação e reavaliar a ontologia de modo geral, principalmente os pontos que a ferramenta não está automatizada.

Tabela 3.2: Classes and Sub-classes of ontology prototype

Class	Sub-class	Sub-class
Documentation	Abstract	Keyword
		Limitations
		Conclusions
		Results
		Method
		Goal
	Introduction	Context
		Research Objective (GQM)
	Related Work	Issue Issues
		Related Studies
		Technology under Investigation
		Alternative Technology
		Relevance in Practice
	Template	Observation
	References	
	Acknowledgment	
	Conclusion	
	Appendix	
Planning	SPL Domain	
	Context	Local
	Experimental Unit	Selection of Participants
	Experimental Material	SPL artifacts
	Experimental Design	
	Goals	
	Hypotheses	
	Tasks	
	Parameters	
	Variables	
	Procedures	
	Analysis of procedures	
	Preparation	
Execution	Execution	Pilot
	Deviation from Planning	
Analyze	Type of Analysis	Type of Hypothesis Analysis
	Hypothesis Testing	
	Descriptive statistics	
	Qualitative analysis	
	Preparing the Dataset	
Discussion	Threats to Validity	Threat to Validity in SPL
	Evaluation of Results	
	Inferences	
Packaging		
Evaluation		

Tabela 3.3: Ontology Design - Classes and Properties modeling
to — p1.4cm — p15.6cm —

Element	Definition
Classes	Abstract, Acknowledgments, Analysis, Appendices, ConclusionsFutureWork, Discussion, DiscussionSPL, Documentation, Evaluation, ExecutionSection, Experiment, ExperimentSPL, ExperimentPlanning, ExperimentPlanningSPL, Introduction, Package, References, RelatatedWork, TypeContextExperiment, TypeContextSelection, TypeDesignExperiment, TypeEsperiment, TypeEsperimentSPL, TypeSelectioParticipantObjects
Object Properties	documentation, experiment, typeContextxperiment, typeContextSelection, typeDesignExperiment, typeExperiment, typeExperimentSPL, typeSelectionOfParticipants
Data Properties	idExperiment, title, authorship, publicationYear, publicationType, publicationVenue, pagesNumber, idExperimentSPL, nameSPLUsed, wasTheSPLSourceUsedInformed, idDocumentation, useTemplate, template, observationsAboutTemplateUsed, idAbstract, objective, abstractBackground, methods, results, limitations, conclusions, keywords, idIntroduction, problemStatement, researchObjective, context, idRelatedWork, technologyUnderInvestigation, alternativeTechnologies, relatedStudies, relevancePractice, idConclusionsFutureWork, summary, impact, futureWork, idExperimentPlanning, goals, experimentalUnits, experimentalMaterial, tasks, hypotheses, parameters, variables, experimentDesign, procedureProcedure, explicitQuesiExperimentInStudy, isAQuasiExperiment, idExperimentPlanningSPL, artifactSPLused, idExecutionSection, preparation, deviations, pilotProjectCarriedOut, howManyPilotProjectCarriedOut, idAnalysis, descriptiveStatistics, datasetPreparation, hyp othesisTesting, whatQualitativeAnalysisPerformed, howDatahasBeenAnalyzed, experimentAnalysisBasedPValue, hasQualitativeAnalysisOfExperiment, studyHasPerformMetaAnalysis, idDiscussion, evaluationOfResultsAndImplications, inferences, lessonsLearned, threatsValidity, isFollowThreatsByWohlin, idDiscussionSPL, threatsValiditySPL, idAcknowledgements, acknowledgments, idReferences, references, idAppendices, appendicies, idEvaluation, theAuthorsConcernedEvaluatingTheQuality, idPackage, isExperimentalPackageInformed, url, isLinkAvailable

Tabela 3.4: Result Summary of the OOPS! evaluation!

Pitfall	Description	Critical Level
P08	Missing annotations in 119 cases	Minor
P10	Missing disjointness on ontology*	Important
P12	Equivalent properties not explicitly declared in 1 case	Important
P13	Inverse relationships not explicitly declared in 8 cases	Minor
P19	Defining multiple domains or ranges in properties in 6 cases	Critical
P41	No license declared on ontology	Important

4

Avaliação da SMartOntology

Este tópico apresenta a avaliação aplicada a SMartOntology.

4.1 Considerações Iniciais

4.2 Objetivos

4.3 Metodologia

4.4 Avaliação

4.5 Considerações Finais

5

Proposta de Melhorias e Trabalhos Futuros para SMartOntology

Este tópico apresenta propostas de melhorias e trabalhos futuros a serem aplicadas a SMartOntology.

5.1 Considerações Iniciais

5.2 Considerações Finais

6

Conclusão

A realização de experimentos de SE no SPL traz informações mais relevantes para fornecer evidências de uma teoria para o mundo real. A capacidade de entender, estudar e replicar experimentos torna esse método ainda mais rico. Ao organizar os dados e informações sobre os experimentos de SE no SPL, acreditamos que essa tarefa é mais fácil. Usando as tecnologias adjacentes, tais como formalizar o conhecimento através de ontologias e inferências para gerar dados de informação personalizada através de um sistema de recomendação.

Neste trabalho, foi desenvolvida uma proposta para um modelo de ontologia (SMartyOntology) com o objetivo de organizar e estruturar o conhecimento adquirido sobre experimentos de SE em SPL. Este conhecimento foi previamente levantado em um trabalho de dissertação do nosso grupo de pesquisa, onde foram relatados mais de 200 artigos que relatam experimentos em SPL. Com este levantamento foi possível criar um modelo de ontologia para representar o conhecimento sobre experimentos em NPS, inserindo os dados dos artigos neste modelo, denominamos esta fase de inserção dos indivíduos na ontologia. Desta forma, estruturamos o TBox e o Abox para a SMartyOntology. Com a composição da ontologia podemos fazer uma breve avaliação sobre algumas armadilhas que o modelo de ontologia poderia ter, utilizamos a ferramenta OOPS! para esta avaliação. Também foi possível criar um exemplo simples de recomendação, fazendo a inferência das informações para a SMartyOntology.

A SMartyOntology se destaca por levar em conta um domínio específico de experimentos de SE. SPL são construídos através de um domínio de aplicação, semelhanças, avaliações do núcleo e variabilidades, que distingue um produto do outro dentro da família de produtos, todas essas características estão presentes na ontologia. No entanto, a

SMartyOntology pode ser facilmente estendida a outros domínios de experimentos de SE, uma vez que todas as classes dentro da ontologia que lidam com SPL são subclasses de representações de experimentos em geral.

Os principais objetivos deste trabalho, estão relacionados a propor um sistema de recomendação que, possa gerar processos e diretrizes para realização de experimentos em LPS. Para foi realizado um estudo aprofundado nos conceitos de Sistemas de Recomendação em ES e modelos de Ontologias para representação dos dados levantados sobre a qualidade dos experimentos em LPS, encontrados no trabalho do grupo.

Portanto, acredita-se que após a realização do projeto de software, será possível desenvolver o sistema de recomendação para que os usuários que interagirem com este sistema, e possa receber recomendações de processos e diretrizes para suas pesquisas experimentais em LPS.

Como trabalho futuro, identificamos na avaliação vários pontos de melhoria na modelagem da ontologia, esses ajustes devem ser feitos para padronizar o modelo com o objetivo de possibilitar a extensão e divulgação do mesmo.