UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”
Programa de Pós-Graduação em Ciência da Computação

JORGE LUÍS GREGÓRIO

ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE A PARTIR DE ONTOLOGIAS REPRESENTATIVAS DE MODELOS DE PROCESSOS DE NEGÓCIO

Rio Claro - SP
2019

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
Programa de Pós-Graduação em Ciência da Computação

JORGE LUÍS GREGÓRIO

ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE A PARTIR DE ONTOLOGIAS REPRESENTATIVAS DE MODELOS DE PROCESSOS DE NEGÓCIO

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, área de concentração em Computação Aplicada, linha de Pesquisa Sistemas de Informação, da Universidade Estadual Paulista "Júlio de Mesquita Filho".

Orientadora: Profa. Dra. Hilda Carvalho de Oliveira
Coorientadora: Profa. Dra. Simone das Graças Domingues Prado

Rio Claro - SP
2019

G821e

Gregório, Jorge Luís
Especificação de requisitos de software a partir de ontologias representativas de modelos de processos de negócio / Jorge Luís Gregório. -- Rio Claro, 2019
149 p. : il., tabs. + 1 CD-ROM

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Instituto de Geociências e Ciências Exatas, Rio Claro
Orientadora: Hilda Carvalho de Oliveira
Coorientadora: Simone das Graças Domingues Prado

1. Computação. 2. Modelagem de Informação. 3. Ontologias. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Geociências e Ciências Exatas, Rio Claro. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

JORGE LUÍS GREGÓRIO

ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE A PARTIR DE ONTOLOGIAS REPRESENTATIVAS DE MODELOS DE PROCESSOS DE NEGÓCIO

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação, junto ao Programa de Pós-Graduação em Ciência da Computação, área de concentração em Computação Aplicada, linha de Pesquisa Sistemas de Informação, da Universidade Estadual Paulista "Júlio de Mesquita Filho".

Comissão Examinadora

Profa. Dra. Hilda Carvalho de Oliveira
UNESP - Rio Claro
Orientadora

Prof. Dr. Ivan Luiz Marques Ricarte
UNICAMP (FT) - Limeira

Prof. Dr. Rogério Eduardo Garcia
UNESP - Presidente Prudente

Conceito: Aprovado

Rio Claro - SP

5 de setembro de 2019

Agradecimentos

Agradeço a Deus, ao qual entrego toda minha vida. Apesar de minhas falhas e limitações, acredito que Ele sempre está comigo.

Agradeço à minha esposa, Patrícia, pelo companheirismo, apoio, incentivo e paciência durante a elaboração deste trabalho.

Ao meu filho, Emanuel, que com apenas 4 anos, me motivou e me inspirou com seu singelo sorriso e inocência.

Ao meu pai, Francisco, agradeço o apoio e incentivo em toda a minha vida pessoal e profissional.

Aos pais de minha esposa, Iracema e Natalino, que também são meus pais e me ajudaram muito nessa caminhada, principalmente ao cuidar do meu filho enquanto eu estava estudando.

Agradeço à minha orientadora, Profa. Dra. Hilda Carvalho de Oliveira, pela paciência, pelo seu amor à profissão e por ter acreditado e investido seu tempo em mim. Suas orientações, dicas e cobranças foram muito valiosas em todos os aspectos, não apenas no acadêmico. Muito obrigado!

Agradeço à minha coorientadora, Profa. Dra. Simone das Graças Domingues, pelas construtivas discussões sobre ontologias.

Agradeço ao Prof. Dr. Norian Marranghello, pelo incentivo para reiniciar os estudos no Mestrado.

Aos colegas de Mestrado, principalmente Fabiana Pupin Masson Caravieri, a quem agradeço pela amizade, companheirismo e pela interação durante as disciplinas e outros trabalhos acadêmicos.

A Lukas Riehl Figueiredo, pelas reuniões, discussões e dicas sobre ontologias e modelos de processos de negócio.

A Fernando Aparecido Nogueira, pelas construtivas discussões sobre modelos de processos de negócio e Engenharia de Requisitos.

Agradeço à Direção e ao corpo docente da Faculdade de Tecnologia “Prof. José Camargo” (Fatec Jales) e da Escola Técnica Estadual “Dr. José Luiz Viana Coutinho”. Obrigado a todos pelo incentivo, amizade e companheirismo.

“Porém, aqueles que esperam no SENHOR renovarão as suas forças. Eles se elevarão com asas como águias, eles correrão e não estarão cansados, e eles caminharão e não desfalecerão.”

(BÍBLIA, Isaías, 40,31).

RESUMO

O documento de especificação de requisitos de software é parte essencial de um projeto de software, pois contém representações e modelos que orientam todas as demais fases do desenvolvimento. A elaboração desse documento deve ser colaborativa, com interação entre as equipes de negócio e de desenvolvimento do software. Entretanto, essas duas equipes operam em diferentes níveis de abstração e usam diferentes modelos e notações para a compreensão do domínio, causando problemas de comunicação e compartilhamento de conhecimento. De acordo com a abordagem orientada a modelos MDA (*Model-Driven Architecture*), os requisitos podem ser extraídos a partir de modelos com maior nível de abstração, como é o caso dos modelos de processos de negócio em BPMN (*Business Process Model and Notation*). Apesar da notação BPMN ser suficientemente expressiva para diferentes perfis de usuários, ela oferece uma visão de processos e não de conhecimento. Entretanto, esses modelos podem ser mapeados para ontologias em OWL (*Ontology Web Language*), formalizando o conhecimento de maneira compartilhável e propiciando inferência de novos conhecimentos. Considerando que as ontologias em OWL são legíveis por máquina, é possível o uso de técnicas e ferramentas para se extrair informações das ontologias de maneira automatizada. Assim, este trabalho apresenta um processo sistemático para a extração de requisitos de software a partir de ontologias representativas de modelos de processos de negócio, na notação BPMN v2.0. As ontologias consideradas neste trabalho são geradas por uma versão estendida do sistema PM2ONTO (*Process Model to Ontology*). Os dados são extraídos de maneira automática pelo sistema OnToSRS que gera o documento de especificação de requisitos de software, segundo o padrão ISO/IEC/IEEE 29148:2018, com a finalidade de orientar a automação dos processos de negócio. Esse documento inclui requisitos funcionais, não funcionais, regras de negócio, entre outras informações, bem como diagramas de caso de uso e de classes. O trabalho apresenta estudos de caso que mostram a viabilidade do processo definido. De modo geral, este trabalho mostra que as ontologias podem representar modelos de processos de negócio, possibilitando a extração e a complementação de informações, além de permitir a integração entre diferentes modelos de processos de negócio.

Palavras-chave: Modelos de processos de negócio. BPMN. Ontologias. OWL. SPARQL. Sistema PM2ONTO. Engenharia de Requisitos. Especificação de requisitos de software. Sistema OnToSRS.

ABSTRACT

The software *requirements specification document is an essential part of a software project because it contains representations and models that guide all other phases of development. The preparation of this document should be collaborative, with interaction between the business and software development teams. However, these two teams operate at different levels of abstraction and use different models and notations to understand the domain, causing communication and knowledge sharing problems. According to the Model-Driven Architecture (MDA) approach, requirements can be extracted from models with high level of abstraction, such as Business Process Model and Notation (BPMN) business process models. Although BPMN notation is sufficiently expressive for different user profiles, it offers a process view, not a knowledge view. However, these models can be mapped to Ontology Web Language (OWL) ontologies, formalizing knowledge in a shareable manner and providing inference to new knowledge. Since OWL ontologies are machine-readable, it is possible to use techniques and tools to extract information from ontologies automatically. Thus, this paper presents a systematic process for extracting software requirements from representative ontologies of business process models, in the BPMN v2.0 notation. The ontologies considered in this paper are generated by an extended version of the PM2ONTO (Process Model to Ontology) system. The data is automatically extracted by the OnToSRS system that generates the software requirements specification document according to ISO/IEC/IEEE 29148:2018 to guide business process automation. This document includes functional requirements, non-functional requirements, business rules, and other information, as well as use case and class diagrams. The paper presents case studies that show the viability of the defined process. In general, this paper shows that ontologies can represent business process models, enabling the extraction and complementation of information, as well as allowing the integration between different business process models.*

Keywords: *Business Process Models. BPMN. Ontologies. OWL. SPARQL. PM2ONTO system. Requirements Engineering. Software requirements specification. OnToSRS system.*

LISTA DE FIGURAS

Página

LISTA DE TABELAS

Página

LISTA DE ABREVIATURAS E SIGLAS

5W1H	*Who, What, Where, When, Why, How*
ABPMP	*Association of Business Process Management Professionals*
API	*Application Programming Interface*
ATL	*ATLAS Transformation Language*
BABOK	*Business Analysis Body of Knowledge*
BPM	*Business Process Management*
BPMI	*Business Process Management Initiative*
BPMN	*Business Process Management and Notation*
BPMN2UC	*BPMN to Use Case*
CMMN	*Case Management Model and Notation*
CSS	*Cascade Style Sheets*
DFD	Diagrama de fluxo de dados
DevOps	*Development and Operations*
DMN	*Decision Model and Notation*
EMF	*Eclipse Modeling Framework*
ERS	*Especificação de Requisitos de Software*
HN	Heurística de negócio
HNa	Heurística de negócios para atividades
HNd	Heurística de negócios para artefatos
HNe	Heurística de negócios para eventos
HNg	Heurística de negócios para fluxo de decisão
HR	Heurística de requisitos
HTML	*Hypertext Markup Language*
IEC	*International Electrotechnical Commission*
IEEE	*Institute of Electrical and Electronics Engineers*

IRI	*Internationalized Resource Identificator*
ISERN	*International Software Engineering Requirement Research Network*
ISO	*International Organization for Standardization*
ITIL	*Information Technology Infrastructure Library*
MDA	*Model-Driven Architecture*
NaPiRE	*Naming the Pain in Requirements Engineering*
OCL	*Object Constraint Language*
OMG	*Object Management Group*
OnToSRS	*Ontology to Software Requirements Specification*
OWL	*Ontology Web Language*
PIM	*Platform-Independent Model*
PM2ONTO	*Process Model to Ontology*
PSM	*Platform-Specific Model*
RDF	*Resource Description Framework*
RDF-S	*Resource Description Framework Schema*
REA	*Resource, Event, Agent*
REBOK	*Requirements Engineering Body of Knowledge*
REO	*Requirements Engineering Ontology*
RM	Regra de mapeamento
RNF	Requisito não-funcional
SKOS	*Simple Knowledge Organization System*
SPARQL	*SPARQL Protocol and RDF Query Language*
SRPD	*Software Requirements from Process Definitions*
SRS	*Software Requirements Specification'*
SVG	*Scalable Vector Graphics*
SWEBOK	*Software Engineering Body of Knowledge*
TI	Tecnologia da Informação
TIC	Tecnologias da Informação e Comunicação
UML	*Unified Modeling Language*

W3C	*World Wide Web Consortium*
WfMC	*Workflow Management Coalition*
WWW	*World Wide Web*
XML	*Extensible Markup Language*
XPDL	*XML Process Definition Language*
XSD	*XML Schema Definition*

SUMÁRIO

Página

1 INTRODUÇÃO

As dificuldades de compreensão do negócio pela equipe de Tecnologias da Informação e Comunicação (TIC) durante o levantamento de requisitos podem trazer inúmeras incertezas aos processos da Engenharia de Requisitos (FERNÁNDEZ *et al*., 2017). Diversas soluções têm sido propostas para aproximar as equipes de negócio das equipes de desenvolvimento de software, de modo a produzir sistemas mais aderentes às necessidades do cliente. Entretanto, aproximar essas equipes não é algo simples, visto que operam em diferentes níveis de abstração, usando diferentes técnicas, linguagens e notações para a compreensão e descrição do domínio.

Pesquisas realizadas no programa NaPiRE (*Naming the Pain in Requirements Engineering),* apoiadas pela comunidade internacional ISERN (*International Software Engineering Requirement Research Network*), mostraram que os maiores problemas da Engenharia de Requisitos estão relacionados à lacuna de conhecimento existente entre os profissionais do negócio e de TIC (FERNÁNDEZ *et al*., 2017). Portanto, é necessária a formalização do conhecimento corporativo, de maneira que possam ser inteligíveis e compartilháveis entre as equipes multidisciplinares envolvidas em um projeto de software.

Convém destacar que a integração e a colaboração entre equipes multidisciplinares estão no centro das discussões em torno dos novos desafios da Engenharia de Software Contínua. Diante da necessidade de corrigir erros e atualizar funcionalidades em aplicações on-line, evitando indisponibilidade do serviço, a Engenharia de Software Contínua envolve a automação na construção, teste e implantação do software, substituindo os ciclos de lançamento por entregas contínuas (DITTRICH *et al*., 2018). Nesse contexto, a filosofia DevOps (Desenvolvimento e Operações) surge como um paradigma que envolve princípios e práticas para integrar as equipes de desenvolvimento e operações, objetivando entregas rápidas e contínuas, enquanto garante a qualidade (CHUNG, 2017).

Atualmente, muitas organizações vêm utilizando modelos de processos de negócios

com o objetivo de formalizar o conhecimento corporativo, além de analisar seus processos para prover melhorias e adaptações às constantes mudanças do mercado. Esses modelos possibilitam organização, documentação e análise de processos para o desenvolvimento de produtos e serviços de ponta-a-ponta na organização, executados por diferentes setores, usando diferentes recursos. Observa-se que a modelagem dos processos é o ponto primário da abordagem de gerenciamento de processos de negócio, conhecida como BPM (*Business Process Management*). Essa abordagem estabelece a governança dos processos sob uma perspectiva integrada e transversal entre os setores da organização (ABPMP BRAZIL, 2013).

A modelagem de processos de negócio objetiva a criação de uma representação precisa, inteligível e compartilhada dos processos existentes ou propostos para a organização (ABPMP BRAZIL, 2013; CORREIA; ABREU, 2012). Para isso, podem ser usadas diferentes linguagens e notações, como: fluxogramas, Diagramas de Fluxos de Dados (DFDs), redes de Petri, diagrama de atividades em UML (*Unified Modeling Language*), etc. Nesse contexto, a notação BPMN (*Business Process Model and Notation*) tem se destacado devido à sua expressividade e por ter uma abrangente especificação, elaborada pelo consórcio OMG (*Object Management Group*). A especificação de BPMN recomenda a documentação textual dos elementos gráficos da notação em um modelo de processos de negócio, o que colabora com a representação do conhecimento expresso no modelo (OMG, 2013). Entretanto, não determina uma linguagem, notação ou padrão para isso. É importante ressaltar que a documentação textual também contribui para a expressividade do modelo para outros perfis de usuários, como a equipe de TIC, por exemplo, incluindo engenheiros de software. Além disso, os modelos em BPMN podem ser exportados para o formato XPDL (*XML Process Definition Language*), possibilitando transformação de modelos e a integração com diferentes sistemas de software.

Extrair requisitos de software a partir de modelos de processos de negócio em BPMN tem motivado diversas pesquisas, visando maior aderência do software desenvolvido ao ambiente organizacional e a mitigação dos problemas destacados por Fernández *et al*. (2017). A ideia é aproximar as equipes de negócio e de TIC, apoiando a construção do documento de especificação de requisitos de software (NOGUEIRA, 2017). Assim, têm sido desenvolvidas metodologias e técnicas, de modo manual ou automatizado, nos últimos anos. Geralmente, utiliza-se o conhecimento implícito e explícito nos modelos de processos de negócio para auxiliar o conjunto de requisitos de software extraídos. Portanto, é possível representar as funcionalidades do software a partir das informações e conhecimentos do negócio, que inclui o vocabulário usado no domínio.

Nesse contexto, podem ser mencionados os seguintes trabalhos: Bouzidi *et al*.

(2017), Leshob (2016), Nogueira (2017) e Park *et al*. (2017). De modo geral, esses trabalhos focam nas funcionalidades e no comportamento do software, ao invés da tecnologia de implementação – ideia expressa no padrão MDA (*Model-Driven Architecture*), especificado pelo consórcio OMG (2014). Nesse sentido, o mapeamento entre modelos independentes de plataforma, conhecido como PIM para PIM (*Plataform-Independent Model*), é recomendável quando os modelos precisam ser aprimorados ou especializados durante o ciclo de vida do processo de desenvolvimento de software e estão relacionados ao refinamento de modelos (MIKZA *et al*., 2012).

Por outro lado, as ontologias também podem ser usadas para representar o conhecimento corporativo, tornando-o compartilhável e inteligível em diversos níveis e contextos, indo além da visão de processos fornecida pela notação BPMN. Diante das características de expressividade, compartilhamento e, principalmente, considerando-se a necessidade da formalização do conhecimento para o desenvolvimento de software, as ontologias têm sido um importante objeto de estudo no campo da Engenharia de Software (DERMEVAL *et al*., 2016).

Considerando que os modelos de processos de negócio em BPMN possuem conhecimento implícito (documentação textual) e explícito (parte gráfica), é possível construir ontologias a partir desses modelos, de modo a formalizar o conhecimento organizacional. Isso permite inferir novos conhecimentos e extrair novas informações úteis aos mais diversos contextos, como, por exemplo, para a Engenharia de Requisitos. Alguns trabalhos encontrados na literatura geram ontologias a partir de modelos de processos de negócio, como Haller *et al*. (2008), Di Martino *et al*. (2015), Ternai, Török e Varga (2016) e Figueiredo (2018).

De modo geral, as ontologias contribuem para os ambientes organizacionais em diversos aspectos. Entre eles, pode-se mencionar: a) formalização e compartilhamento do conhecimento do domínio, inclusive de um vocabulário comum entre os vários setores da empresa; b) inferência de novos conhecimentos; c) recuperação de informações estruturadas; d) integração com diferentes ontologias; e) complementação de informações (BARTUSSEK *et al*., 2018).

Na Engenharia de Software, as ontologias podem contribuir de diferentes maneiras. Em trabalhos como os de Saito, Iimura e Aoyama (2015) e de Avdeenko e Pustovalova (2015), as ontologias são apresentadas como ferramentas de apoio aos processos da Engenharia de Requisitos, propondo a formalização do domínio. Por outro lado, Shunxin e Leijun (2010) extraem requisitos de software de ontologias, embora manualmente. Por sua vez, Mohamed, Ellatif e Farhan (2017) propõem o uso de ontologias para a criação de

mapas conceituais que apoiem a compreensão do domínio.

Frente ao exposto, a *Seção 1.1* apresenta os objetivos deste trabalho e a *Seção 1.2* apresenta a metodologia usada no seu desenvolvimento. A *Seção 1.3*, por sua vez, mostra como estão organizados os demais capítulos.

1.1 OBJETIVOS DO TRABALHO

O principal objetivo deste trabalho é desenvolver um processo sistemático para a especificação de requisitos de software a partir de ontologias representativas de modelos de processos de negócio em BPMN. Esse processo é automatizado por meio de um sistema denominado OnToSRS (*Ontology to Software Requirements Specification*), que foi implementado usando a plataforma Java SE (*Standard Edition*).

Um documento denominado "Especificação de Requisitos de Software" (ERS) é gerado em conformidade com as orientações do padrão ISO/IEC/IEEE 29148:2018 (ISO, 2018). Esse documento contempla informações úteis para os times de desenvolvimento de software, como, por exemplo: atores, requisitos funcionais e não funcionais, regras de negócios, entre outras. Além disso, contém diagramas de caso de uso e de classe, segundo a linguagem UML, que representam importantes aspectos comportamentais e estruturais do software a ser modelado. Para a geração desses diagramas, foi usada a biblioteca pública PlantUML e foram definidos algoritmos que processam dados recuperados da ontologia.

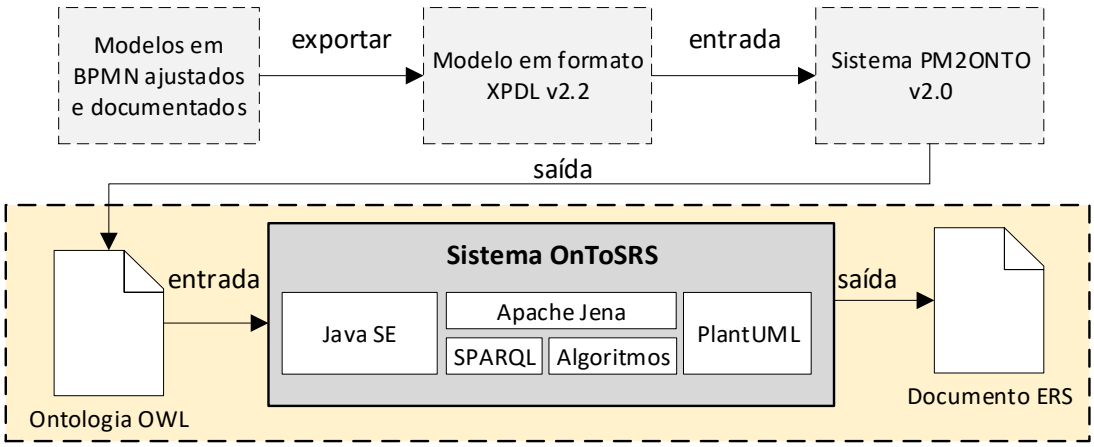
As ontologias consideradas neste trabalho são geradas pelo sistema PM2ONTO, de autoria de Figueiredo (2018), que constrói ontologias em OWL (*Ontology Web Language*) a partir de modelos de processos de negócio em BPMN v2.0, usando um metamodelo definido pelo autor. Para isso, os modelos devem satisfazer boas práticas de modelagem definidas quanto à modelagem gráfica e documentação. Como o sistema PM2ONTO é de domínio público (*freeware*) e tem código-fonte aberto, foi possível, para este trabalho, fazer alterações no código, gerando a versão 2.0. Essas alterações visaram inserir a documentação de atributos de dados na ontologia, o que é importante para gerar diagramas de classe mais completos.

É importante mencionar que as ontologias geradas neste trabalho também podem ser utilizadas para auxiliar a navegação e entendimento das interdependências entre os elementos do modelo de processos de negócio. Além disso, contribuem para organizar e complementar o conhecimento sobre o modelo. Por outro lado, são compreensíveis por diversos perfis de usuários, inclusive das equipes de desenvolvimento de software, que podem automatizar os processos de negócio da organização. As ontologias também são

legíveis por máquina e permitem a realização de consultas semânticas para extração de conhecimento contextualizado, bem como inferências de novos conhecimentos acerca do domínio (FIGUEIREDO, 2018). As consultas semânticas neste trabalho usam a linguagem SPARQL (*SPARQL Protocol and RDF Query Language*) e o *framework* Apache Jena.

Uma visão geral deste trabalho encontra-se na **Figura 1**, onde pode ser observado que o modelo de processos de negócio deve ser convertido para o formato XPDL v2.2, para então constituir a entrada do sistema PM2ONTO v2.0. A ontologia em OWL, por sua vez, constitui a entrada para o sistema OnToSRS, a fim de gerar o documento ERS.

Figura 1 - Visão geral do processo sistematizado proposto, incluindo o sistema OnToSRS.

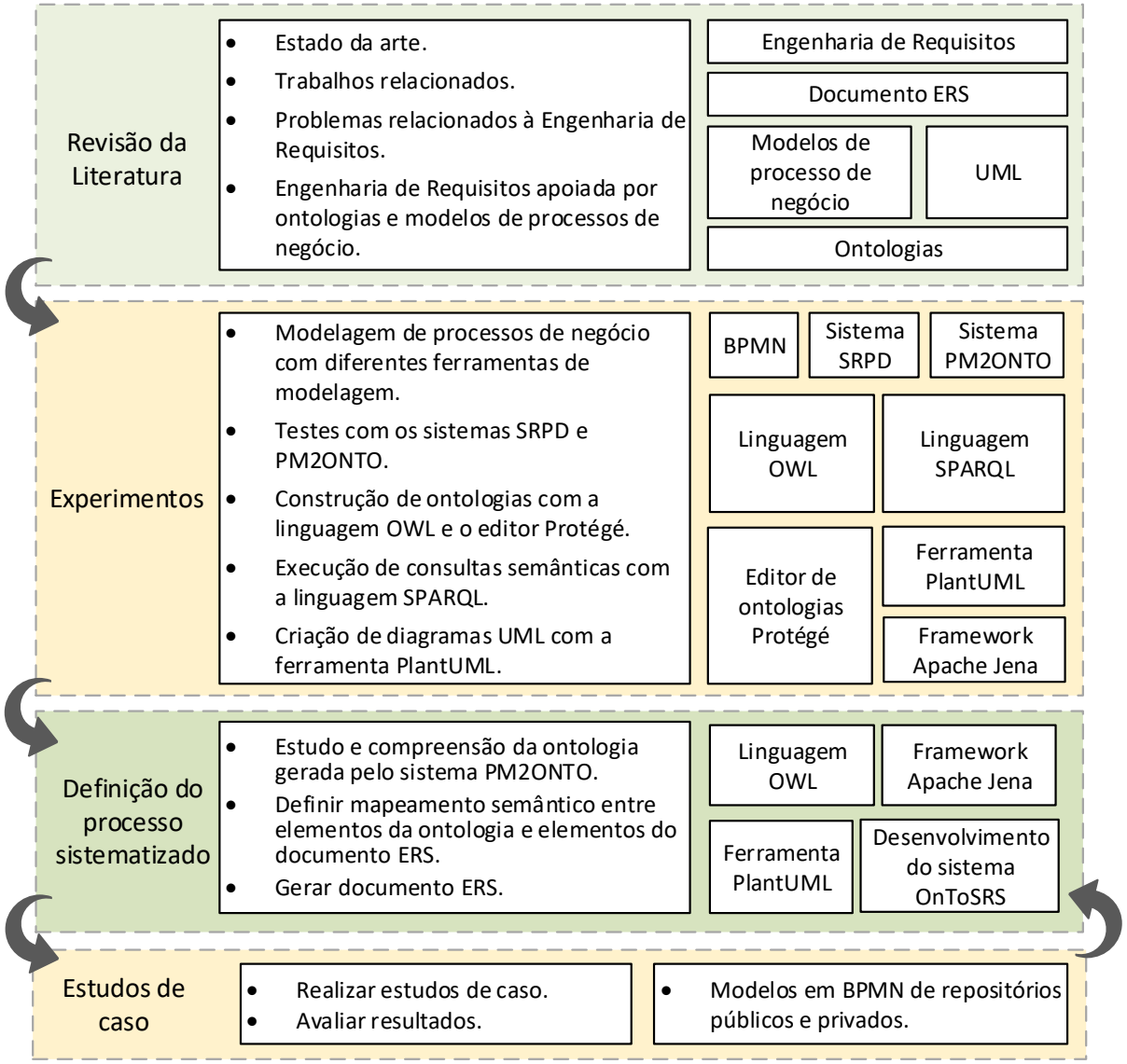


Fonte: elaborado pelo autor.

1.2 METODOLOGIA DE TRABALHO

Buscando atingir os objetivos apresentados na *Seção 1.1*, foram realizadas diversas atividades, como mostrado na **Figura 2**. Inicialmente, foram realizados estudos com o objetivo de compreender os principais problemas da Engenharia de Requisitos e como eles podem impactar na qualidade do software. Assim, foi identificado que grande parte desses problemas tem origem nas incertezas relacionadas ao conhecimento do domínio e, principalmente, à comunicação entre os diferentes perfis de profissionais envolvidos. Assim, os estudos foram direcionados a averiguar meios de aproximar a equipe de negócios e os profissionais de TIC no contexto de um projeto de software. Segundo a literatura, modelos de processos de negócio em BPMN podem ser utilizados com esse objetivo, mediante a transformação de modelos. Entretanto, processos de transformação entre diferentes modelos podem ocasionar perdas de informação, o que pode ser mitigado mediante boas práticas de modelagem e documentação textual.

Figura 2 - Visão geral das atividades desenvolvidas neste trabalho.



Fonte: elaborado pelo autor.

De modo geral, independentemente da notação utilizada, um modelo de processos de negócio possui incertezas nos aspectos sintáticos, semânticos e pragmáticos, que devem ser considerados durante o processo de modelagem. Neste contexto, foram realizados diversos estudos com o objetivo de compreender como esses aspectos impactam na qualidade dos modelos de processos de negócio e na conversão desses para outros modelos derivados. Também foram realizados experimentos com ferramentas de modelagem com o objetivo de compreender a especificação de BPMN v2.0 e, principalmente, aplicar os ajustes e boas práticas propostos por Figueiredo (2018).

Em relação às ontologias, foram realizados estudos para compreender a teoria relacionada e como podem ser úteis em ambientes corporativos. Nessa direção, foram encontradas propostas para representar o conhecimento implícito nos modelos de

processos de negócios por meio de ontologias. O trabalho de Figueiredo (2018) teve uma importante contribuição, ressaltando que foi possível interações com o próprio autor. Análise e testes com o sistema PM2ONTO foram relevantes, permitindo compreender os meios e técnicas para a construção de ontologias de processos a partir de modelos em BPMN v2.0, exportados para XPDL v2.2.

A partir dos estudos mencionados, foram investidos esforços na investigação de como extrair requisitos de software a partir de modelos em BPMN e de ontologias em OWL. Experimentos para recuperação de informações das ontologias possibilitaram identificar quais elementos ontológicos podem ser mapeados para elementos do documento ERS e para a criação de diagramas em UML.

1.3 ORGANIZAÇÃO DOS CAPÍTULOS

Considerando os objetivos deste trabalho, apresentados na *Seção 1.1*, e a metodologia apresentada na *Seção 1.2*, este trabalho está organizado em mais seis capítulos.

No *Capítulo 2*, são apresentados conceitos e aspectos relevantes para este trabalho sobre os modelos de processos de negócio em BPMN, bem como questões relacionadas a boas práticas de modelagem. De modo complementar, o APÊNDICE A apresenta uma visão geral dos elementos gráficos do vocabulário de BPMN v2.0.

No *Capítulo 3*, é apresentada uma visão geral sobre ontologias e suas aplicações no contexto de modelos de processos de negócio e da Engenharia de Requisitos. Esse capítulo também aborda a linguagem OWL, para representação de ontologias, e a linguagem SPARQL, para consultas em ontologias.

No *Capítulo 4*, processos, conceitos e problemas da Engenharia de Requisitos são abordados, bem como maneiras dos modelos de processos de negócios em BPMN e ontologias em OWL apoiarem a especificação de requisitos de software.

No *Capítulo 5*, o processo sistematizado para a extração de requisitos de software a partir de ontologias de processos é apresentado. Esse capítulo também inclui as ferramentas utilizadas no desenvolvimento do sistema OnToSRS (*Ontology to Software Requirements Specification*), que implementou o referido processo, definido neste trabalho.

No *Capítulo 6,* é apresentada uma forma de avaliação do processo sistematizado, incluindo quatro estudos de casos, os quais mostram a eficácia do processo apresentado no *Capítulo 5*.

Por fim, no *Capítulo 7* são apresentadas as considerações finais e os trabalhos futuros que podem dar continuação ao aprimoramento e extensão deste trabalho.

2 MODELAGEM DE PROCESSOS DE NEGÓCIO COM BMPN

Seguindo as recomendações da abordagem de gerenciamento de processos de negócios, conhecida como BPM (*Business Process Management*), os processos de negócio devem ser definidos sob uma ampla perspectiva, integrando estratégias, objetivos e necessidades de clientes em processos ponta a ponta, objetivando melhoria contínua por meio de constantes análises, monitoramento e transformação de processos (ABPMP BRAZIL, 2013). Para Van der Aalst (2013), a disciplina BPM consiste em combinar os conhecimentos da Tecnologia da Informação (TI) e de Gestão, aplicando-os nos processos operacionais de um negócio, com o objetivo de ir além da visão de processos tradicionais e isolados, incluindo desde a automação e integração até a análise, gerenciamento e monitoramento de processos.

A Associação Internacional de Profissionais de Gerenciamento de Processos de Negócios no Brasil, identificada como ABPMP Brazil (*Association Of Business Process Management Professionals International, Brasil*), define processo de negócio como “uma agregação de atividades e comportamentos executados por humanos ou máquinas para alcançar um ou mais resultados” (ABPMP BRAZIL, 2013). Segundo Baldam, Valle e Rozenfeld (2014), “o propósito de qualquer processo de negócio é transformar uma entrada qualquer (energia, informação, materiais ou clientes) em uma ou mais saídas, com maior valor econômico ou social”. A modelagem de processos de negócio, por sua vez, consiste em abstrair a estrutura dinâmica da organização em um ou mais modelos que descrevam suas atividades (ex.: modelo *AS IS* e *TO BE*) sob os mais diversos aspectos (BALDAM; VALLE; ROZENFELD, 2014).

Frente ao exposto, a *Seção 2.1* apresenta uma visão geral da notação BPMN, que é usada na modelagem dos processos de negócio considerados neste trabalho. Na *Seção 2.2*, são apresentados aspectos relacionados à qualidade dos modelos de processos de negócio. Na *Seção 2.3*, são apresentados trabalhos relevantes no contexto das boas

práticas de modelagem em BPMN. Por fim, na *Seção 2.4*, são apresentadas as considerações finais deste capítulo.

2.1 NOTAÇÃO BPMN

A notação BPMN tem sido muito utilizada atualmente para a modelagem de processos de negócio, pois é especificada pelo consórcio OMG, de ampla repercussão internacional. Segundo a associação ABPMP Brazil (2013), essa notação foi inicialmente definida pela grupo BPMI (*Business Process Management Initiative*), que foi incorporado pelo consórcio OMG. A versão atual da especificação BPMN é a 2.0.2, publicada em dezembro de 2013. Entretanto, pelo que se pode avaliar neste trabalho, grande parte dos sistemas modeladores de processos de negócio, atualmente, consideram a versão 2.0, que foi publicada em 2011.

Segundo Ditoro (2017), antes da notação BPMN, muitos analistas de negócio usavam simples e intuitivos diagramas de fluxos de dados ou fluxogramas para modelar processos. Os modelos de processos de negócio, apesar de eficientes em muitos casos, eram extremamente limitados, visto que não conseguiam expressar aspectos essenciais de um processo de negócio. Além do mais, os diagramas de fluxos de dados e fluxogramas não possuem padronização de documentação, o que dificulta a exportação para outros modelos. Outra desvantagem é o número de símbolos gráficos que é muito pequeno se comparado a notação BPMN. Ademais, após a compreensão do processo de conversão e mapeamento para outras notações, tais como a UML, o modelo era totalmente descartado. Sob o ponto de vista de Engenharia de Requisitos, isso impedia a rastreabilidade dos requisitos funcionais, além de impedir que o modelo se tornasse um ativo da organização.

A primeira versão da notação BPMN (v1.0) trouxe um conjunto de símbolos padronizados voltados exclusivamente a processos, o que fez com que analistas de negócio pudessem modelar processos com diferentes níveis de detalhamento. Dentre as inovações introduzidas pela notação BPMN, destacam-se: (a) os meios para capturar os fluxos de exceção, que costumavam ser ignorados pelos analistas de negócio; (b) os *gateways* (portas), que controlam o sequenciamento das atividades de um processo; (c) as piscinas e raias (*pools* e *lanes*), que deixam claras as interações entre os diferentes membros envolvidos em um processo (DITORO, 2017).

De modo geral, a notação BPMN é uma linguagem visual semanticamente rica se comparada a outras linguagens ou diagramas usados para modelar processos. Enquanto a

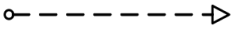
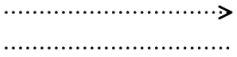
elaboração de um diagrama de atividades em UML conta com aproximadamente 20 símbolos de modelagem, a elaboração de um modelo de processos de negócio em BPMN pode utilizar mais de 100 símbolos. A especificação de BPMN inclui diferentes tipos de eventos, dados, fluxos, mensagens, entre outros elementos, o que possibilita modelos mais expressivos e robustos (CORREIA; ABREU, 2012). Entretanto, essa grande quantidade de símbolos pode se tornar uma grande desvantagem, dependendo da complexidade do processo, da maneira como o processo foi modelado e, principalmente, do nível de conhecimento do profissional modelador, em termos de domínio e de técnicas de modelagem (ABPMP BRAZIL, 2013; KROGSTIE, 2016).

A notação BPMN, sob o ponto de vista do padrão MDA, é um importante recurso para a consolidação da construção de softwares usando modelos focados em processos de negócios, independentemente de tecnologia e de implementação.

De acordo com a especificação BPMN, há cinco categorias de elementos de modelagem (**Tabela 1**): objetos de fluxo (*Flow Objects*), dados (*Data Objects*), objetos de conexão (*Connecting Objects*), vias (*Swimlanes*) e artefatos (*Artifacts*). Os objetos de fluxo são os principais elementos que definem o comportamento de um processo e são eles: eventos, atividades e *gateways*. Os dados são representados por quatro elementos: objetos de dados, entrada, saída e armazenamento. Os objetos de conexão podem ser: fluxos de sequência, fluxos de mensagem, associações e associações de dados. A vias são usadas para agrupar os outros elementos de modelagem e são divididas em: raias (*lanes*) e piscinas (*pools*), as quais agrupam duas ou mais raias (OMG, 2013). Entretanto, para simplificar o entendimento dos elementos de modelagem, podem ser consideradas, inicialmente, apenas os elementos básicos de modelagem mostrados na **Tabela 1**.

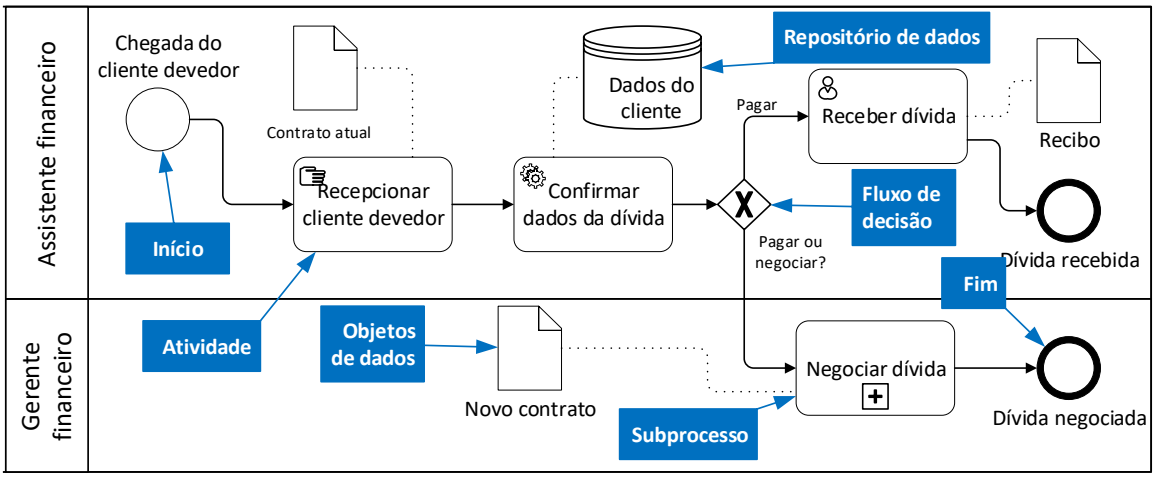
Geralmente, a maioria dos elementos gráficos usados em um modelo em BPMN fazem parte do conjunto de elementos básicos mostrados na **Tabela 1,** embora possa ter outros elementos mostrados no APÊNDICE A. Os elementos gráficos BPMN possuem especializações, de modo que há diferentes tipos de eventos, atividades, *gateways* e fluxos. O vasto conjunto de elementos gráficos contribui para a expressividade do modelo, porém aumenta consideravelmente sua complexidade, seja na criação ou na leitura. Assim, é necessário que os profissionais envolvidos com a modelagem usem um guia ou manual de referência e conheçam boas práticas de modelagem. Também é necessário investir em capacitação, visando construir modelos válidos sintática e semanticamente (HAISJACKL *et al*., 2018; KROGSTIE, 2016). Na **Figura 3**, é mostrado um exemplo simples de um diagrama de processos BPMN, criado usando a ferramenta de modelagem Microsoft Visio, para modelar o processo “Negociar dívida”. Os retângulos azuis não fazem parte do modelo e estão presentes apenas para fins explicativos.

Tabela 1 - Elementos básicos para modelagem em BPMN.

Elemento	Descrição	Notação
Evento	Algo que acontece durante o curso do processo.	
Atividade	Termo genérico usado para uma tarefa que a organização executa em um processo.	
Gateway	Usado para controlar a divergência e a convergência no fluxo de execução de um processo.	
Fluxo de Sequência	Usado para mostrar a ordem em que as atividades são executadas em um processo.	
Fluxo de Mensagem	Usado para mostrar o fluxo das mensagens entre os participantes do processo.	
Associação	Usado para vincular informações e artefatos com elementos de modelagem. Anotações e outros artefatos podem ser associados com elementos gráficos. A linha com a seta indica a direção do fluxo, quando necessário.	
Pool (piscina)	Representa um participante em processos colaborativos. Uma piscina pode conter detalhes do processo ou ser uma caixa-preta.	
Lane (raia)	Representa uma subpartição em um processo, às vezes dentro de uma piscina. Usado para organizar e categorizar as atividades.	
Data Object (objeto de dados)	Fornecem informações sobre o que as atividades precisam para serem executadas e/ou o que elas irão produzir. Pode ser um objeto singular ou uma coleção.	
Mensagem	Descreve o conteúdo de uma comunicação entre dois participantes.	
Grupo	Agrupa elementos que estão na mesma categoria. Uma categoria é um rótulo e podem ser usadas para documentação e análise. Esse elemento não altera o fluxo do processo.	
Anotação (vinculada a uma associação)	Usada para prover informações textuais adicionais para quem vai ler o modelo BPMN.	

Fonte: OMG (2013).

Figura 3 – Exemplo de um modelo de processos de negócio simples: “Negociar dívida”.



Fonte: Elaborado pelo autor.

Como pode ser observado na **Figura 3**, a piscina (*pool*) representa o processo “Negociar dívida” e é composta por duas raias (*lanes*) identificadas como “Assistente financeiro” e “Gerente financeiro”. Cada uma das raias representa um ator ou um departamento da organização. As atividades são unidades de trabalho, executadas sequencialmente, de acordo com o fluxo de sequência. É possível notar que cada uma das atividades está marcada com ícones que indicam como ela será executada. Por exemplo, a atividade “Recepcionar cliente devedor” é do tipo manual, ou seja, executada manualmente, sem uso de um sistema. A atividade identificada como “Receber dívida” é do tipo usuário, ou seja, executada pelo usuário com o auxílio de um sistema. A atividade nomeada “Confirmar dados da dívida” é do tipo serviço, requerendo o uso de um sistema. A saída dessa atividade define qual dos dois caminhos deve ser seguido: execução da atividade “Receber dívida” ou do subprocesso “Negociar dívida”. Essa decisão é representada pelo *gateway* exclusivo (representado com um “X” ao centro), nomeado “Pagar ou negociar”. Um subprocesso possui seu próprio fluxo de execução, agrupando atividades, *gateways* e outros elementos.

Além de elementos gráficos para modelagem de processos, a especificação de BPMN define elementos para diagramas de colaboração, conversação e coreografia, mas que não fazem parte do escopo desse trabalho (KLUZA *et al.*, 2017; OMG, 2013). Um diagrama de “colaboração” representa um processo colaborativo, com foco na troca de mensagens entre os participantes. Um diagrama de “conversação” pode ser considerado uma versão simplificada de um diagrama de colaboração, com foco na lógica da troca de mensagens entre os participantes. Por fim, um diagrama de “coreografia” define o comportamento esperado entre dois ou mais participantes de um processo.

Geralmente, os sistemas modeladores permitem a exportação de modelos BPMN para o formato XPDL (*XML Process Definition Language*), que é um formato de documento XML (*eXtensible Markup Language*) definido pelo consórcio WfMC (*Workflow Management Coalition*), com o objetivo de possibilitar intercâmbio entre as diferentes ferramentas de modelagem. O formato XPDL é baseado no padrão XSD (XML *Schema Definition*), que formaliza as representações dos elementos de um digrama de processos em BPMN. Assim, é possível transformar um modelo de processos de negócio em BPMN para o formato XPDL sem que haja perda de informações (WFMC, 2012).

Vale ressaltar que, apesar de sua grande capacidade expressiva, a notação BPMN não fornece recursos para modelagem de decisão (regras de negócio), nem para a modelagem de casos (instância particular da execução de um processo dotada de aspectos que não foram previstos, sendo necessário o conhecimento humano para sua resolução). Sendo assim, as notações DMN (*Decision Model and Notation*) e CMMN (*Case Management Model and Notation*) são definidas para lidar respectivamente com regras de negócio e gerenciamento de casos. As notações DMN e CMMN não fazem parte deste trabalho, visto que o objetivo é extrair requisitos de ontologias criadas a partir de modelos de processos de negócio. Entretanto, essas duas notações poderão ser abordadas em trabalhos futuros.

2.2 FATORES QUALITATIVOS EM MODELOS DE PROCESSOS DE NEGÓCIOS

Um modelo, independentemente da sintaxe, possui três propósitos: (a) “representação” dos aspectos ou fenômenos do mundo real; (b) “simplificação” ou “redução” do que está sendo modelado a um conjunto de símbolos relacionados entre si; (c) “intencionalidade” específica para substituir um fenômeno por uma conceituação dentro de um contexto da modelagem (KROGSTIE, 2016).

O processo de modelagem usando BPMN envolve diversos profissionais com diferentes níveis de conhecimento sobre o domínio, que devem participar ativamente do processo. São modeladores de processos, analistas de sistemas, usuários, especialistas do domínio e demais *stakeholders.* Se esses profissionais não possuem suficientes conhecimentos de domínio e de modelagem em BPMN, o modelo pode não cumprir totalmente seus propósitos, causando problemas de comunicação.

Frente ao exposto, nesta seção são apresentados aspectos importantes na modelagem de processos de negócio em BPMN, levando em consideração os aspectos

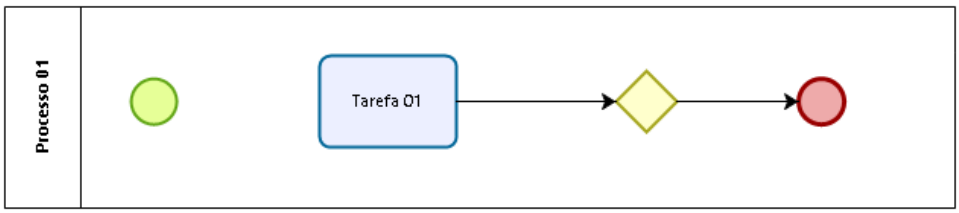
sintático, semântico e pragmático nas *Subseções 2.2.1* a *2.2.3*, respectivamente. Em cada subseção são exemplificadas boas práticas de modelagem, de moldo que os modelos que cumpram os propósitos definidos por Krogstie (2016).

2.2.1 QUALIDADE SINTÁTICA

A qualidade sintática de um modelo de processos de negócio relaciona o modelo e a linguagem usada na modelagem. Isso quer dizer que o modelo é avaliado por meio da verificação do uso correto dos elementos dessa linguagem. Geralmente, problemas de ordem sintática são evitados com o uso de ferramentas de modelagem que oferecem de recursos para validação sintática. Essa validação é possível graças ao metamodelo da especificação de BPMN, que impõe restrições às associações dos elementos de modelagem.

Na **Figura 4**, é ilustrado um exemplo de modelo com BPMN criado com a ferramenta *Bizagi Modeler*. O modelo possui dois erros sintáticos: (1) um evento de início que não possui fluxo de saída; (2) um *gateway* exclusivo com apenas um fluxo de saída. Ao ser validado, a ferramenta de modelagem *Bizagi Modeler* indica esses dois erros no modelo.

Figura 4 - Exemplo de modelo em BPMN com erros sintáticos.



Fonte: Krogstie (2016).

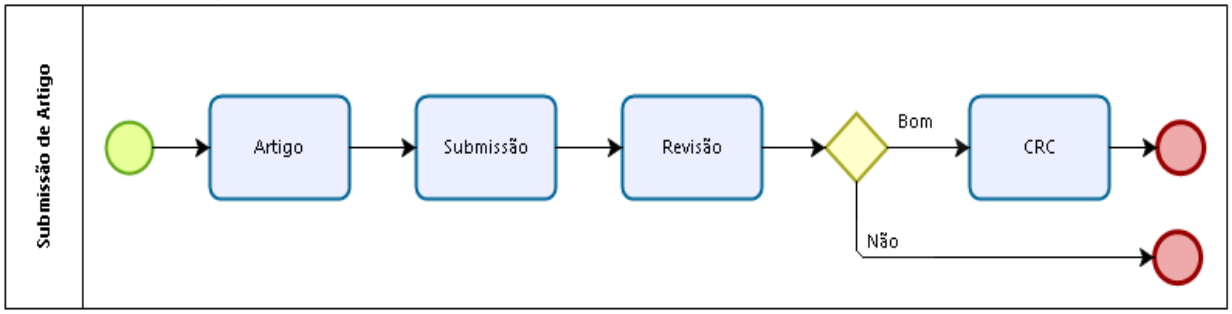
2.2.2 QUALIDADE SEMÂNTICA

A qualidade semântica diz respeito à correspondência entre o modelo e o domínio, o que expressa significado ao modelo. Essa qualidade é avaliada com critérios para validação e verificação da completude do modelo. Ressalta-se também que a conversão dos modelos em linguagem que pode ser compreendida por máquina é relevante para a qualidade semântica.

Considerando a qualidade semântica, os problemas podem estar relacionados à parte visual e/ou na parte das anotações textuais do modelo. Todavia, não é simples avaliar o grau de completude de um modelo nessas duas partes. Alguns trabalhos propõem mecanismos para se avaliar o grau de completude das anotações de um modelo, como Nogueira (2017), por exemplo, que faz uso de heurísticas denominadas “heurísticas de negócio” (HN) e critérios que possibilitam esse tipo de avaliação, apresentado na *Seção 4.3*.

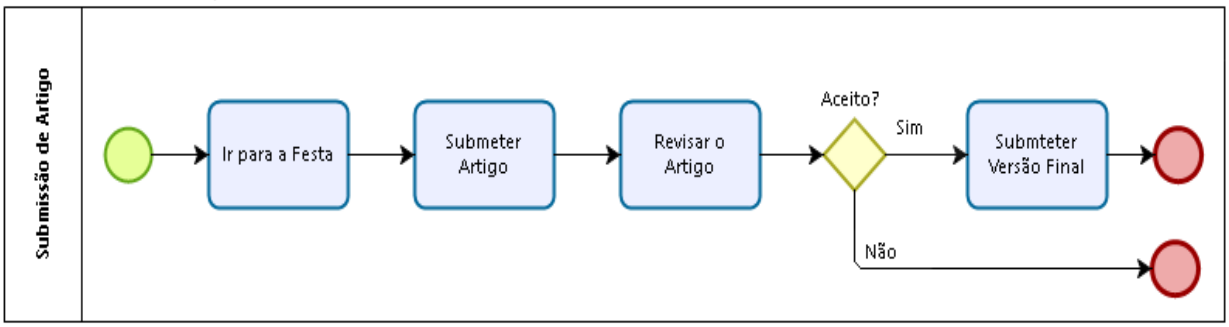
Convém observar que, se um modelo foi validado semanticamente, todos os elementos do modelo foram aplicados corretamente. Na **Figura 5**, há um exemplo de modelo inválido semanticamente, visto que as tarefas estão rotuladas de maneira superficial, dando margem a diversas interpretações. Por outro lado, um modelo tem alto grau de completude semântica quando suas declarações (parte textual) são consideradas corretas e relevantes ao domínio. Na **Figura 6**, é mostrado um exemplo de um modelo incompleto semanticamente, pois a primeira tarefa, “Ir para a Festa”, claramente não diz respeito ao domínio.

Figura 5 - Exemplo de um modelo em BPMN inválido semanticamente.



Fonte: Krogstie (2016).

Figura 6 - Exemplo de modelo em BPMN incompleto semanticamente.



Fonte: Krogstie (2016).

Considerando a documentação textual, os problemas de qualidade semântica dizem respeito à ausência parcial ou total de anotações semânticas. Ressalta-se que a especificação de BPMN não padroniza as formas de anotação, visto que a linguagem foi

criada com a capacidade de modelar processos independentemente do domínio. Isso significa que a notação BPMN é fortemente direcionada aos aspectos sintáticos, sustentada por um metamodelo que possui pacotes, metaclasses e relacionamentos necessários ao processo de modelagem. A falta de informação semântica padronizada em um modelo dificulta seu entendimento, comparação e reutilização. Portanto, é importante pensar em uma abordagem para anotações semânticas capaz de dar significados claros sobre os conceitos do domínio em questão (DI MARTINO *et al.*, 2015).

Assim, as anotações de ordem semântica nos modelos em BPMN são definidas informalmente. Desse modo, as anotações podem ser associadas a cada um dos elementos gráficos do modelo, usando atributos próprios (predefinidos) do sistema modelador. Alguns sistemas fazem uso de atributos estendidos, permitindo que os profissionais definam mais livremente suas anotações. De modo geral, os atributos objetivam adicionar mais expressividade aos modelos, documentando tarefas, papéis, regras e outros aspectos que o modelador julgar importante.

2.2.3 QUALIDADE PRAGMÁTICA

A qualidade pragmática diz respeito à relação entre o modo como o modelo está expresso tecnicamente e o que os interessados interpretam. Essa qualidade pode ser avaliada sob dois pontos de vista: (a) se a interpretação do modelo pelos participantes está correta em relação ao que se pretendia expressar; (b) se a interpretação do modelo por uma ferramenta de modelagem é correta em relação ao que o modelo desejava expressar, sendo possível a importação, exportação e execução. Enquanto a avaliação da qualidade pragmática por uma ferramenta de software se torna possível pelo suporte das especificações BPMN e XPDL, a avaliação pela compreensão humana depende muito da maneira como o processo de modelagem foi conduzido e de como os participantes se comunicam (FIGL, 2017; KROGSTIE, 2016).

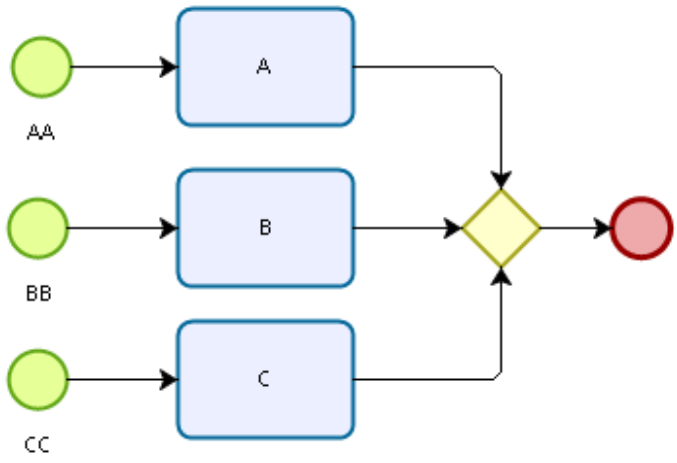
De modo geral, a compreensão de um modelo de processos de negócio pode ser difícil devido a diversos fatores, tais como: falta de conhecimento em relação à linguagem utilizada, complexidade, tamanho do modelo e/ou esforço necessário para deduzir propriedades importantes. Tsagkani e Tsalgatidou (2015) propuseram usar a abstração por meio de um conjunto de regras de exclusão e agregação de elementos, visando a concepção de diagramas mais simples, expressivos e inteligíveis, de modo a melhorar a qualidade pragmática.

Por outro lado, diferentes pessoas interessadas em um modelo de processos de negócio usam diferentes processos cognitivos e estratégias para interpretá-los, de acordo com o nível de compreensão sobre o domínio e o nível de conhecimento em BPMN. No entanto, a maioria das pessoas, inicialmente, fazem uma verificação geral do modelo, depois analisam cada um dos seus componentes. As pessoas com maior conhecimento em BPMN partem inicialmente para uma navegação sistemática. Além disso, todas as pessoas buscam analisar os modelos segundo experiências anteriores, gerando problemas de interpretação (HAISJACKL *et al.*, 2018).

Além de Tsagkani e Tsalgatidou (2015), foram encontrados na literatura outros trabalhos voltados a melhorar as práticas de modelagem, usando regras complementares com o objetivo de construir modelos mais compreensíveis. Esses trabalhos serão apresentados na *Seção 2.3*.

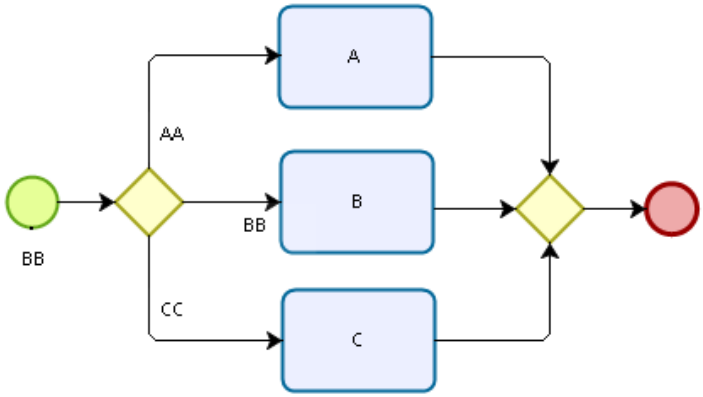
Na **Figura 7**, há um exemplo de como a qualidade pragmática pode ser comprometida por práticas inadequadas de modelagem. O referido modelo possui três eventos de início, porém, pragmaticamente, deveria ser apenas um evento com certas condições. Após um ajuste, mostrado na **Figura 8**, os três eventos de início foram substituídos por apenas um, seguido por um *gateway* exclusivo, deixando o modelo mais simples e expressivo.

Figura 7 - Exemplo de modelo com mais de um evento de início.



Fonte: Oca e Snoeck (2014).

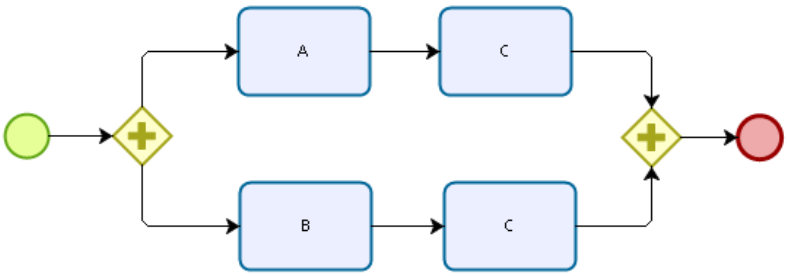
Figura 8 - Substituição dos eventos de início mostrados na Figura 7 por apenas um evento.



Fonte: Oca e Snoeck (2014)

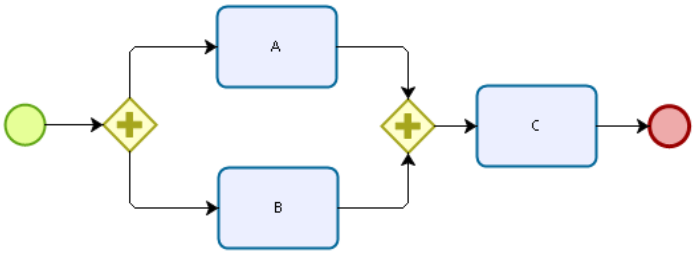
Um outro exemplo pode ser observado na **Figura 9**, com um modelo que possui duas atividades em duplicidade (“C”). Isso pode dar margem a diferentes interpretações, como, por exemplo: um dos interessados poderia imaginar que são duas atividades distintas, quando na verdade não são. Após a correção, mostrada na **Figura 10**, o modelo ficou mais simples e pragmático. Outras regras de junção, exclusão e substituição podem ser encontradas no trabalho de Tsakani e Tsalgatidou (2015), bem como de Oca e Snoeck (2014).

Figura 9 - Exemplo de modelo de processo com atividade duplicada (atividade “C”).



Fonte: Oca e Snoeck (2014).

Figura 10 - Unificação da atividade duplicada do exemplo mostrado na Figura 9.



Fonte: Oca e Snoeck (2014).

2.3 BOAS PRÁTICAS DE MODELAGEM DE PROCESSOS DE NEGÓCIO

Modelos de processos de negócios em BPMN podem se tornar extremamente complexos e ininteligíveis, devido à quantidade de símbolos e à sua capacidade expressiva. Portanto, é necessário treinamento e a utilização de um guia de referência, tanto na criação quanto na interpretação desses modelos. A comunicação padronizada de processos é um dos pilares da abordagem BPM e um dos benefícios da notação BPMN. Apesar disso, os profissionais envolvidos na modelagem podem criar diagramas demasiadamente complexos, o que pode dificultar a comunicação (ABPMP BRAZIL, 2013).

Diante do exposto, foram identificados diversos trabalhos voltados a boas práticas de modelagem, cujo objetivo é melhorar a qualidade dos modelos de processos de negócio em BPMN, objetivando melhor comunicação e compartilhamento dos modelos. Na *Seção 2.2* foram apresentadas algumas propostas para melhorar os aspectos semântico e pragmático dos modelos em BPMN. Nesta seção, por sua vez, serão apresentadas outras propostas relevantes para este trabalho voltadas a boas práticas de modelagem.

Assim, a *Subseção 2.3.1* apresenta alguns trabalhos da literatura relevantes para este trabalho no contexto da modelagem de processos de negócio. A *Subseção 2.3.2* apresenta critérios de modelagem para a geração de ontologias, definidas por Figueiredo (2018), necessários para a geração das ontologias consideradas neste trabalho.

2.3.1 TRABALHOS SOBRE BOAS PRÁTICAS DE MODELAGEM DE PROCESSOS DE NEGÓCIO

Um dos trabalhos relacionados a boas práticas de modelagem de processos de negócio que merece destaque é o estudo conduzido por Mendling, Reijers e Van Der Aalst (2010). Os referidos autores propuseram sete diretrizes, independentemente da notação, que ajuda a melhorar a qualidade dos modelos. As diretrizes são: (1) usar o menor número de elementos possível, objetivando a simplicidade e melhor expressividade do modelo; (2) minimizar o número de *gateways*, para evitar a criação de modelos demasiadamente complexos; (3) usar um evento de início e um evento de fim; (4) cada conector de divisão (*split*) deve ser unido a um conector de junção (*join*) do mesmo tipo; (5) evitar a utilização de elementos OR de roteamento, ou seja, se deve utilizar apenas os conectores XOR (*exclusive* OR) e AND, visto que são menos propensos a erros; (6) usar o sistema verbo-objeto para rotular as atividades; (7) decompor o modelo se ele tiver mais de 50 elementos

de modelagem. O trabalho de Mendling, Reijers e Van Der Aalst (2010) inspirou diversos outros trabalhos posteriormente.

O trabalho de Correia e Abreu (2012) propõe regras descritas na linguagem OCL (*Object Constraint Language*), normalmente utilizada para melhorar a semântica de diagramas UML. Os autores aplicaram 755 expressões em OCL como complemento no metamodelo da especificação BPMN. No trabalho de Figueiredo (2018), importante para este trabalho, foram consideradas quatro das regras destacadas por Correia e Abreu (2012):

1. O evento de início de um processo não pode receber nenhum fluxo de sequência, visto que esse evento define o início de um processo;
2. Fluxos paralelos de sequência que foram iniciados por um *gateway* paralelo devem ser unidos também por um gateway paralelo;
3. Eventos de início implícitos em um processo necessitam de eventos de fim implícitos;
4. Eventos de início do tipo não interrompíveis são apenas permitidos em subprocessos do tipo evento.

Correia e Abreu (2012) também classificam as regras em três tipos: (1) regras de controle de fluxo, que são referentes à interação e vinculação entre os elementos de modelagem; (2) regras de fluxo de dados, que se referem ao compartilhamento de objetos e depósito de dados pelas atividades; (3) recomendações e boas práticas, que se referem a regras gerais e opcionais que recomendam certos elementos de modelagem. Com o objetivo de validar as regras propostas, em conformidade com o metamodelo da especificação BPMN v2.0, os autores conduziram um estudo usando cerca de 50 modelos disponíveis em repositórios públicos. Dentre os modelos avaliados, apenas 53,6% estavam em conformidade com o metamodelo da especificação BPMN v2.0. Ademais, se as boas práticas definidas pelos autores fossem levadas em consideração, apenas 3,6% dos modelos seriam válidos.

De maneira similar, no trabalho de Leopold, Mendling e Gunther (2016) foram analisados 585 modelos de processos de negócio em BPMN (versão 2.0). Os autores encontram diversos problemas de qualidade, muitos deles já apontados por Oca e Snoeck (2014), Correia e Abreu (2012) e Krogstie (2016). Dentre os problemas, se destacam: (1) problemas de divisão (*split*) e união (*join*); (2) fluxos de mensagens; (3) decomposição e tamanho do modelo; (4) rotulagem de elementos. Diante dos problemas encontrados, os autores apresentaram cinco recomendações:

1. **Evitar junções e separações implícitas** - o uso inadequado de junções e separações é a principal causa dos *deadlocks* (combinação errada de *gateways* que impedem a continuidade da execução) e múltiplas junções. Nesse caso, recomenda-se a diretriz número quatro de Mendling, Reijers e Van Der Aalst (2010);
2. **Usar ferramentas de suporte para a decomposição de modelos** - à medida que os modelos crescem, eles podem apresentar inconsistências e não serem totalmente compreensíveis. As ferramentas de decomposição podem notificar os modeladores quando os modelos se tornem extensos ou até mesmo sugerir a decomposição desses modelos;
3. **Omitir eventos do tipo lançamento de mensagens** - segundo os estudos de Leopold, Mendling e Gunther (2016), muitos modeladores ligam incorretamente esse tipo de evento a outros elementos, causando confusão entre os modeladores. Portanto, os autores sugerem a utilização de atividades, no lugar desse tipo de evento;
4. **Criar um glossário compartilhado para manutenção de conceitos** - a reutilização de conceitos, como papéis de trabalho e objetos de dados, torna-se conveniente e necessária para a arquitetura de um processo;
5. **Usar ferramentas para verificações linguísticas** - o uso consistente de linguagem natural é algo difícil em ambientes de modelagem. Assim, os autores recomendam que a modelagem poderia usar técnicas de refatoração para rotulagens incompatíveis dos elementos de um modelo. Dessa forma, problemas de rotulagem seriam reduzidos, facilitando a comunicação entre os envolvidos.

Por fim, Oca e Snoeck (2014) catalogaram 27 problemas conhecidos e recorrentes na modelagem de processos em BPMN a partir de uma revisão sistemática da literatura. A partir desses problemas, foram criadas diretrizes com o objetivo de melhorar a qualidade pragmática dos modelos. As diretrizes estão agrupadas em três categorias:

1) **Quantidade de elementos** - identifica problemas relacionados à quantidade dos elementos e não ao relacionamento entre eles. Por exemplo, a primeira diretriz sugere decompor modelos que possuem mais de 31 elementos. As demais diretrizes tratam de outros elementos, como, por exemplo: elementos duplicados e desnecessários, múltiplos eventos de início e fim, ausência de eventos de início e fim, quantidade de eventos intermediários, número de arcos, número de *gateways*, número de atividades etc.;

2) **Morfologia** - trata da aparência do modelo. Por exemplo, a primeira diretriz desse grupo sugere técnicas para reduzir a profundidade dos *gateways* aninhados. Outras diretrizes tratam de questões de ciclicidade, alto número de atividades paralelas, caminhos muito longos entre evento de início e fim do processo, diversidade de *gateways*, complexidade do modelo, modularidade, legibilidade, entre outros;

3) **Apresentação** - trata de práticas para melhorar *layout* do modelo. Essas práticas não interferem nas qualidades semântica pragmática, mas ajudam a criar modelos mais apresentáveis e legíveis. Ademais, sugerem recomendações para organizar os elementos de modelagem e criação de rótulos adequados nos elementos.

2.3.2 CRITÉRIOS DE AJUSTE PARA MODELOS DE PROCESSOS DE NEGÓCIO

Objetivando ajustar modelos de processos de negócio em BPMN para posteriormente transformá-los em ontologias, Figueiredo (2018) definiu doze critérios que os modelos de processos de negócios devem seguir, sendo eles:

C1: todo processo deve possuir um único evento de início e pelo menos um de fim, com nomes “Início” e “Fim”, respectivamente;

C2: um processo deve conter pelo menos um participante;

C3: a documentação textual de cada elemento de BPMN deve conter, no mínimo, uma de suas propriedades descritivas básicas: nome, descrição ou documentação;

C4: o nome de uma atividade deve utilizar um verbo no infinitivo;

C5: o tipo de uma atividade deve ser sempre definido;

C6: elementos do mesmo tipo não devem possuir o mesmo nome;

C7: uma atividade deve ser executada por pelo menos um ator.

C8: as ações das atividades não devem ser atribuídas como nomes dos *gateways* (*gateways* representam a lógica de direcionamento após a tomada de uma decisão);

C9: os fluxos de sequência de um *gateway* exclusivo devem ser nomeados, com exceção do caso em que só há dois fluxos representando “Sim” ou “Não”. Nesse caso, apenas um dos nomes pode ser explícito e o outro poderá ser deduzido;

C10: os fluxos de sequência de um *gateway* inclusivo devem ser nomeados;

C11: atividades que são precedidas por um *gateway* paralelo devem ser unidas no final por outro *gateway* do tipo paralelo;

C12: atividades precedidas por um *gateway* inclusivo devem ser unidas ao final por outro *gateway* do tipo inclusivo.

Além desses critérios, Figueiredo (2018) recomenda diretamente as heurísticas de negócio (HN) e heurísticas de requisitos (HR) definidas por Nogueira (2017) e apresentadas na *Seção 4.3*. As heurísticas de negócio promovem melhorias na parte visual sem alterar a documentação textual. Por sua vez, as heurísticas de requisitos (HR) permitem a identificação de requisitos de software. Os critérios definidos por Figueiredo (2018) têm importância significativa neste trabalho, visto que as ontologias usadas nos estudos de caso apresentados no *Capítulo 6* são geradas pelo sistema PM2ONTO v2.0, que requer que os modelos de origem estejam em conformidade com esses critérios.

2.4 CONSIDERAÇÕES FINAIS

A modelagem de processos de negócio em BPMN possui grande importância neste trabalho, visto que as ontologias representativas de modelos de processos de negócio devem ser construídas a partir de modelos em BPMN devidamente ajustados segundo boas práticas de modelagem e, principalmente, documentação textual.

A modelagem padrão, levando em conta apenas os aspectos sintáticos do modelo, não é suficiente para a construção de modelos expressivos, aderentes ao negócio, compreensíveis aos envolvidos e, principalmente, adequadas para a transformação de modelos. Portanto, os aspectos sintático, semântico e pragmático devem ser considerados, além da documentação textual, a fim de construir modelos que permitam o mapeamento PIM para PIM de maneira adequada. Assim, é possível aplicar processos sistematizados com o objetivo de extrair informações contextualizadas a partir dos modelos ajustados, como mostrado no *Capítulo 5* e no *Capítulo 6*.

Os critérios definidos por Figueiredo (2018), apresentados na *Subseção 2.3.2*, são importantes no sentido de preparar os modelos para a posterior geração de ontologias de processo. De fato, esses critérios são considerados um padrão dentro do contexto da geração da ontologia de processos definida pelo autor. Isso significa que o aspecto sintático de um modelo não é suficiente em cenários que exigem maior expressividade, principalmente considerando o padrão MDA e a transformação de modelos.

No próximo capítulo serão abordados aspectos sobre as ontologias, que se mostram não apenas como uma alternativa aos modelos em BPMN, mas, principalmente, como um complemento, capaz de estender as qualidades semântica e pragmática dos modelos de processos de negócio.

3 VISÃO GERAL DA ABORDAGEM ONTOLÓGICA UTILIZADA

No atual contexto dos sistemas de informação, diversas pesquisas vêm sendo feitas com o intuito de desenvolver tecnologias para indexar, organizar, recuperar e compartilhar informações e conhecimentos. Considerando as necessidades tecnológicas das organizações contemporâneas, um sistema de informação deve utilizar mecanismos de representação do conhecimento de um domínio sob o ponto de vista humano e computacional. Nesse contexto, uma aplicação semântica é definida como um software que utiliza explícita ou implicitamente a semântica de um domínio, com o objetivo de melhorar sua usabilidade e validade. As ferramentas de busca na WWW, como Google, Bing e Yahoo, são típicos exemplos de aplicações semânticas, pois usam sinônimos e termos relacionados para enriquecer os resultados de uma pesquisa baseada em texto (BARTUSSEK *et al*., 2018). Sendo assim, as ontologias se apresentam como o principal recurso usado para representar o conhecimento em aplicações semânticas.

Neste capítulo são apresentados os principais conceitos relacionados a ontologias relevantes no contexto deste trabalho. Na *Seção 3.1*, é apresentada uma visão geral sobre ontologias. Na *Seção 3.2*, são apresentados meios para a representação de ontologias, com destaque para a linguagem OWL. Na *Seção 3.3*, é mostrado como usar a linguagem SPARQL para a realização de consultas semânticas em uma base de conhecimento. Na *Seção 3.4*, são apresentados trabalhos relacionados à construção de ontologias, manual ou automaticamente, a partir de modelos de processos de negócio. Na *Seção 3.5*, é apresentada uma breve introdução à integração entre ontologias. Por fim, na *Seção 3.6*, são apresentadas as considerações do capítulo.

3.1 VISÃO GERAL SOBRE ONTOLOGIAS

No contexto da Ciência da Computação, uma ontologia é a representação de um corpo de conhecimento compartilhado por uma comunidade. Essa representação é um modelo das entidades que fazem parte de um domínio e suas respectivas relações. Esse modelo deve permitir que um grupo de sistemas e usuários se comuniquem no nível adequado à execução de um conjunto de tarefas. Uma ontologia também pode ser definida como uma teoria lógica, responsável pelo significado pretendido de um vocabulário formal e limitado, capaz de representar uma visão particular de um domínio (GRIMM *et al*., 2011; GUARINO, 1998).

Uma ontologia é constituída por um vocabulário específico de um domínio, capaz de descrever uma realidade, mais um conjunto de suposições explícitas. Esse conjunto de suposições pode ser representado pela teoria da lógica de primeira ordem, em que as palavras do vocabulário aparecem como nomes de predicados unários ou binários, chamados, respectivamente, de “conceitos” e “relações”. As ontologias mais simples descrevem uma taxonomia de conceitos relacionados por subsunção, lembrando a ideia de generalização e especialização. Já as ontologias mais complexas possuem também axiomas, isto é, proposições verdadeiras por definição que expressam relações entre conceitos e, principalmente, restringem a intepretação esperada, possibilitando inferir novos conhecimentos sobre o sistema de conceitos (BARTUSSEK *et al*., 2018; GUARINO, 1998).

De acordo com Bartussek *et al*. (2018), apesar de destacar a terminologia de um domínio, o foco principal de uma ontologia é a sua estrutura ontológica inerente. Essa estrutura contempla os objetos que existem no domínio (conceitos), como eles podem ser organizados em classes, e como essas classes estão definidas e relacionadas. Assim, a interpretação das ontologias não depende do usuário que está lendo, mas do conhecimento que é especificado explicitamente (EUZENAT; SHVAIKO, 2013).

Assim, as ontologias possuem cinco características:

- **Formalidade** - fornece uma semântica formal que garante que a especificação do conhecimento do domínio possa ser processada por máquina;
- **Explicitude** - define o conhecimento explicitamente para torná-lo acessível por máquinas. Noções que não são explicitamente incluídas na ontologia não podem ser interpretadas por máquina;
- **Compartilhamento** - reflete um acordo sobre uma conceituação de um domínio entre pessoas em uma comunidade. Portanto, a construção de uma ontologia

deve ser associada a um processo de obtenção de consenso entre os interessados;

- **Conceitualidade** - especifica o conhecimento de maneira conceitual em termos de símbolos que representam conceitos e suas relações. Tais conceitos e relações podem ser intuitivamente compreendidos por seres humanos, visto que correspondem a modelos mentais;
- **Específicas de domínio** - são limitadas ao conhecimento de um domínio específico. Quanto mais restrito for o escopo do domínio, mais o engenheiro ontológico poderá se concentrar em criar premissas com os detalhes desse domínio. Dessa forma, a especificação do conhecimento do domínio poderá ser modularizada e expressa por meio de ontologias diversas.

De acordo com Euzenat e Shvaiko (2013), os componentes típicos de uma ontologia são:

- **Classes** - estruturas que organizam os conceitos de um domínio em uma taxonomia, usando a mesma ideia de generalização e especialização, onde classes genéricas podem ter subclasses mais específicas;
- **Atributos** - propriedades relevantes do conceito;
- **Instâncias** - objetos específicos de um conceito;
- **Relações** - tipos de relações entre os conceitos de um domínio;
- **Axiomas** - proposições sempre verdadeiras, usadas para restringir a interpretação e inferir novos conhecimentos;
- **Funções** - eventos que podem ocorrer no contexto do domínio.
- **Tipos de dados** - partes específicas de um domínio que especificam valores (ex.: *String* e *Integer* são tipos de dados aplicados aos atributos);
- **Valores de dados** - são os valores presentes nos atributos.

Assim, uma ontologia pode ser formalizada como ***O = {C, R, I, A}***, onde “C” representa o conjunto de classes, “R” o conjunto de relações, “I” as instâncias e “A” os axiomas (ISOTANI; BITTENCOURT, 2015).

Segundo Mizoguchi (2004), as ontologias podem ser classificadas em cinco categorias:

- **Ontologias genéricas** - ontologias gerais, que descrevem termos mais amplos e comuns a todos (ou quase todos) os domínios. Esse tipo de ontologia descreve conceitos como, por exemplo, eventos, tempo, objetos físicos e crenças;
- **Ontologias de domínio** - descreve um domínio particular ou "micromundo", como, por exemplo, medicina, mecânica, matemática, física, entre outros. Esse é o tipo de ontologia mais comum, visto que pode ser construído para representar o domínio de uma organização, por exemplo.
- **Ontologias de tarefas** - descrevem tarefas genéricas que são executadas no mundo real, como, por exemplo, vendas, compras, agendamentos, etc.;
- **Ontologias de aplicação** - descrevem conceitos que dependem de um domínio e tarefas específicos, sendo especializações das ontologias de domínio e tarefas, respectivamente. Correspondem a regras aplicadas a entidades de um domínio enquanto executam uma tarefa específica;
- **Ontologias de representação** - explicam as conceituações que fundamentam os formalismos da representação do conhecimento, deixando claro os compromissos ontológicos embutidos nesses formalismos.

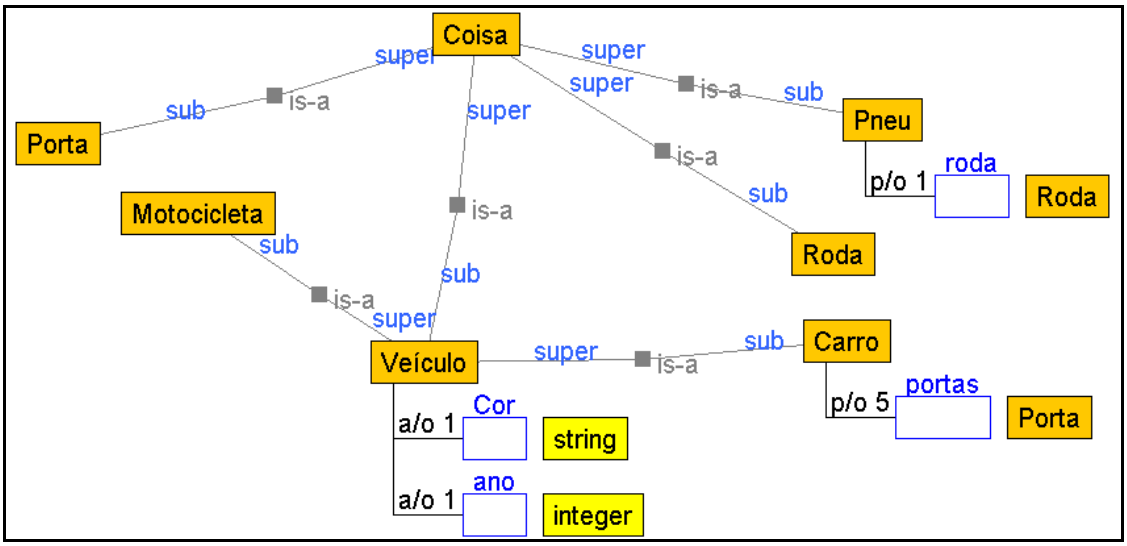
Diante do apresentado, entende-se que alguns dos principais papéis de uma ontologia dentro do contexto da Ciência da Computação são: (a) descrever um vocabulário comum aceito por diversas pessoas envolvidas em um contexto; (b) descrever uma estrutura de dados legível a humanos e computadores; (c) explicar o que é deixado implícito para facilitar o entendimento de um contexto; (d) sistematizar o conhecimento; (e) conectar sistemas, objetivando a navegação, armazenamento, pesquisa e troca de conhecimentos em uma comunidade (BARTUSSEK *et al*., 2018; GUARINO, 1998; MIZOGUCHI, 2004).

3.2 REPRESENTAÇÃO DE ONTOLOGIAS

As ontologias podem ser representadas de modo gráfico ou formal. A representação gráfica é usada para compreensão humana, enquanto a representação formal possibilita a compreensão da ontologia por máquina. Para que a representação formal seja possível, há diversas linguagens propostas, dentre as quais se destacam: RDF (*Resource Description Framework*), RDF-S (RDF Schema) e OWL (*Ontology Web Language*).

Considerando a representação gráfica, na **Figura 11** é mostrada uma ontologia criada com a ferramenta *Hozo – Ontology Editor* (*http://www.hozo.jp*). As relações "*is-a*" (é um) são as relações hierárquicas entre os conceitos (subclasses e superclasses). Por sua vez, as relações *"a/o"* (*attribute of*) são os atributos das classes, seus respectivos tipos e sua cardinalidade. Já as relações *"p/o"* (*part of*) denotam que um conceito faz parte de outro. No exemplo da **Figura 11**, um "Veículo" é uma "Coisa" que pode ser classificada como "Motocicleta" ou "Carro" (hierarquia) e possui os atributos "cor" e "ano". O "Carro" pode ter no máximo cinco portas.

Figura 11 - Representação gráfica de uma ontologia que representa um veículo.



Fonte: elaborado pelo autor.

A linguagem ontológica RDF, usada na representação formal de ontologias, consiste em um *framework* proposto pelo consórcio W3C (*World Wide Web Consortium*) para descrever metadados. Esse *framework* permite a criação de uma estrutura chamada "tripla", formada por um nó sujeito, uma relação chamada de predicado e um nó objeto (*sujeito → predicado → objeto*). Por meio dessa tripla, é possível, por exemplo, estabelecer a relação entre um autor e seus livros, de maneira que não apenas pessoas compreendam essa relação, mas, principalmente, computadores. Para que a semântica seja expressa por meio de triplas, conhecidas como "vocabulário RDF", é necessário a definição de *tags* usando a especificação RDF-S. A especificação RDF-S representa triplas por meio de classes, propriedades e seus relacionamentos, incluindo a definição de *tags* e sua estrutura hierárquica (taxonomia), com o objetivo de restringir os valores de uma "tripla" em RDF. Sendo assim, a especificação RDF-S fornece os elementos mínimos (um vocabulário para definição de classes, propriedades e hierarquias) para a representação de uma ontologia,

porém com diversas limitações, principalmente no que diz respeito a apoiar os mecanismos de inferência. Assim, foi necessária a criação de uma linguagem mais expressiva, conhecida como OWL (ISOTANI; BITTENCOURT, 2015).

A linguagem OWL também uma especificação do consórcio W3C, usada para representar o conhecimento sobre coisas, grupos de coisas e as suas relações. É definida como uma linguagem computacional que possibilita a representação do conhecimento e sua leitura e processamento por computadores, possibilitando a verificação da consistência e explicitação do conhecimento implícito. A linguagem também faz parte da pilha de tecnologias da Web Semântica do consórcio W3C, que também inclui os conceitos de RDF e SPARQL (W3C, 2012). Convém destacar que o propósito a linguagem OWL é satisfazer o formalismo exigido no contexto da Web Semântica, possibilitando que agentes (pessoas ou programas) possam compreender e responder a consultas por meio de descrições ontológicas (ISOTANI; BITTENCOURT, 2015).

A linguagem OWL é dividida em três sublinguagens, com o objetivo de satisfazer diferentes perfis de usuários e comunidades: OWL Lite, OWL DL e OWL Full. A linguagem OWL Lite é adequada a simples classificações hierárquicas e restrições. A linguagem OWL DL possibilita expressividade máxima, enquanto retém a completude computacional, isto é, garante que as conclusões são computáveis e serão realizadas em um tempo finito. A linguagem OWL Full possibilita expressividade máxima, mas sem garantia de completude computacional, ou seja, as restrições sintáticas da linguagem não são consideradas, fazendo com que uma ontologia aumente o vocabulário pré-definido. Observa-se que é pouco provável que um software seja capaz de implementar o raciocínio completo para todos os recursos da linguagem OWL Full (W3C, 2012).

A linguagem OWL possui uma estrutura sintática obrigatória bem definida em RDF/XML, como exemplificado em **Tabela 2**, **Tabela 3** e **Tabela 4**. Essa estrutura permite o reconhecimento por qualquer software com suporte a OWL. Há também outras sintaxes específicas para diferentes propósitos.

O propósito da linguagem OWL é representar conhecimento. Para tal, são considerados três aspectos (W3C, 2012; ISOTANI; BITTENCOURT, 2015):

1. **Entidades** - elementos usados para representar um objeto do mundo real;
2. **Expressões** - combinações de entidades com o objetivo de formar descrições mais complexas. Por exemplo, as entidades <Professor> e <Doutor> podem ser combinadas para formar a entidade <ProfessorDoutor>;
3. **Axiomas** - as declarações básicas que uma ontologia OWL pode expressar. Por meio dos axiomas, pode-se fazer inferências sobre as entidades definidas,

como, por exemplo: <Estudante><subClassOf><Pessoa>, que permite concluir que um Estudante é uma subclasse de Pessoa.

Tabela 2 - Exemplos de axiomas em OWL para classes e instâncias.

Sintaxe OWL (RDF/XML)	Descrição
`<Person rdf:about=”Mary Jane” />`	Define uma instância (Mary Jane) da classe “Person”.

`<owl:Class rdf:about="Woman">`
`  <rdfs:subClassOf rdf:resource="Person"/>`
`</owl:Class>`
	Define uma subclasse (“Woman”) de “Person”.

`<owl:Class rdf:about="Person">`
`  <owl:equivalentClass rdf:resource="Human"/>`
`</owl:Class>`
	Define equivalência semântica entre classes. “Person” equivale a “Human”.

`<owl:AllDisjointClasses>`
`  <owl:members rdf:parseType="Collection">`
`  <owl:Class rdf:about="Man"/>`
`  <owl:Class rdf:about="Woman"/>`
`  </owl:members>`
`</owl:AllDisjointClasses>`
	Define classes disjuntas, ou seja, se um indivíduo é uma instância de “Man”, logo ele não pode ser uma instância de “Woman”.

Fonte: W3C, 2012.

Tabela 3 - Exemplos de axiomas OWL para relacionamentos entre instâncias.

Sintaxe OWL (RDF/XML)	Descrição

`<rdf:Description rdf:about="John">`
`  <hasWife rdf:resource="Mary"/>`
`</rdf:Description>`
	Define um relacionamento (“*hasWife*”) entre as instâncias John e Mary. Neste caso, John tem uma esposa, que é Mary.

`<owl:NegativePropertyAssertion>`
`  <owl:sourceIndividual rdf:resource="Bill"/>`
`  <owl:assertionProperty rdf:resource="hasWife"/>`
`  <owl:targetIndividual rdf:resource="Mary"/>`
`</owl:NegativePropertyAssertion>`
	Define que duas instâncias não estão conectadas por uma propriedade. Neste caso, Mary não é esposa de Bill.

`<owl:ObjectProperty rdf:about="hasWife">`
`  <rdfs:subPropertyOf rdf:resource="hasSpouse"/>`
`</owl:ObjectProperty>`
	É possível definir sub-propriedades para os relacionamentos. Por exemplo, Se A é esposa de B, A também é cônjuge de B. Entretanto, B não é esposa de A.

Fonte: W3C, 2012.

Tabela 4 - Exemplos de axiomas OWL para restrições, diferenciação, igualização e tipos de dados.

Domínio e restrições de possíveis valores	

`<owl:ObjectProperty rdf:about="hasWife">`
`  <rdfs:domain rdf:resource="Man"/>`
`  <rdfs:range rdf:resource="Woman"/>`
`</owl:ObjectProperty>`
	Define que a propriedade "hasWife" está no domínio Man e os valores possíveis devem ser instâncias da classe Woman.

Diferenciação e igualização e de indivíduos	

`<rdf:Description rdf:about="John">`
`  <owl:differentFrom rdf:resource="Bill"/>`
`</rdf:Description>`
	Define explicitamente que dois indivíduos são diferentes entre si. Nesse caso, John e Bill não são a mesma pessoa.

`<rdf:Description rdf:about="James">`
`  <owl:sameAs rdf:resource="Jim"/>`
`</rdf:Description>`
	Define explicitamente que James e Jim são o mesmo indivíduo.

Tipos de dados	

`<Person rdf:about="John">`
`  <hasAge rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">51</hasAge>`
`</Person>`
	É possível definir valores para certos tipos de relacionamentos. Nesse caso, John possui a propriedade "hasAge" e o valor é do tipo inteiro, 51. Os tipos de dados são definidos pela especificação XML *Schema Datatypes*.

Fonte: W3C, 2012.

Os recursos representados em OWL também devem possuir um identificador IRI (*Internationalized Resource Identifier*), com o objetivo de referenciar o recurso na Internet. Outro elemento importante é o de anotação (*Annotation*), que permite adicionar informações sobre a ontologia, ou seja, definir metainformações, como: autor, versão, entre outras informações relevantes.

A especificação OWL contempla elementos básicos de modelagem: classes (*owl:Class*), propriedades (*owl:ObjectProperty* e *owl:DataProperty*), indivíduos e instâncias (owl:*Individual*). Também possui quantificadores universal e existencial (*allValuesFrom* e *someValuesFrom*), restrições de cardinalidade (*maxCardinality*, *minCardinality* e *exactCardinality*) e outros recursos avançados para a representação do conhecimento. A especificação completa pode ser encontrada em W3C (2012). Na **Figura 12,** está ilustrado um fragmento da ontologia mostrada na **Figura 11**, representada na linguagem OWL.

Figura 12 - Parte da ontologia mostrada na Figura 11 escrita em OWL (sintaxe RDF/XML).

```
<owl:Class rdf:ID="Motocicleta">
    <rdfs:label>Motocicleta</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Veículo" />
</owl:Class>
<owl:Class rdf:ID="Carro">
    <rdfs:label>Carro</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Veículo" />
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:cardinality rdf:datatype="http://www.w3.org/
                2001/XMLSchema#nonNegativeInteger">5</owl:cardinality>
            <owl:onProperty rdf:resource="#has_portas" />
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="#has_portas" />
            <owl:allValuesFrom rdf:resource="#Porta" />
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>
```

Fonte: elaborado pelo autor.

3.3 CONSULTAS EM OWL

Após a construção de uma ontologia, é possível extrair informações sobre o conhecimento representado. A linguagem SPARQL, acrônimo recursivo para *SPARQL Protocol and RDF Query Language*, pode ser utilizada para essa finalidade. Atualmente na versão 1.1, a linguagem SPARQL possui semântica e sintaxe baseada no *framework* RDF (sujeito → predicado → objeto), permitindo a realização de consultas em bases de dados armazenados ou visualizados em formato RDF (grafos) por meio de um *middleware* (W3C, 2013). Diferente da linguagem SQL, que realiza consultas em bancos de dados relacionais e bem definidos, a linguagem SPARQL realiza consultas em bases de dados variadas, bem definidas ou não (como é o caso de ontologias), permitindo a recuperação de dados relacionados em uma escala mais complexa que SQL.

A estrutura de uma consulta SPARQL possui cinco elementos (W3C, 2013):

1. Prefixos (opcional) - define uma abreviação para um vocabulário que possui uma IRI específica:

```
PREFIX foo: <...>
```

2. Os dados que serão retornados:

SELECT ... ?source ?data1 ?data2 ... ?dataN

3. O conjunto de dados que será consultado (opcional):

FROM <...>

4. O padrão ou restrição para consulta:

WHERE { ... }

5. Modificadores da consulta (opcional): *GROUP BY, HAVING, ORDER BY, LIMIT, OFFSET* e *VALUES*.

Para exemplificar o uso da linguagem SPARQL, foi realizada uma consulta no *site* DBPedia (*http://dbpedia.org*), um projeto que permite extrair informações estruturadas da Wikipedia (*http://wikipedia.org*) por meio de consultas sofisticadas usando SPARQL. A consulta foi definida por meio do editor *Virtuoso SPARQL Query Editor* (*http://dbpedia.org/sparql*). Nessa consulta, mostrada na **Figura 13**, foram encontradas todas as bandas de *rock* que iniciaram suas atividades nos anos 1980.

Figura 13 - Exemplo de consulta ontológica na linguagem SPARQL.

```
01  PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
02  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
03  PREFIX dbo: <http://dbpedia.org/>
04     SELECT distinct ?s ?name ?genre ?year
05              FROM <http://dbpedia.org/sparql>
06     WHERE {
07                ?s a dbo:Band .
08                ?s foaf:name ?name .
09                ?s dbo:genre ?genre .
10                ?s dbo:activeYearsStartYear ?year
11            FILTER (regex(?genre, "rock", "i"))
12    FILTER (?year >= "1980"^^xsd:date && ?year <= "1990"^^xsd:date)
13    }
14    ORDER BY ?year ?name
```

Fonte: elaborado pelo autor.

Como observado na **Figura 13**, nas linhas 1, 2 e 3 há prefixos e na linha 4 tem uma cláusula SELECT e os dados que serão retornados. A linha 5 evidencia a base de dados RDF e a linha 6 a cláusula WHERE, que determina o padrão ou as restrições da consulta. Vale ressaltar que, no contexto do editor *Virtuoso*, os prefixos e a cláusula FROM são

desnecessários, visto que o editor carrega os prefixos mais comuns e determina a DBPedia como a base de dados padrão. Na linha 7, há o recurso (dbo:Band – classe Band da ontologia DBPedia) que será recuperado, representado por "?s". O caractere ponto (".") é usado para formar a expressão de busca no grafo; assim, cada tripla na cláusula *WHERE* é concatenada às outras. As propriedades listadas nas linhas 7 a 10 fazem parte do vocabulário específico. Dentro da cláusula *WHERE*, também há as regras de filtragem, definidas na cláusula *FILTER*, em que é possível determinar condições para o retorno dos resultados. Na linha 11, é aplicado um filtro para trazer todas as bandas cujo gênero possui os caracteres "rock". Na linha 12, é aplicado um filtro para selecionar as bandas que iniciaram suas atividades entre 1980 e 1990. Na linha 14, o resultado é ordenado pelo ano e depois pelo nome da banda.

3.4 ENGENHARIA ONTOLÓGICA

A Engenharia Ontológica é uma disciplina que consiste na construção de ontologias envolvendo metodologias e melhores práticas, que variam de acordo com o domínio (BARTUSSEK *et al.*, 2018). Apesar de existir diversas metodologias com esse propósito, nesta seção são apresentadas as metodologias propostas por Hoppe e Tolksdorf (2018) e De Nicola e Missikoff (2016), visto que mais se aproximam da metodologia definida neste trabalho.

O trabalho de Hoppe e Tolksdorf (2018) apresenta uma guia para a construção de ontologias em cenários corporativos, considerando os fatores complexidade e heterogeneidade. A abordagem dos referidos autores inicia com a modelagem de um número limitado de conceitos importantes para o domínio, chamados de conceitos categóricos. Esses conceitos devem ser modelados hierarquicamente, fazendo com que o engenheiro tenha uma visão clara dos termos que estão sendo usados na sua construção. O número limitado de conceitos faz com que a ontologia evolua, aumentando o número de termos e seus respectivos relacionamentos. Essa característica incremental estende a ontologia em partes conceituais, em que cada parte consiste em um conjunto de termos ainda não modelados, identificados a partir de novos documentos do domínio. A principal característica dessa abordagem é o fato dela ser puramente orientada a dados e pelos termos necessários para modelar a realidade do domínio, evitando a modelagem de conceitos irrelevantes para o contexto.

Por sua vez, De Nicola e Missikoff (2016) propõem uma metodologia identificada como UPON Lite, cujo propósito, segundo os autores, é ser mais simples e direta que outra

metodologias que exigem critérios rigorosos em cada uma das etapas da criação. A metodologia UPON Lite é baseda na metodologia UPON e é definida em seis etapas, onde cada uma delas gera uma saída que é a entrada da etapa posterior:

1) Terminologia do domínio - criar uma lista com todos os termos relevantes ao domínio;
2) Glossário - criar uma definição textual para cada um dos termos do domínio, indicando possíveis sinônimos;
3) Taxonomia - organizar os termos em uma estrutura hierárquica de generalização/especialização;
4) Predicados – indicar os termos que representam propriedades do glossário e que são conectados às entidades que eles caracterizam;
5) “Parthood” - definir relações estruturais do tipo *partOf*, inverso de *hasPart*, visto que esse tipo de relação é importante em diversos cenários;
6) Ontologia – implementar a ontologia usando uma linguagem ontológica apropriada.

Neste trabalho as atividades da Engenharia Ontológica são executadas automaticamente, visto que as ontologias usadas no processo sistematizado de geração do documento ERS são geradas pelo sistema PM2ONTO v2.0, de autoria de Figueiredo (2018). Entretanto, é possível identificar semelhanças na construção das ontologias do PM2ONTO v2.0 com as metodologias apresentadas nesta seção. Na *Seção 5.2* são apresentados mais detalhes referentes à ontologia considerada neste trabalho.

3.5 ONTOLOGIAS REPRESENTATIVAS DE MODELOS DE PROCESSOS DE NEGÓCIO

Os trabalhos destacados nesta seção tratam de como construir ontologias a partir de modelos de processos de negócios. Segundo Haller *et al*. (2008), as principais vantagens de representar modelos de processos de negócios ontologicamente são:

- Representar explicitamente a semântica de um modelo de processos de negócio melhora a interoperabilidade entre interessados e sistemas;
- Representações ontológicas de modelos de processos de negócio provêm suporte à realização de consultas em um alto nível de abstração. Dessa forma,

torna-se mais simples o compartilhamento de conhecimento e integração entre diferentes modelos;

- A utilização de linguagens ontológicas para uso na Web permite a utilização de vocabulários e conhecimentos já existentes, reduzindo o esforço de modelagem e auxiliando na definição de consultas compatíveis com diversos modelos e bases de conhecimentos;
- No contexto de serviços para a Web (*Web Services*), representações ontológicas de modelos de processos de negócio proporcionam consistência entre as descrições externas (coreografia) e os modelos de processos de negócios internos, permitindo a geração de interfaces de coreografia por meio de modelos de processos.

Ternai, Török e Varga (2016) propuseram um processo para a criação automática de uma ontologia de processos usando um arquivo XML bem estruturado e padronizado como modelo intermediário. Baseados na plataforma de modelagem intitulada BOC ADONIS *Modeling Platform* (*https://uk.boc-group.com/adonis*), os autores investigaram uma forma de associar cada elemento do modelo de processos de negócio em BPMN com os elementos de um arquivo XML. Dentro do arquivo XML exportado, os elementos do modelo são associados a uma *tag* chamada de *Instance*, e os respectivos atributos a uma *tag* chamada *Attribute*. Essa associação se deu por meio de um metamodelo que evidenciou os relacionamentos entre os elementos de um modelo de processos de negócio.

De maneira semelhante, Figueiredo (2018) propõe um processo para o mapeamento dos elementos de BPMN para uma ontologia de processos, bem como da respectiva documentação textual associada a cada elemento em um modelo de processos de negócio. Para que o processo aconteça, a modelagem de processos deve satisfazer doze critérios baseados nas boas práticas de modelagem definidas pelo autor. Opcionalmente, durante o processo de modelagem, podem ser usadas as heurísticas de negócio definidas por Nogueira (2017), objetivando trazer melhorias na parte gráfica e textual dos modelos. Como o trabalho de Figueiredo (2018) é relevante para este trabalho, são apresentados mais detalhes na *Seção 5.2*, considerando a estrutura ontológica definida.

3.6 INTEGRAÇÃO ENTRE ONTOLOGIAS

Em um ambiente organizacional é muito comum a existência de diferentes usuários usando diferentes sistemas e lidando com diferentes níveis de abstração e/ou detalhamento da informação e do conhecimento. Assim, é possível existir várias ontologias que representam o mesmo domínio e usam a mesma linguagem, porém com significativas diferenças. No contexto deste trabalho de mestrado, a integração entre ontologias é importante, visto que, em um ambiente organizacional, é possível haver diferentes ontologias que podem servir de entrada para o sistema de extração de requisitos de software. Tais ontologias, mesmo que sejam compatíveis semanticamente, podem apresentar diferentes tipos de heterogeneidade, dificultando o processo de integração, como mencionam Euzenat e Shvaiko (2013):

- **Heterogeneidade sintática** - ocorre quando duas ontologias não são representadas na mesma linguagem. Nesse caso, mesmo que sejam semanticamente idênticas, foram modeladas com diferentes propósitos e diferentes linguagens. Em muitos casos é possível fazer uma tradução entre as linguagens mantendo o seu significado;
- **Heterogeneidade semântica** - também chamada de heterogeneidade conceitual ou incompatibilidade lógica, ocorre quando duas ontologias apresentam diferenças na modelagem do mesmo domínio de interesse. Nesse caso, é comum a utilização de diferentes axiomas para expressar o mesmo conceito. Também é comum duas ontologias apresentarem diferentes níveis de detalhamento do mesmo domínio ou o mesmo nível de detalhamento, mas a partir de diferentes perspectivas;
- **Heterogeneidade pragmática** - também chamada de heterogeneidade semiótica, diz respeito ao modo com que as entidades e relacionamentos explícitos nas ontologias são interpretados pelos usuários. A pragmática depende do contexto em que a ontologia está sendo usada e, por isso, é difícil ser identificada por máquina.

Os tipos de heterogeneidade podem ocorrer ao mesmo tempo, dificultando o processo de integrar o conhecimento de diferentes ontologias.

De acordo com Sowa (2008), a integração entre ontologias é definida como o processo de encontrar elementos comuns entre duas ontologias A e B, criando uma nova,

chamada C, capaz de facilitar a interoperabilidade entre sistemas que são baseados nas ontologias A e B. A ontologia C pode substituir as ontologias A e B, ou pode servir apenas como um modelo intermediário entre os sistemas baseados em A e B. Dependendo da quantidade de alterações necessárias para gerar a ontologia C, diferentes tipos de integração podem ser considerados: a) alinhamento; b) compatibilidade parcial; c) unificação.

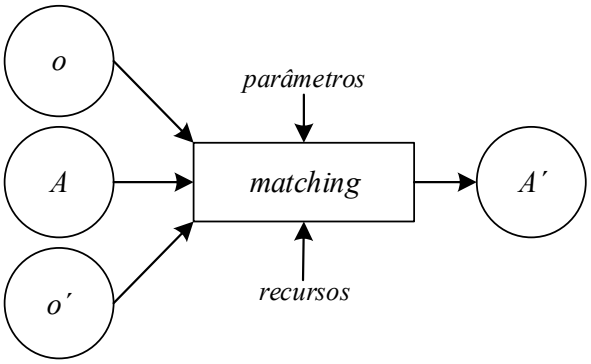
O alinhamento é o tipo de integração mais simples, visto que exige poucas mudanças nas ontologias origem. Assim, tem suporte a tipos limitados de interoperabilidade, sendo mais usado para recuperação de informação. Por sua vez, a compatibilidade parcial requer maiores modificações com o objetivo de suportar maior nível de interoperabilidade. A unificação ou compatibilidade total (fusão) exige mudanças extensas em ambas as ontologias, gerando maior interoperabilidade (SOWA, 2008).

O processo de combinação (*matching*) é o mais usual e o mais importante, visto que viabiliza outros tipos de integração. A operação de *matching* é usada para encontrar relacionamentos entre duas diferentes ontologias. A saída do processo de combinação é uma estrutura conhecida como alinhamento (*alignment*), que expressa um conjunto de correspondências (*correspondence*) entre as entidades de diferentes ontologias. Uma correspondência é a relação entre entidades de diferentes ontologias segundo um alinhamento de entrada específico. Essas entidades podem ser de diferentes tipos, ou seja, uma classe em uma ontologia pode corresponder a uma propriedade em outra (EUZENAT; SHVAIKO, 2013).

O processo de combinação pode ser definido como uma função f que, a partir de um par de ontologias o e o', um alinhamento de entrada A, um conjunto de parâmetros p e um conjunto de recursos r, retorna um alinhamento A' entre essas ontologias. Formalmente tem-se: $A' = f(o, o', A, p, r)$. Na **Figura 14** é mostrado o processo de combinação de ontologias.

Existem diversos sistemas que realizam o processo de *matching* de maneira automática e semiautomática por meio de abordagens diversas, que geralmente consistem em usar ontologias genéricas para vincular entidades de duas ontologias, a fim de aplicar algoritmos de mapeamento. Sistemas e abordagens dessa natureza podem ser vistos em Euzenat e Shvaiko (2013) e Nguyen e Conrad (2015).

Figura 14 - Diagrama sobre processo de combinação de ontologias.



Fonte: Euzenat e Shvaiko (2013).

Os alinhamentos podem ser expressos em diversas linguagens ontológicas. Por exemplo, na linguagem OWL os relacionamentos `owl:equivalentClass` e `rdfs:subClassOf` são usados respectivamente para expressar o relacionamento de equivalência e herança entre duas classes de diferentes ontologias. Usando o modelo de dados SKOS (*Simple Knowledge Organization System*), cujo objetivo é vincular sistemas de conhecimentos via Web Semática, os mesmos relacionamentos citados anteriormente poderiam ser expressos usando `skos:exactMatch` e `skos:broaderMatch`. O código da **Figura 15** traz um exemplo de alinhamento entre duas ontologias `onto1` e `onto2`, expressado em linguagem OWL.

Figura 15 - Fragmento de alinhamento entre duas ontologias representado em linguagem OWL.

```
01  <owl:Property rdf:about="&onto1;#author">
02     <owl:equivalentProperty rdf:resource="&onto2;#author"/>
03  </owl:Property>
04  <owl:Class rdf:about="&onto1;#Book">
05     <owl:equivalentClass rdf:resource="&onto2;#Volume"/>
06  </owl:Class>
07  <owl:Class rdf:about="&onto2;#title">
08     <owl:subClass ="&onto1;#name"/>
09  </owl:Class>
```

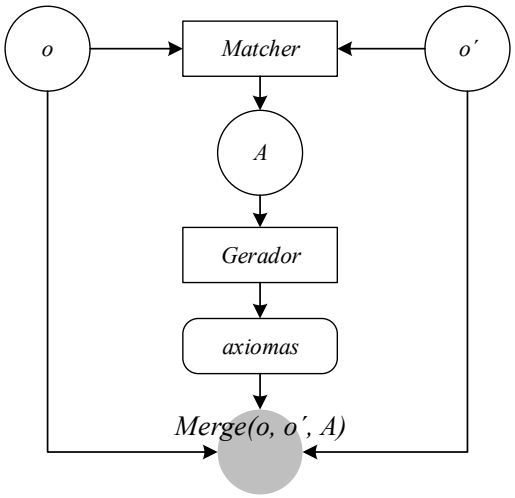
Fonte: Euzenat e Shvaiko (2013)

Como pode ser observado na **Figura 15**, as linhas 1, 2 e 3 representam uma correspondência entre duas propriedades das ontologias `onto1` e `onto2`. A propriedade

`author` de `onto1` possui correspondência com a propriedade `author` de `onto2`. As linhas 4, 5 e 6 representam uma correspondência entre as classes `Book`, de `onto1`, e volume, de `onto2`. Por fim, as linhas 7, 8 e 9 representam que a classe `title` de `onto2` corresponde a subclasse `name` de `onto1`.

Após a combinação, o alinhamento resultante pode ser usado para fazer a fusão (*merging*) de duas ontologias. A fusão consiste em obter uma nova ontologia o'' a partir do alinhamento A entre um par de ontologias o e o'. Se as duas ontologias são expressas na mesma linguagem, formalmente tem-se a seguinte função: $Merge\ (o,\ o',\ A) = o''$. Se as ontologias estiverem em linguagens diferentes, é necessário fazer a tradução a fim de realizar a fusão. Na **Figura 16** está ilustrado o processo de fusão, em que um gerador usa o alinhamento resultante do processo de combinação para gerar os axiomas de articulação, que constrói a ontologia.

Figura 16 - Diagrama sobre o processo de fusão (*merging*) de ontologias.



Fonte: Euzenat e Shvaiko (2015).

No contexto deste trabalho, a operação de *merging* é importante devido ao fato que a modelagem de processos é uma atividade contínua dentro as organizações, principalmente sob a óptica do BPM. Assim, à medida que os processos atuais são modificados e/ou melhorados, ou surjam novos processos, o conhecimento organizacional expressado por meio de ontologias é atualizado. Portanto, é necessário realizar a fusão entre a ontologia já existente e a nova, a fim de possibilitar a extração de requisitos de maneira automatizada com o objetivo de adequar o software o mais rápido possível.

Na *Seção 6.4* é apresentado um exemplo de *merging* entre duas ontologias de processos.

3.7 CONSIDERAÇÕES FINAIS

No contexto da modelagem de processos de negócio, apesar de expressivos no contexto do domínio, os diagramas da especificação BPMN oferecem uma visão de processos, limitada pela semântica da especificação da notação. Portanto, expressar modelos de processos de negócios por meio de ontologias proporciona muitas vantagens que a especificação de BPMN não possui: maior compreensão do modelo; inferência de novos conhecimentos; recuperação de informações estruturadas úteis aos mais diversos contextos, independentemente do tamanho da ontologia; complementação de informações e integração entre diferentes sistemas. Isso significa que a capacidade funcional dos modelos de processos de negócio é estendida por meio das ontologias.

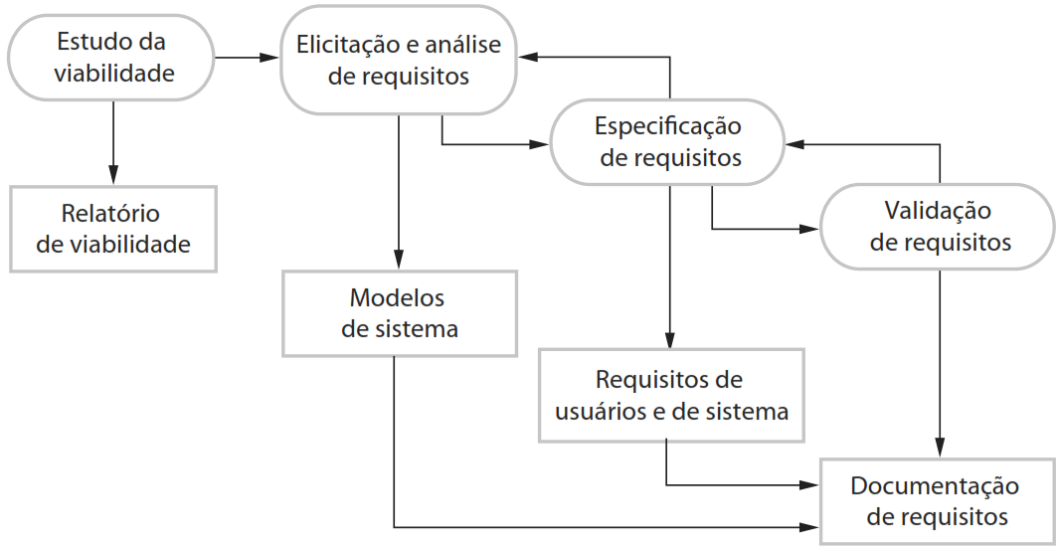
Assim, a especificação de requisitos de software a partir de ontologias representativas de modelos de processos de negócio se apresenta como uma técnica promissora para a formalização do conhecimento implícito e explícito em modelos de processos de negócio em BPMN.

Nesse sentido, as ontologias são estruturas flexíveis, com alto poder de expressividade em diversos cenários, visto que representam conhecimento de domínio. Essas características são importantes no contexto da Engenharia de Requisitos, que exige comunicação padronizada e formalização do conhecimento, objetivando o desenvolvimento de software mais aderente ao negócio e aos usuários.

4 ASPECTOS DA ENGENHARIA DE REQUISITOS ABORDADOS NO TRABALHO

A Engenharia de Requisitos é um processo que envolve diversas atividades multidisciplinares com o objetivo de adquirir conhecimento sobre um determinado domínio, a fim de estabelecer e manter os requisitos a serem atendidos por um sistema, software ou serviço. Trata-se de uma fase crítica em um projeto de software, visto que possíveis erros certamente causarão problemas na arquitetura e na implementação do sistema. Assim, a Engenharia de Requisitos deve ser conduzida sistematicamente por meio de modelos de processos bem definidos, visto que são muitas as variáveis e as incertezas que devem ser consideradas, as quais certamente impactarão nos demais processos de desenvolvimento do software (ISO, 2018; VAZQUEZ; SIMÕES, 2016). Uma visão geral das atividades da Engenharia de Requisitos, segundo Sommerville (2011b), podem ser vistas na **Figura 17**.

Figura 17 - Atividades da Engenharia de Requisitos.



Fonte: Sommerville (2011b).

Os requisitos consistem em descrições das funcionalidades do sistema, serviços a serem oferecidos aos usuários e restrições de funcionamento (ISO, 2018; SOMMERVILLE, 2011a; VAZQUEZ; SIMÕES, 2016). Tais requisitos devem ser descritos em pelo menos dois níveis de abstração: requisitos do usuário (alto nível) e do sistema. Os requisitos do usuário podem ser expressos como declarações em linguagem natural (ex.: textos) e como diagramas sobre o que o sistema deve fazer, considerando as restrições do sistema na visão do usuário. Os requisitos do sistema consistem em descrições detalhadas das funcionalidades do sistema que deverão ser implementadas.

Os requisitos são classificados em funcionais e não funcionais (ISO, 2018; SOMMERVILLE, 2011a; VAZQUEZ; SIMÕES, 2016). Os requisitos funcionais são as declarações dos serviços que o sistema deve oferecer, como ele deve reagir mediante entradas específicas e como ele deve se comportar em diferentes situações; em alguns casos os requisitos funcionais devem deixar claro o que o sistema não deve fazer. Por sua vez, os requisitos não funcionais contemplam restrições que os serviços do sistema devem considerar. Tais requisitos dizem respeito a características gerais do sistema como um todo, como, por exemplo: questões de segurança, restrições de uso, usabilidade e ergonomia, recursos de acessibilidade, *backup*, entre outros.

Os requisitos geralmente estão especificados em um documento chamado “Especificação Funcional” ou “Especificação de Requisitos de Software” (ERS) (PRESSMAN, 2011; SOMMERVILLE, 2011a; VAZQUEZ; SIMÕES, 2016). É recomendável que o documento ERS seja concebido em diferentes níveis de abstração, para possibilitar sua leitura pelos diversos perfis de profissionais interessados no software

Este trabalho tem como objetivo automatizar o processo de especificação de requisitos, pois extrai os requisitos dos usuários e do sistema a partir de uma ontologia representativa de modelos de processos de negócio. Assim, é tomado como premissa que as atividades de estudo de viabilidade, elicitação e análise de requisitos estão implícitas no conhecimento representado pela estrutura ontológica considerada no *Capítulo 5*.

Frente ao exposto, este capítulo traz questões relevantes para este trabalho considerando aspectos da Engenharia de Requisitos. A *Seção 4.1* aborda aspectos do documento ERS, de interesse ao trabalho. Na *Seção 4.2*, são abordados problemas e incertezas que ainda persistem na Engenharia de Requisitos, causados, geralmente, por fatores inerentes à comunicação e representação do conhecimento por parte das equipes envolvidas no desenvolvimento do software. Na *Seção 4.3*, são apresentados alguns conceitos e trabalhos relacionados à Engenharia de Requisitos apoiada por modelos de processos de negócio. Na *Seção 4.4*, são apresentadas abordagens que usam ontologias

para apoiar os processos da Engenharia de Requisitos. Por fim, a *Seção 4.5* apresenta as considerações finais do capítulo.

4.1 DOCUMENTO DE ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE

O documento geralmente denominado “Especificação de Requisitos de Software” (ERS) é a declaração oficial do que os desenvolvedores devem implementar, compreendendo os requisitos do usuário e os requisitos do sistema. Esse documento é um contrato entre os clientes e a equipe de desenvolvimento de software, visto que documenta de maneira completa todas as necessidades do cliente, objetivando receber aprovação de todos os envolvidos. Logo, o documento ERS é uma importante ferramenta de comunicação entre clientes e desenvolvedores, enquanto atesta que as equipes envolvidas e os usuários compreenderam os objetivos do software. Normalmente, o documento ERS contém informações como: propósito do software, público alvo, características do produto, classes de usuários, ambiente operacional, restrições de projeto e de implementação, interfaces do usuário, outras interfaces, documentação para usuários, etc. (ISO, 2018; VAZQUEZ; SIMÕES, 2016).

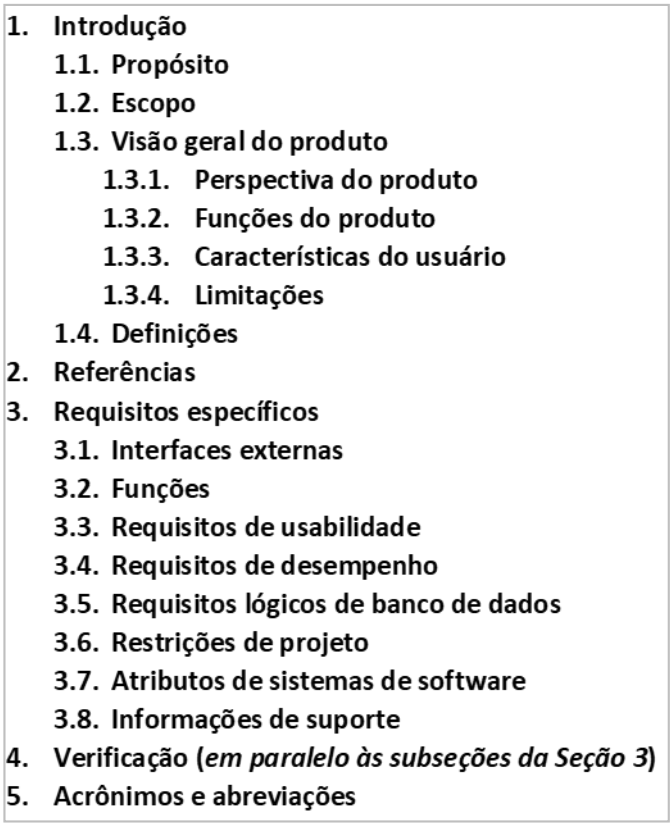
O nível de detalhamento do documento ERS pode variar de acordo com a natureza e o tamanho do sistema a ser desenvolvido. No caso de sistemas voltados a processos de negócio, o documento ERS deve ser bem detalhado, visto que as equipes envolvidas no projeto precisam das informações do negócio e dos sistemas. Portanto é necessária uma comunicação efetiva entre as partes envolvidas. Na **Figura 18** é mostrada a estrutura do documento ERS de acordo com o padrão ISO/IEC/IEEE 29148:2018 (ISO, 2018).

A documentação da especificação dos requisitos de um software é essencial a todo o projeto de seu desenvolvimento, principalmente quando a empresa de desenvolvimento é terceirizada, pois a equipe de TI será externa à organização e deve compreender os termos, conceitos, processos e necessidades próprios da empresa. Essa documentação pode definir as bases do contrato de prestação de serviços e guiar os processos de gerenciamento de projetos.

Na abordagem das metodologias ágeis, a documentação deve contemplar apenas partes essenciais do sistema, visto que os requisitos mudam rapidamente, causando desperdício de tempo, esforço e dinheiro na geração de um documento que poderá mudar assim que for concluído. Para isso, as seguintes características devem ser incorporadas no processo de especificação de requisitos convencional (INAYAT *et al*., 2015): 1) identificar o escopo do sistema por meio requisitos de alto nível, ou seja, como o sistema realmente

funcionará; 2) fazer com que os clientes e usuários participem ativamente do projeto focando na criação, análise e aceitação das estórias dos usuários; 3) tratar os requisitos como pilhas de prioridades; 4) adotar requisitos executáveis e documenta-los de maneira mais direta possível; 5) usar protótipos para validar requisitos; 6) adotar uma terminologia compreensível a todos os envolvidos; entre outras boas práticas.

Figura 18 - Estrutura do documento ERS, segundo o padrão ISO/IEC/IEEE 29148:2018.



Fonte: ISO (2018).

É importante ressaltar que no contexto deste trabalho, sempre que houver uma mudança no conhecimento explícito, a ontologia deverá ser atualizada. Isso deve possibilitar a extração de requisitos atualizados diretamente da fonte do conhecimento, alinhando este trabalho com a filosofia ágil em um certo grau. Nesse sentido, na *Seção 6.4* é apresentado um exemplo de integração entre duas ontologias, simulando uma mudança incremental em uma ontologia de processos.

4.2 PRINCIPAIS PROBLEMAS NA ENGENHARIA DE REQUISITOS

Alinhar um sistema de software com o negócio ao qual ele irá apoiar tem sido um grande desafio há muitos anos. A primeira dificuldade está em entender e modelar, com precisão, os processos de negócio, visto que há muitas variáveis envolvidas no que diz respeito às características do negócio. A segunda dificuldade está em criar uma especificação de requisitos de software que realmente suporte de maneira consistente os processos (PARK *et al*., 2017). De maneira geral, pode-se dizer que os problemas destacados por Christel e Kang (1992) desde o início dos anos 1990 ainda se mantêm:

- **Problemas de escopo** - os limites do sistema (o que ele deve e não deve fazer) são definidos de maneira deficiente ou, em muitos casos, possuem diversos detalhes técnicos desnecessários que confundem os objetivos gerais do sistema. Os envolvidos devem concentrar-se em "o que o sistema deverá fazer" e não em "como o sistema irá fazer";
- **Problemas de entendimento** - os usuários geralmente não estão certos sobre os objetivos do sistema; não possuem conhecimento sobre seu ambiente tecnológico; não possuem total conhecimento do domínio do problema; possuem problemas em transmitir suas necessidades aos engenheiros de sistemas; ocultam informações que, na visão deles, são óbvias; especificam requisitos conflitantes com outros usuários; especificam requisitos redundantes e/ou impossíveis de serem testados;
- **Problemas de volatilidade** - os requisitos podem mudar constantemente, seja por questões referentes ao ambiente econômico e político do negócio, ou por questões acerca do conhecimento dos envolvidos, da legislação, da tecnologia, do prazo, do orçamento, entre outros fatores.

Segundo Fernández et al. (2017), pesquisas da iniciativa NaPiRE entre 2014 e 2015, envolvendo empresas e profissionais de TI, levantaram problemas da Engenharia de Requisitos que "reforçam" os problemas destacados por Christel e Kang (1992), como apresentado na **Tabela 5**. A coluna "Freq. Geral" informa a frequência geral em que o problema foi citado. A coluna "Causa de Insucesso" mostra quantas vezes o problema foi citado por ter relação direta com o insucesso do projeto. As demais colunas mostram o número de vezes que o problema foi classificado (*ranking*) como os cinco problemas mais críticos segundo os entrevistados.

Tabela 5 - Principais problemas da Engenharia de Requisitos, segundo o programa NaPiRE.

Problema	Freq. Geral	Causa de insucesso	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
P1. Requisitos desconhecidos ou incompletos	109	43	34	25	23	17	10
P2. Falha de comunicação entre a equipe de projeto e o cliente	93	45	36	22	15	9	11
P3. Mudanças de objetivos, processos de negócios e/ou requisitos	76	39	23	16	13	12	12
P4. Especificação superficial de requisitos abstratos	76	28	10	17	18	19	12
P5. Prazo insuficiente / Gerenciamento do tempo	72	24	16	11	14	17	14
P6. Falha de comunicação entre os membros do time de projeto	62	25	19	13	11	9	10
P7. *Stakeholders* com dificuldades em separar requisitos de *design* de solução conhecida	56	10	13	13	12	9	9
P8. Suporte insuficiente por parte do cliente	45	24	6	13	12	6	8
P9. Requisitos inconsistentes	44	15	8	9	6	9	12
P10. Pouco acesso às necessidades do usuário e/ou informações sobre o negócio	42	16	7	10	8	8	9

Fonte: Fernández *et al.* (2017).

Todos os problemas apresentados na **Tabela 5** estão relacionados direta ou indiretamente com a dificuldade de extrair adequadamente os requisitos para fundamentar a especificação do software. Destaque deve ser feito aos problemas P1, P4, P8, P9 e P10, por evidenciarem a geração de incertezas provocadas por falta de esforços e/ou meios que permitam a especificação dos requisitos de forma consistente. A formalização do conhecimento corporativo possibilita o compartilhamento entre as equipes envolvidas em um projeto de software.

De fato, a Engenharia de Requisitos envolve essencialmente a transformação de conhecimento tácito (internalizado, baseado em experiências) em conhecimento explícito. Todavia, transformar descrições informais, geralmente representadas em linguagem natural e diagramas diversos, pode gerar diferentes interpretações e suscitar incertezas de várias naturezas, que podem ser potencializadas dependendo do nível de conhecimento dos envolvidos. Por isso, é importante a formalização do conhecimento por meio de uma linguagem que forneça uma interpretação única entre os interessados (ODEH, 2017).

No contexto deste trabalho, as incertezas podem ser mitigadas mediante a extração dos requisitos a partir do conhecimento implícito nas ontologias geradas a partir dos

modelos de processos de negócio. Convém destacar que uma das características principais das ontologias é justamente o fato de sua interpretação não depender do ponto de vista do usuário, visto que sua estrutura formal possui apenas uma interpretação (EUZENAT; SHVAIKO, 2013). Esse aspecto é importante sob o ponto de vista pragmático, pois proporciona uma única interpretação, independentemente do perfil de usuário.

4.3 ENGENHARIA DE REQUISITOS APOIADA POR MODELOS DE PROCESSOS DE NEGÓCIO

Com o objetivo de mitigar os problemas da Engenharia de Requisitos, a extração de requisitos de software a partir de modelos de processos de negócio em BPMN (de maneira manual ou automatizada) tem sido sugerida por diversos trabalhos na literatura, como Odeh (2017), Park *et al.* (2017), Bouzidi *et al.* (2017) e Nogueira (2017). Esses trabalhos consideram que os modelos de processos de negócio expressam gráfica e textualmente o ambiente organizacional, de maneira estruturada e funcional (conhecimento do domínio). Assim, é possível usar modelos de processos de negócio como fontes de informações para apoiar a especificação de requisitos. Essa abordagem é promissora, porém, ainda se percebe lacunas a serem preenchidas para melhorar a interação entre os modelos de processos de negócio e os engenheiros de software. Por exemplo, ainda não há um padrão de correspondência entre os modelos de processos de negócio e os modelos usados na Engenharia de Requisitos. Além disso, ainda é essencial a interveniência humana para se evitar perdas significativas de informação.

Nesse sentido, Odeh (2017) ressalta que o uso da notação BPMN na Engenharia de Requisitos deveria ser encorajado, destacando as seguintes vantagens da notação: (a) melhor compreensão do domínio pelos engenheiros de requisitos e outros membros da equipe de TI; (b) modelos mais expressivos e compartilháveis; (c) maior clareza, consistência, completude e corretude dos requisitos; (d) redução da necessidade de escrita e documentação informal.

Nessa direção, podem ser encontrados alguns trabalhos na literatura que usam essa abordagem de diferentes maneiras. Park *et al*. (2017), por exemplo, apresentam uma abordagem orientada a objetivos, que converte modelos em BPMN em casos de uso da linguagem UML, usando um modelo intermediário apoiado por ontologias. De maneira semelhante, Bouzidi *et al*. (2017) apresentam uma ferramenta denominada BPMN2UC (*BPMN To Use Case*), que aplica um conjunto de regras de transformação com base na equivalência semântica entre os elementos das especificações de BPMN e UML. A regra 1

(R1), por exemplo, consiste em mapear as raias (*swimlanes*) do mais baixo nível de aninhamento em atores do diagrama de casos de uso. Por sua vez, a regra 2 (R2) consiste em mapear as demais raias em pacotes. A regra 6 (R6) define que os *gateways* inclusivos e exclusivos entre dois elementos sejam mapeados em relações do tipo “extend”. As especificações das regras de transformação são definidas por meio da linguagem ATL (ATLAS *Transformation Language*). Outra característica interessante do trabalho desses autores é que também são geradas descrições textuais dos casos de uso. Convém destacar que os autores fazem recomendações de boas práticas de modelagem para que o mapeamento seja realizado de maneira satisfatória.

Nogueira (2017) define um conjunto de heurísticas denominadas “heurísticas de negócio”, com o objetivo de promover ajustes na parte textual e gráfica dos modelos de processos de negócio em BPMN. Tais ajustes objetivam melhorar o aspecto semântico e pragmático dos modelos e prepará-los para a extração automática de requisitos de software segundo o padrão ISO/IEC/IEEE 29148:2011, bem como para geração de diagramas UML. A extração de requisitos de software é realizada de maneira automática usando o sistema SRPD (*Software Requirements from Process Definitions*), desenvolvido por Nogueira (2017). O autor define uma escala de 1 a 4 para estabelecer o nível de expressividade dos elementos de modelagem:

1. **Sem documentação** - não há informação textual associada ao elemento;
2. **Rotulado** - apenas o nome do elemento está presente no diagrama;
3. **Documentado sintaticamente** - além do nome, o elemento apresenta informação textual válida segundo critérios definidos pelo autor;
4. **Completo** - o elemento está no nível 3 e apresenta informações adicionais segundo critérios definidos pelo autor usando a técnica 5W1H (*Who, What, When, Where, Why and How*). Caso o modelo não esteja no nível 4, é necessário aplicar as heurísticas de negócio para documentá-lo e ajustá-lo, pois senão a extração dos requisitos de software pelo sistema SRPD não será completa, com possibilidade de muitos requisitos não serem identificados.

As heurísticas de negócios são classificadas em quatro grupos com o prefixo HN (Heurística de Negócio): (1) HNe para eventos; (2) HNa para atividades; (3) HNg para fluxo de decisão; (4) HNd para artefatos e repositório de dados. As heurísticas de negócio utilizadas neste trabalho estão na **Tabela 6**. Essas heurísticas possuem uma anotação “T” e/ou “V” após sua definição, explicitando que a melhoria é aplicada na parte textual ou

visual. Se o ajuste for realizado pela ferramenta de modelagem, mas sem causar impacto visual no modelo, ou seja, uma mudança interna, então é marcada com “I”.

Tabela 6 - Heurísticas de negócio (HN) usadas neste trabalho.

Heurística	Descrição
HNa1	Cada atividade deve possuir a informação de seu executor. (I)
HNa3	O nome da atividade deve ser único no modelo (T/V)
HNe5	A informação sobre o início do evento deve ser validada (T)
HNa6, HNa9, HNg5, HNg11, HNa12	Se regra de negócio envolvida, o atributo estendido do tipo “REGRA” deve ser usado. (I/T)
HNd2 e HNd7	O repositório de dados e os artefatos devem ser relacionados às atividades que manipulam suas informações. (V)
HNd7	O repositório de dados ou artefato deve estar relacionado às atividades onde essas informações são manipuladas. (V)
HNd9, HNd10	- Os atributos do repositório de dados ou artefato devem ser registrados como atributos estendidos anotados como “ATRIBUTOS”. (I/T) - O tipo de dados de cada atributo deve ser informado junto aos atributos, conforme tabela de domínio. (I/T)

Fonte: Adaptado de Nogueira (2017).

Foram também definidas as chamadas heurísticas de requisitos (HR), que são usadas para extrair requisitos de software dos modelos de processos de negócio exportados para o formato XPDL. Essas heurísticas são compostas por três informações: o elemento de BPMN em que ela é aplicada, uma descrição desse elemento e as instruções de como extrair requisitos de software a partir do uso desse elemento. Assim, elas são organizadas em 3 categorias: (1) extração de casos de uso; (2) extração de requisitos textuais; (3) geração de diagramas UML. Cada heurística de requisitos é identificada usando a sigla “HR” seguida de um número sequencial, como mostrado na **Tabela 7** (NOGUEIRA, 2017).

De maneira geral, as heurísticas de negócio definem melhorias na parte textual (documentação) e gráfica dos modelos, enquanto as heurísticas de requisitos contribuem para que requisitos de software fiquem explícitos no modelo de processos de negócio. Assim, os modelos de processos de negócio em BPMN ficam expressivos e padronizados, contemplando aspectos da Engenharia de Requisitos.

É importante destacar que as heurísticas de negócio e as heurísticas de requisitos apresentadas nesta subseção são recomendadas diretamente por Figueiredo (2018), cujo trabalho é relevante na geração das ontologias a partir de modelos em BPMN, consideradas neste trabalho.

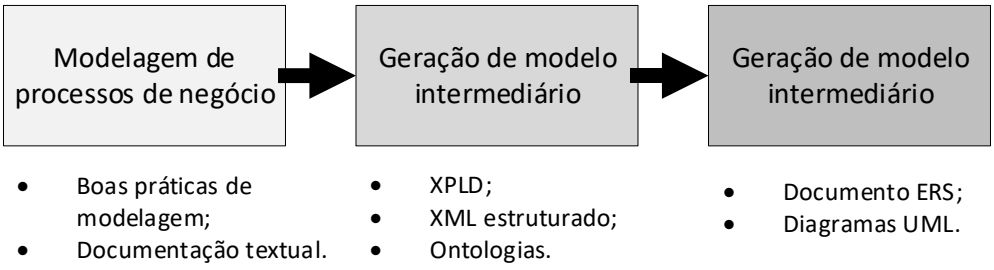
Tabela 7 - Heurísticas de requisitos (HR) usadas neste trabalho.

Heurística	Descrição
HR4	- As atividades devem ter seu tipo selecionado: “Usuário”, “Serviço” ou “Script”; - Identificar nas atividades os atributos estendidos do tipo “RNF”;
HR5	Identificar os fluxos de decisão; Identificar regras de negócio a partir da documentação desses fluxos;
HR6	- Identificar as atividades do tipo “Usuário”, “Serviço” ou “Script”; - Identificar regras de negócio a partir dos atributos estendidos do tipo “REGRA”;
HR8	- Identificar as atividades do tipo “Usuário”, “Serviço” ou “Script”; - Identificar requisitos funcionais a partir da documentação textual das atividades; - Identificar a existência de sistemas já utilizados a partir dos atributos estendidos do tipo “SISTEMA”.

Fonte: Adaptado de Nogueira (2017).

Outros trabalhos, como Heredia (2012) e Rhazali, Hadi e Mouloudi (2014), apresentam propostas para extração de requisitos funcionais e não funcionais de software a partir de modelos de processos de negócios em BPMN, devidamente documentados. Na **Figura 19**, pode-se observar o processo genérico utilizado pela maioria dos trabalhos voltados à extração de requisitos a partir de modelos em BPMN.

Figura 19 - Extração de requisitos de software a partir de modelos em BPMN.



Fonte: elaborado pelo autor.

Nessas abordagens, problemas de modelagem nos modelos de processos de negócios impactam diretamente na extração dos requisitos. Porém, não foram encontrados trabalhos que mostram de maneira explícita esses impactos. Nesse sentido, Park *et al*. (2017) destacam que ainda há diversas discussões a serem feitas, abordando os seguintes tópicos:

a) se os modelos de processos podem levar a diferentes interpretações, então pode haver transformações que se desviam do seu real significado;

b) se uma atividade de negócio pode ser realizada por uma pessoa ou por um sistema, então há problemas em se garantir que ambas forneçam o mesmo resultado;

c) o nível de detalhamento de um modelo de processos de negócio ou caso de uso não é necessariamente o mesmo de uma atividade que acontece no mundo real, ou seja, pode haver omissão de informações importantes nessa correspondência;

d) casos de uso em UML e modelos em BPMN não consideram requisitos não funcionais.

É possível notar que essas discussões evidenciam problemas na correspondência entre os modelos de processos de negócio e os modelos usados na Engenharia de Requisitos. Logo, problemas na modelagem de processos de negócios, principalmente considerando as qualidades semânticas e pragmáticas agravam esses problemas nos processos da Engenharia de Requisitos apoiada por modelos de processos de negócio, como mostrado na **Tabela 8**.

Tabela 8 - Impactos das incertezas em modelos em BPMN na Engenharia de Requisitos.

Incertezas nos modelos em BPMN		Impacto nos requisitos funcionais
Semântica	No nível visual (diagrama de processos)	• Incertezas a respeito do domínio • Incertezas no levantamento de requisitos • Incertezas na modelagem UML • Documento ERS incompleto
	No nível das anotações	• Problemas de reutilização • Problemas na compreensão por máquina • Problemas na extração automática de informações
Pragmática	Quanto à interpretação coletiva	• Problemas na etapa de validação dos requisitos

Fonte: elaborado pelo autor.

4.4 ENGENHARIA DE REQUISITOS APOIADA POR ONTOLOGIAS

Na literatura foram encontrados diversos trabalhos que mostram como as ontologias podem ser úteis no apoio ao processo de Engenharia de Requisitos para os mais diversos fins, desde a representação do conhecimento do domínio até a extração de requisitos de software. Segundo Dermeval *et al*. (2016), as principais aplicações das ontologias nas

Engenharia de Requisitos são: (1) redução da ambiguidade, inconsistência e incompletude dos requisitos; (2) representação do conhecimento do domínio com o objetivo de guiar a elicitação de requisitos; (3) apoiar o gerenciamento e a evolução dos requisitos.

Nesse sentido, Saito, Iimura e Aoyama (2015) propõem a criação de uma ontologia direcionada à Engenharia de Requisitos denominada REO (*Requirements Engineeering Ontology*). Essa ontologia apresenta uma classificação sistemática dos vocabulários presentes no domínio da Engenharia de Requisitos em seus diversos contextos. A ontologia REO unifica os corpos de conhecimento BABOK (*Business Analysis Body Of Knowledge*), SWEBOK (*Software Engineering Body Of Knowledge*) e REBOK (*Requirements Engineering Body Of Knowledge*), com o objetivo de permitir que os engenheiros de requisitos selecionem os vocabulários e termos mais adequados ao contexto durante a execução dos processos da Engenharia de Requisitos.

Por sua vez, Avdeenko e Pustovalova (2015) propõem a criação de uma ontologia para adequar a estrutura da Engenharia de Requisitos no que diz respeito aos termos utilizados no vocabulário para descrever conceitos de um domínio específico. O objetivo é fazer com que requisitos de software satisfaçam as propriedades de correção, completude e consistência. Além de criar um mecanismo de classificação de requisitos usando outros mecanismos existentes, as autoras sugeriram classes e relacionamentos com o objetivo de manter a coerência nos termos usados nos requisitos, permitindo a seleção de determinados tipos de requisitos adequados ao contexto, rápida transição entre diferentes tipos de documentos ERS e compartilhamento de conhecimento.

No contexto da transformação de modelos, Shunxin e Leijun (2010) propõem criação de uma ontologia de domínio a partir de requisitos funcionais representados em forma textual, com o objetivo de compartilhar e extrair conhecimento dentro do domínio. Os textos dos requisitos são divididos em termos, mapeados para uma ontologia específica e, então, são estabelecidos os respectivos relacionamentos entre os conceitos. Como exemplo, pode-se considerar o seguinte requisito de um sistema de biblioteca: “um aluno pode pedir emprestado dois livros digitais”. Então é possível criar uma ontologia contendo os conceitos “aluno” e “livro”. Entre esses dois conceitos há a relação “emprestar”, enquanto “digitais” pode ser uma propriedade do conceito “livro” e o número dois pode se referir a uma restrição de cardinalidade. Assim, foi criado manualmente, por meio da interpretação da ontologia, um diagrama de classes para representar graficamente as entidades e os seus respectivos relacionamentos.

De maneira semelhante, Mohamed, Ellatif e Farhan (2017) propõem a utilização de ontologias para a criação de mapas conceituais, com o objetivo de proporcionar maior

entendimento do domínio pelos envolvidos. Usando a ferramenta denominada Text2Onto (*Text to Ontology*), foi gerada uma ontologia a partir de uma especificação formal em formato de texto. Devido à quantidade de erros e imprecisões na ontologia gerada de maneira automática, foi aplicado um processo de refinamento manual por especialistas de domínio, por meio do sistema editor de ontologias *Protégé* (*https://protege.stanfor.edu*). Depois de validar a ontologia, foi aplicada uma ferramenta denominada CmapTools (*https://cmap.ihmc.us/cmaptools/*), com o intuito de transformá-la em um mapa conceitual. Após a geração do mapa conceitual, foi aplicado um método estatístico para verificar o grau de aceitação dos termos. No estudo apresentado, ficou evidente que a representação do domínio por meio do mapa conceitual é mais inteligível e compreensível por parte dos envolvidos. Neste caso, a ontologia serviu apenas como um modelo intermediário para gerar um modelo com maior nível de abstração.

Já o trabalho de Leshob (2016) propõe a extração entidades de negócio por meio de modelos de processos de negócios em BPMN baseado em uma ontologia de domínio de voltada a negócios intitulada REA (*Resource, Event, Agent*). Nessa proposta, a ontologia é usada para fornecer o conhecimento corporativo necessário para a construção dos modelos de domínio. Usando uma abordagem MDA, o processo é realizado em quatro etapas: (1) os modelos BPMN são anotados (o autor não detalhou como é realizado o processo de anotação) com o objetivo de distinguir os recursos, agentes econômicos e os contratos entre os agentes colaborativos, baseado na ontologia REA; (2) decompor os modelos de processos de negócio em processos de negócios elementares; (3) criar o modelo de domínio para cada processo elementar; (4) elaborar os modelos de domínio baseados em regras de transformação. No final do processo, é gerado o diagrama de classes. O autor implementou um protótipo para a realização da prova de conceito usando o EMF (*Eclipse Modeling Framework)* versão 2.11.1. Assim, foram gerados os objetos de negócio baseado na plataforma Java.

4.5 CONSIDERAÇÕES FINAIS

As atividades da Engenharia de Requisitos não devem ser realizadas apenas por um perfil profissional. Apesar de analistas e desenvolvedores de sistemas possuírem conhecimento de domínio em um certo grau, os especialistas de domínio, usuários e demais *stakeholders* devem ser envolvidos em todas as atividades, pois possuem conhecimento empírico e de domínio, que certamente serão úteis ao projeto de software. De fato, as equipes envolvidas e seus mais diversos perfis profissionais, operam em diferentes níveis

de abstração, usam diferentes notações e possuem diferentes expectativas e visões em relação ao sistema a ser desenvolvido. Assim, são necessários meios para prover a formalização do conhecimento de domínio de modo compartilhável e inteligível para todos os perfis de usuários, além de permitir que o conhecimento seja legível por máquina.

Nesse contexto, o padrão MDA oferece meios para a transformação de modelos PIM para PIM e, principalmente, PIM para PSM. Assim, os modelos de processos de negócio se mostram como uma rica fonte de informações de domínio, principalmente considerando as boas práticas de modelagem, proporcionadas, inclusive, pelas heurísticas de negócios. Tais heurísticas ampliam a capacidade expressiva dos modelos em BPMN, explicitando aspectos da engenharia de requisitos, úteis no contexto deste trabalho.

Portanto, a criação de um documento ERS aderente ao domínio e que satisfaça as necessidades dos usuários deve usar técnicas de apoio que auxiliem os envolvidos a manter uma comunicação padronizada, independentemente do nível de abstração. Os modelos de processos de negócio em BPMN, bem como as ontologias, se mostram como importantes recursos para apoiar a Engenharia de Requisitos, segundo o padrão MDA, mitigando os problemas apresentados na *Seção 4.2*. No próximo capítulo, é apresentado o processo sistematizado para a especificação de requisitos de software a partir de ontologias de processo.

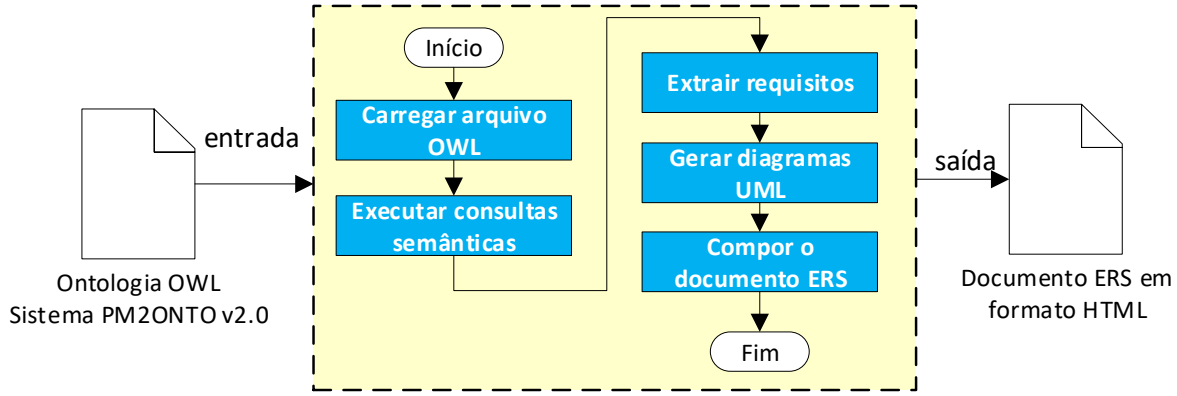
5 ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE A PARTIR DE ONTOLOGIAS DE PROCESSOS DE NEGÓCIO

Este capítulo apresenta o processo sistematizado para extração de requisitos de software a partir de ontologias de processos, mencionado na *Seção 1.1* como um dos principais objetivos deste trabalho. Estudos importantes para a fundamentação desse processo sistematizado foram apresentados nos *Capítulos 2* a *4*, contemplando aspectos relacionados ao trabalho sobre modelos de processos de negócios, ontologias e Engenharia de Requisitos. As ontologias são geradas pelo sistema PM2ONTO v2.0, com modificações significativas para o contexto deste trabalho em relação ao sistema domínio público (*freeware*) e fonte aberta PM2ONTO v1.0, de autoria de Figueiredo (2018). As ontologias são geradas em conformidade com a especificação de BPMN v2.0 e possibilitam meios de se extrair informações no contexto da Engenharia de Requisitos, usando a linguagem de consultas SPARQL. A partir dessas ontologias, o processo sistematizado proposto gera um documento conhecido como “Especificação de Requisitos de Software” (ERS), compatível com o padrão ISO/IEC/IEEE 29148:2018, apresentado na *Seção 4.1*. Assim, a intenção é contribuir com a mitigação de problemas nos processos da Engenharia de Requisitos, apresentados na *Seção 4.2*.

As boas práticas de modelagem na notação BPMN, incluindo o enriquecimento semântico com documentação textual, apresentadas no *Capítulo 2*, são determinantes para a geração das ontologias e, posteriormente, para a especificação dos requisitos dos softwares. Para que o processo sistematizado proposto funcione de maneira satisfatória, é necessário seguir as recomendações de boas práticas de modelagem e os critérios definidos por Figueiredo (2018), apresentados na *Subseção 2.3.2*, e, opcionalmente, as heurísticas de negócio e de requisitos de Nogueira (2017), apresentadas na *Seção 4.3*.

O processo proposto consiste nas seis etapas seguintes, como ilustrado na **Figura 20**:

Figura 20 - Visão geral do processo sistematizado para geração do documento ERS.



Fonte: elaborado pelo autor.

1. Carregar a ontologia OWL gerada pelo sistema PM2ONTO v2.0;
2. Executar consultas semânticas para recuperar informações da ontologia;
3. Extrair os requisitos de acordo com as informações recuperadas;
4. Executar os algoritmos para geração de diagramas UML (no momento: casos de uso e classes);
5. Compor o documento ERS;
6. Apresentar o documento em uma página Web.

Frente ao exposto, a *Seção 5.1* apresenta o sistema PM2ONTO v2.0, responsável por gerar as ontologias consideradas neste trabalho, abordadas, por sua vez, na *Seção 5.2*. O processo sistematizado proposto (**Figura 20**) é apresentado de modo mais refinado na *Seção 5.3*, sendo que a *Seção 5.4* aborda mais especificamente a etapa 4, relativa à geração dos diagramas de casos de uso e classes em UML. A *Seção 5.5*, por sua vez, apresenta o sistema OnToSRS (*Ontology to Software Requirements Specification*), desenvolvido neste trabalho para aplicar o processo proposto, usando *frameworks* e bibliotecas específicas para consultar ontologias e gerar diagramas UML. Por fim, a *Seção 5.6* apresenta as considerações finais deste capítulo.

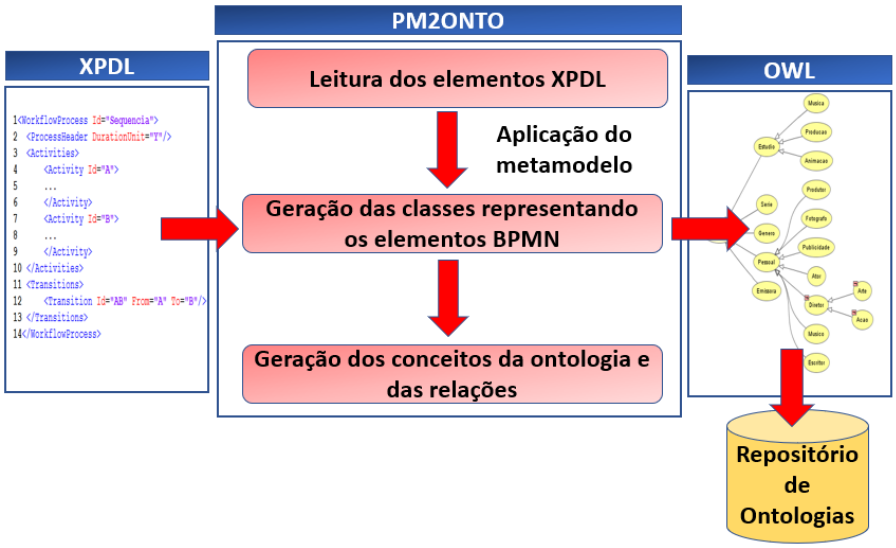
5.1 SISTEMA PM2ONTO

As ontologias consideradas neste trabalho são geradas automaticamente pelo sistema PM2ONTO v2.0, baseado na versão 1.0 de autoria de Figueiredo (2018). As

alterações feitas sobre a versão 1.0 possibilitaram representar, na ontologia, atributos de objetos e depósitos de dados, que são essenciais para a criação de diagramas de classes com mais conteúdo extraído diretamente da ontologia, como abordado na *Seção 5.4*.

A entrada do sistema PM2ONTO v2.0 (**Figura 21**) é um modelo de processos de negócio, documentado textualmente e ajustado de acordo com doze critérios definidos por Figueiredo (2018) (ver *Subseção 2.3.2*). Além desses critérios, podem ser incluídos os princípios de modelagem e documentação textual definidos por Nogueira (2017), chamados de "heurísticas de negócio" (HN) e "heurísticas de requisitos" (HR).

Figura 21 - Visão geral do sistema PM2ONTO v2.0.



Fonte: Figueiredo (2018).

Para este trabalho, o atributo estendido "ATRIBUTOS" (HNd9 e HNd10 - ver *Seção 4.3*, **Tabela 6**) deve ser adicionado à documentação dos atributos de dados referente aos símbolos de modelagem do tipo "Repositório de dados" e "Objeto de dados". Outros dois atributos estendidos relevantes na documentação do modelo são "RNF" e "REGRA" (HR4 e HR6 - ver *Seção 4.3*, **Tabela 7**). Esses dois últimos atributos estendidos, objetivam documentar textualmente os requisitos não funcionais e as regras de negócio.

Também foi necessário definir duas condições (COND) a serem seguidas durante a modelagem em BPMN para que o algoritmo apresentado na *Subseção 5.4.1* consiga processar a ontologia de processos corretamente:

COND-1: evitar *gateways* conectados diretamente;

COND-2: todos os subprocessos devem ser do tipo simples, sem tipo definido. Neste trabalho, o subprocesso simples é semanticamente suficiente para a extração dos requisitos de software.

Assim, antes de usar o sistema, recomenda-se avaliar se o modelo de processos de negócio deve ser ajustado, para ficar mais expressivo textualmente e graficamente, de acordo com os doze critérios apresentados e com as condições COND-1 e COND-2. Então, o modelo em BPMN deve ser convertido para a linguagem XPDL v2.2 - o que é feito automaticamente pelo sistema de modelagem *Bizagi Modeler* e outros sistemas similares. O modelo convertido em XPDL é a entrada para o sistema PM2ONTO v2.0. A ontologia gerada é então armazenada em um repositório do sistema.

A estrutura da ontologia gerada é baseada no metamodelo Meta2Onto, também de autoria de Figueiredo (2018), que define uma hierarquia de classes, bem como restrições de relacionamentos entre elas. Também são criadas relações entre classes e instâncias baseadas na especificação BPMN v2.0, objetivando manter a compatibilidade semântica entre o modelo e a ontologia. Vale destacar que o metamodelo Meta2Onto não contempla a totalidade da especificação BPMN v2.0. Entretanto, é determinante para a elaboração das consultas semânticas em SPARQL, objetivando recuperar conhecimento adequado para o contexto dos diagramas de casos de uso e de classes, considerados neste trabalho.

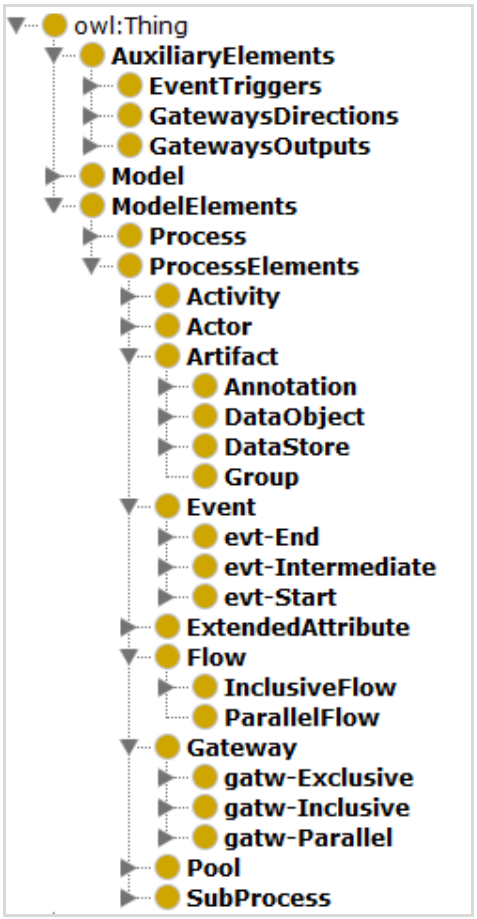
O sistema PM2ONTO também possui um subsistema que possibilita a criação e execução de consultas semânticas sobre a ontologia, com uma interface simples para o usuário. Assim, é possível obter informações relevantes sobre o domínio e as interdependências entre as atividades e processos do modelo.

5.2 ESTRUTURA DA ONTOLOGIA DE PROCESSOS

Na **Figura 22**, é apresentada uma pequena parte da hierarquia de classes ontologia gerada pelo sistema PM2ONTO v2.0. A hierarquia das classes é determinada pelo axioma *subClassOf*, que relaciona duas classes numa relação de generalização/especialização. Por exemplo, a classe *ModelElements* é a superclasse que generaliza os elementos ou símbolos de modelagem. Convém enfatizar que cada subclasse da ontologia contempla a semântica de cada um dos símbolos do modelo em BPMN. Assim, a classe *Activity* possui subclasses para cada tipo de atividade na notação BPMN: envio, recebimento, usuário, manual, regra de negócio, serviço e *script*. De modo similar, a classe *Gateway* possui subclasses para

cada tipo específico de *gateway* em BPMN: exclusivo, baseado em evento, paralelo, baseado em evento paralelo, inclusivo e complexo. Portanto, cada elemento do modelo possui classes correspondentes na ontologia gerada, caracterizando um mapeamento “um para um” (1:1).

Figura 22 - Categorização das classes para a ontologia.



Fonte: Figueiredo (2018).

De fato, o principal axioma usado na estrutura ontológica considerada é *subClassOf*, visto que determina a hierarquia de classes da ontologia. Entretanto, outros axiomas são considerados, como a propriedade *inverseOf*, que determina relações inversas entre as relações *isPerformedBy* e *isExecutorOf.* A relação *isPerformedBy* pertence aos domínios *SubProcess* e *Activity*, ou seja, só pode estar presente em subclasses desses domínios. Por fim, o predicado da relação *isPerformedBy* deve ser uma subclasse de *Actor* (propriedade de objeto “*range*”). Ademais, um conjunto de atributos foi definido para documentar características básicas dos elementos, sendo os mais comuns: identificador, nome e descrição.

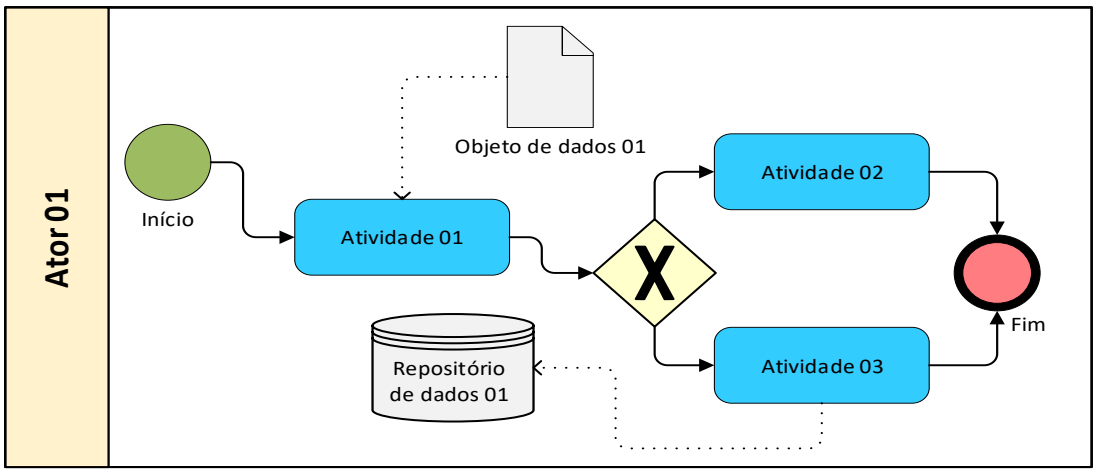
As relações/axiomas definidas na ontologia são importantes para a identificação dos elementos que compõem o documento ERS, além de possibilitarem a geração dos diagramas de casos de uso e de classes. A seguir são apresentadas as relações/axiomas usadas neste trabalho:

- ***isPerformedBy*** - relação entre uma instância de atividade (classe *Activity*) e o seu executor (classe *Actor*), possibilitando identificar os atores e seus respectivos casos de uso (restrição de quantificação - *subclassOf(Activity)* → *IsPerformedBy <allValuesFrom>* → *subClassOf(Actor)*);
- ***IsExecutorOfActivity*** - relativo às atividades que o ator executa (relação inversa da relação *IsPerformedBy*). Essa relação possui restrição de quantificação *allValuesFrom*, que restringe sua relação apenas com subclasses de *Activity* e *SubProcess*;
- ***isPartOfSubProcess*** e ***isPartOfProcess*** - definem se um elemento de modelagem faz parte de um determinado subprocesso ou do processo principal, respectivamente. Possuem restrição de quantificação (*allValuesFrom*), restringindo a relação apenas com subclasses de *SubProcess*;
- ***isPrecededBy*** e ***isSucceededBy*** - definem a relação de precedência e sucessão entre os elementos do modelo, respectivamente. Essas relações são determinantes para a geração do diagrama de casos de uso. Também possuem restrição de quantificação (*allValuesFrom*);
- ***UsesInput*** e ***ProducesOutPut*** - representam os dados de entrada (persistentes ou não) e de saída de uma atividade, respectivamente; esses dados são relacionados com instâncias das subclasses. Possuem restrição de cardinalidade (*qualifiedCardinality*), explicitando exatamente as relações com subclasses de *Artifact*;
- ***hasExtendedAttribute*** and ***isExtendedAttribute*** - duas relações inversas que determinam, respectivamente, se uma instância possui atributo estendido e se a propriedade é um atributo estendido. O conteúdo dos atributos estendidos consiste nas seguintes informações de heurísticas: (1) “REGRA” (HR6), para regras de negócio; (2) “RNF” (HR4), para requisitos não funcionais. Possuem restrições de cardinalidade *allValuesFrom* e *someValuesFrom*.

Com o objetivo de compreender a relação entre o modelo BPMN e a ontologia definida, é necessário observar as **Figuras 23** e **24**. Na **Figura 23**, é mostrado um simples

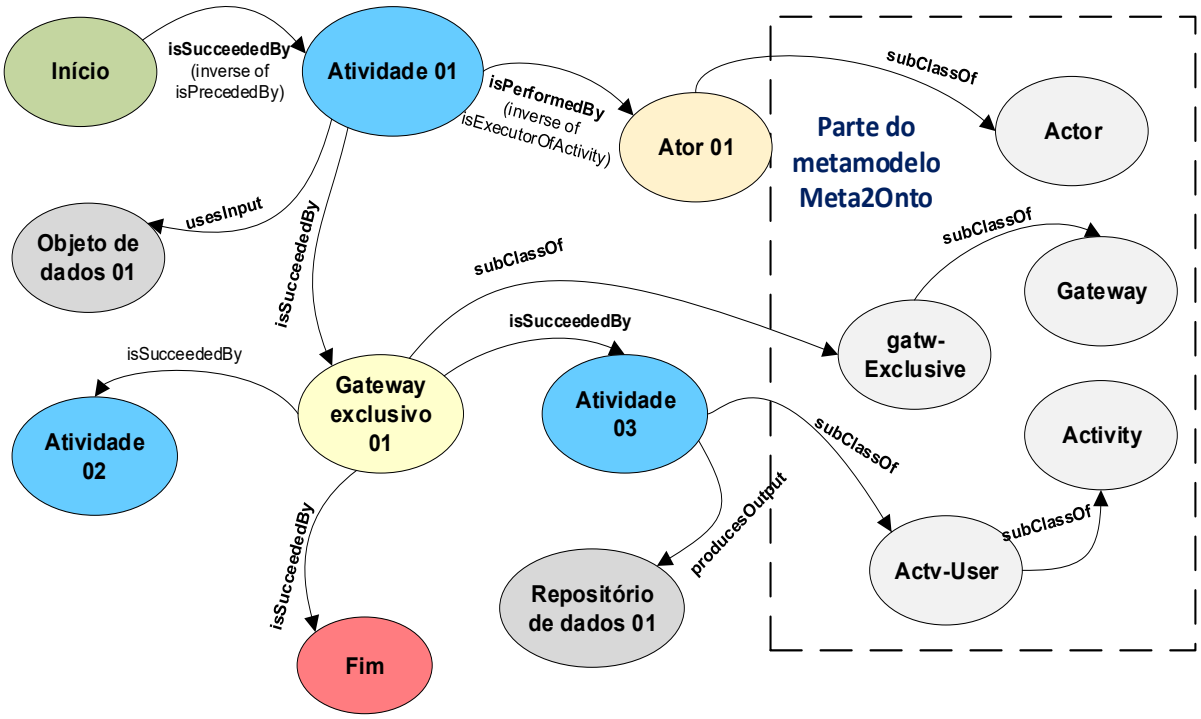
modelo genérico em BPMN. Por sua vez, na **Figura 24**, é mostrada a respectiva representação ontológica, em que é possível observar que as relações de fluxo de sequência (*isPrecededBy* e *isSucceededBy*), relação entre ator e atividade (*isPerformedBy* e *isExecutorOfActivity*), relações com objetos e depósitos de dados (*producesOutput* e *usesInput*) e subclasses (*subClassOf*) são mapeados para a ontologia de maneira a preservar a semântica do modelo.

Figura 23 - Modelo simples na notação BPMN para exemplificar a representação ontológica.



Fonte: elaborado pelo autor.

Figura 24 - Representação ontológica do modelo em BPMN mostrado na Figura 23.



Fonte: elaborado pelo autor.

O mapeamento é do tipo “1:1”, ou seja, cada elemento do modelo em BPMN possui um correspondente na ontologia. Na **Figura 24**, também é mostrado parte do metamodelo Meta2Onto no retângulo com linha tracejada destacado à direita.

Além dos elementos apresentados nas **Figura 23** e **24**, a ontologia gerada pelo sistema PM2ONTO v2.0 possui propriedades de dados importantes para extrair informações sobre a ontologia. Essas propriedades são mapeadas diretamente no modelo BPMN de origem, a partir da documentação textual composta por atributos básicos (nativos do sistema modelador) e atributos estendidos, determinados segundo os critérios definidos por Figueiredo (2018). No contexto deste trabalho, as principais propriedades de dados são:

- ***i1-id*** - identificador do elemento;
- ***i2-name*** - texto contendo o nome do elemento;
- ***i3-description*** - texto contendo a descrição textual do elemento;
- ***i4-documentation*** - texto contendo a documentação do processo;
- ***i6-isFunctionalRequirement*** - valor booleano que determina se o elemento é um requisito funcional;
- ***i7-isNonFunctionalRequirement*** - valor booleano que determina se o elemento é um requisito não funcional (HR4);
- ***i8-isBusinessRule*** - valor booleano que determina se um elemento possui uma regra de negócio (HR6).

Após analisar a ontologia gerada, foi possível identificar algumas inconsistências na a geração dos diagramas UML. Ao comparar a semântica modelo BPMN com a ontologia, foram identificadas relações inexistentes, sendo necessária a inclusão de modo manual, usando um editor de ontologias. Nos testes realizados, foi observado que, nos subprocessos, as relações *usesInput* e *producesOutput* não foram criadas entre as classes *Activity* e *Artifact*. Ademais, foram criadas relações que não existem no modelo de origem, como alguns atores (classe ontológica *Actor*) que fazem parte de subprocessos. Essas relações tiveram que ser incluídas manualmente, além de determinar as restrições definidas no metamodelo Meta2Onto. É importante destacar que esses problemas foram corrigidos manualmente editando a ontologia. Portanto, para evitar esses problemas, é necessário que o sistema PM2ONTO seja atualizado, de modo a fazer automaticamente o que foi feito de modo manual.

5.3 PROCESSO DE GERAÇÃO DO DOCUMENTO ERS

O documento ERS proposto neste trabalho segue as recomendações do padrão ISO/IEC/IEEE 29148:2018 (ver *Seção 4.1*), sendo composto por descrição, classes de usuários, comportamentos, entidades do mundo real e funcionalidades. O documento ERS também possui diagramas UML de casos de uso e de classes, que, segundo OMG (2015), representam, respectivamente, aspectos comportamentais e estruturais de um sistema proposto. Como não há correspondência direta entre os elementos da ontologia e os elementos do documento ERS, foram definidas regras de mapeamento semântico entre eles. Na **Tabela 9**, são mostrados os campos do documento (1ª coluna), bem como os elementos de origem na ontologia (2ª coluna).

Tabela 9 - Estrutura do documento ERS e correspondentes semânticos na ontologia.

Campos do documento ERS	Elementos ontológicos de origem (Classe - Propriedade - Relação)
1. Nome	*subClassOf* Model - propriedade *i2-name*
2. Descrição	*subClassOf* Model - propriedade *i3-description*
3. Data de criação	*subClassOf* Model - propriedade *i6-creationDate*
4. Data de modificação	*subClassOf* Model - propriedade *i7-modificationDate*
5. Autor	*subClassOf* Model - propriedade *i5-author - subClassOf Actor*
6. Documentação	*subClassOf* Model - propriedade *i4-documentation*
7. Atores	*subClassOf* Actor - propriedades *i2-name* e *i3-description*; relação *isExecutorOf* - define a relação entre atores e casos de uso
8. Precondições	*subClassOf* evt-Start (evento de início) - propriedades *i2-name* e *i3-description*
9. Pós-condições	*subClassOf* evt-End (evento de fim) - propriedades *i2-name* e *i3-description*
10. Regras de negócio	*subClassOf* Activity, *subClassOf* Gateway, *subClassOf ExtendedAttribute* - propriedades *i2-name* e *businessRule*
11. Requisitos funcionais	*subClassOf Activity* - propriedade *isFunctionalRequirement* = true;
12. Requisitos não-funcionais	*subClassOf ExtendedAttribute* - propriedade *i7-isNonFunctionalRequirement* = true
13. Diagrama de casos de uso	*Indefinido*
14. Diagrama de classes	*Indefinido*

Fonte: elaborado pelo autor.

Como observado na **Tabela 9**, todos os campos do documento definido possuem um correspondente semântico na ontologia, com exceção dos campos 13 e 14, que são os diagramas de casos de uso e de classes. Para o mapeamento dos campos 1 a 13, são executadas consultas SPARQL relativamente simples, segundo a estrutura ontológica definida no metamodelo Meta2Onto. Por exemplo, para recuperar os atores (campo 7), basta selecionar todas as instâncias que são subclasses da classe ontológica *Actor*. De maneira semelhante, para recuperar a precondição e as pós-condições (campos 8 e 9), é necessário selecionar todas as instâncias que são subclasses de *evt-Start* (evento de início) e *evt-End* (evento de fim), respectivamente. Para recuperar as regras de negócio (campo 10), é preciso consultar as instâncias que são subclasses da classe ontológica *Gateway*, mais as instâncias que são subclasses de *Activity*; todas as instâncias devem possuir a propriedade de dados *isBusinessRule* igual a verdadeiro (*true*). A geração dos diagramas em UML exige processamento de dados por meio de algoritmos que serão apresentados nas *Subseções 5.4.1* e *5.4.2*, respectivamente.

Assim, o processo de elaboração do documento ERS é composto de três etapas:

1. **Extração de informações básicas da ontologia –** requer execução de consultas SPARQL para a recuperação das informações correspondentes aos campos 1 a 6 da **Tabela 9**;
2. **Extração de requisitos do software –** requer execução de consulta SPARQL para a recuperação das informações correspondentes aos campos 7 a 12 da **Tabela 9**;
3. **Geração de diagramas UML –** requer execução de consultas em SPARQL considerando as relações do metamodelo Meta2Onto (ver *Seção 5.2*), aplicando regras de mapeamento para gerar os diagramas de caso de uso e de classes.

A aplicação dessas etapas sobre uma ontologia (arquivo OWL) pode gerar mais de um documento ERS, considerando a relação *isPartOfSubProcess*, que define classes e relações que fazem parte de um ou mais subprocessos. Um subprocesso é dependente do processo principal, porém ele agrupa eventos, atividades, *gateways* e outros elementos de modelagem, possuindo seu próprio fluxo de execução. Na ontologia gerada, os subprocessos são mapeados para classes filhas da classe *SubProcess*, de acordo com a especificação da notação BPMN (Reutilizável, Transacional, Adhoc, Loop, Múltiplas Instâncias de Sequência e Paralelas, Simples, Evento). Assim, objetivando manter a coerência semântica entre o modelo de processos de negócio em BPMN original e o documento ERS, o processo de geração desse documento resulta em *1+S* documentos,

onde “1” se refere ao processo principal e “*S”* se refere ao número de subprocessos existentes.

5.4 GERAÇÃO DE DIAGRAMAS UML

Assim como apresentado na seção anterior, não há correspondência direta entre os elementos da ontologia e os elementos dos diagramas UML usados na composição do documento ERS. Portanto, é necessário definir mapeamentos semânticos que associem elementos da ontologia aos elementos dos diagramas. Assim, esta seção apresenta os mapeamentos semânticos necessários para a criação dos diagramas de casos de uso (*Subseção 5.4.1*) e de classes (*Subseção 5.4.2*).

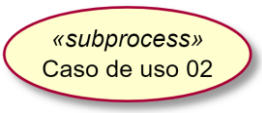
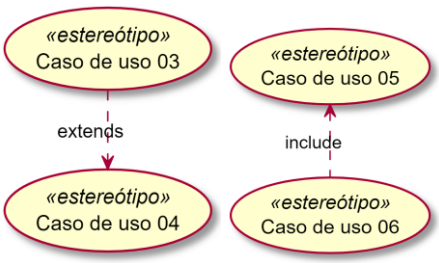
5.4.1 GERAÇÃO DE DIAGRAMAS DE CASOS DE USO

Segundo Lima (2011), a visão de caso de uso mostra o sistema sob a perspectiva do usuário e pode ser usada como um contrato entre cliente e desenvolvedor. Assim, o diagrama de casos de uso é usado para representar as principais funcionalidades do sistema, servindo de base para as demais visões e modelos. No contexto da transformação de modelos, o diagrama de casos de uso não consegue expressar todos os aspectos de um modelo de processos de negócio, principalmente o fluxo de execução das atividades. Entretanto, ele consegue expressar os aspectos comportamentais do sistema, possibilitando apoiar as equipes de TI nos processos de especificação de requisitos de software, sendo útil na modelagem dos atores e seus papeis, comportamentos e requisitos funcionais (BOOCH; RUMBAUGH; JACOBSON, 2005; OMG, 2015).

A fim de possibilitar uma aproximação semântica entre os elementos ontológicos e os elementos dos diagramas de casos de uso, foi definido um conjunto de regras de mapeamento semântico, mostrado na **Tabela 10**. Assim, os casos de uso representam atividades e subprocessos. Os relacionamentos entre casos de uso (*include* e *extend)* são usados para representar a semântica dos fluxos de execução das atividades do modelo em BPMN. Isso foi definido porque um relacionamento do tipo *include* entre dois casos de uso implica que uma instância do primeiro caso de uso conterá obrigatoriamente o comportamento definido pelo segundo (BOOCH; RUMBAUGH; JACOBSON, 2005; LIMA, 2011). Por outro lado, um relacionamento do tipo *extend* define que uma instância do primeiro caso de uso conterá opcionalmente, ou de acordo com regras definidas, o

comportamento do segundo. Portanto, relacionamentos do tipo *include* representam os fluxos de execução direta entre duas atividades, além de *gateways* paralelos e inclusivos. Por sua vez, relacionamentos do tipo *extend* representam fluxos de execução que envolvem *gateways* exclusivos.

Tabela 10 - Mapeamento semântico entre elementos da ontologia e do diagrama de casos de uso.

	Classe/relação ontológica	Elemento do diagrama	Representação gráfica no diagrama
1	*subClassOf:Actor* e propriedade *i2-name*	Atores e seus respectivos nomes	
2	*subClassOf:Activity* e propriedade *i2-name*	Caso de uso estereotipado de acordo com sua superclasse ontológica	
3	*Activity-isPerformedBy-Actor;* ou *Actor-isExecutorOfActivity-Activity*	Cria uma relação entre um ator e um caso de uso:	
4	*subClassOf:subProcess*	Caso de uso estereotipado como <<*subprocess*>>. Cada subprocesso gera outro documento ERS, logo, outro(s) diagrama(s) de caso de uso	
5	*subClassOf:Gateway* e instâncias sucedentes *(*relação *isSucceededBy)*	*Gateways* exclusivos geram relacionamentos do tipo *extend*; *gateways* do tipo inclusivo e paralelo geram relacionamentos do tipo *include.*	

Fonte: elaborado pelo autor.

Identificadas as classes ontológicas e suas respectivas relações no contexto do diagrama de casos de uso, os seguintes passos devem ser executados para a geração desse diagrama:

1) Todas as instâncias da classe *Actor* devem ser obtidas por meio de uma consulta SPARQL;

2) Para cada instância da classe Actor deve ser executado um desses passos:

a) recuperar todas as instâncias da classe *Activity* relacionadas com classes do tipo *Actor* por meio da tripla *Activity* → *isPerformedBy* → *Actor*;

b) recuperar todas as instâncias da classe *Activity* relacionadas por meio da relação inversa: *Actor* → *isExecutorOf* → *Activity*;

3) Considerar como "caso de uso" toda instância que é subclasse de *Activity*, com exceção das que correspondem à atividade do tipo *Manual*;

4) Considerar como "caso de uso" toda instância de classes *SubProcess*;

5) As instâncias das classes ontológicas *Gateway* são mapeadas para relacionamentos entre casos de uso do tipo *extend* (*gateway* exclusivo) e *include* (*gateway* do tipo inclusivo ou paralelo).

Com base nos cinco passos definidos, o algoritmo em linguagem natural, mostrado na **Figura 25**, deve ser executado *1+S* vezes, ou seja, uma vez para o processo principal e S vezes, uma para cada instância da classe ontológica *SubProcess*. Assim, o diagrama de casos de uso é capaz de representar atores, casos de uso propriamente ditos e relacionamentos entre eles (*include* e *extend*).

Figura 25 - Algoritmo para geração do diagrama de casos de uso.

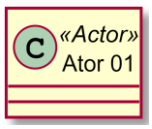
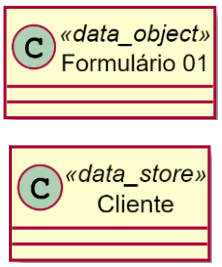
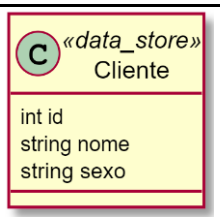
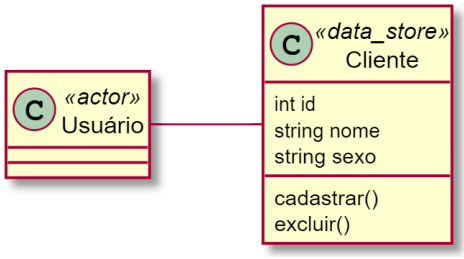
```
recuperar todas as instâncias de Actor em Actors
para (cada Actor) {
  recuperar todas as instâncias de Activity em Activities
    para (cada Activity) {
      definir um caso de uso "Actor → Activity"
      recuperar o elemento sucessor de Activity em S
      se (S = gateway exclusivo) {
         recuperar todos os elementos sucessores de S em SE
         para (cada SE) definir um caso de uso "Activity ← SE" (extend)
      }
      se (S = gateway inclusivo ou S = gateway paralelo) {
         recuperar todos os elementos sucessores de S em SI
         para (cada SI) definir um caso de uso "Activity → SI" (include)
      }
      se (S não é um gateway e S não é um evento)
         definir um caso de uso "S → Activity" (include)
    }
}
```

Fonte: elaborado pelo autor.

5.4.2 GERAÇÃO DE DIAGRAMAS DE CLASSES

O diagrama de classes representa a modelagem estrutural e a relação entre as classes que fazem parte do sistema. Assim, esse diagrama possui grande importância no contexto da especificação de requisitos, visto que expressa características estruturais no sistema modelado (BOOCH; RUMBAUGH; JACOBSON, 2005; OMG, 2015). Na notação BPMN, os artefatos denominados "objetos" e "repositórios de dados" (*Data Object* e *Data Store*) expressam dados e informações que são usados ou produzidos durante a execução dos processos. Ademais, os executores dos processos (Atores) também são entidades que fazem parte do sistema. Assim, para representar semanticamente esses elementos das ontologias em diagramas de classes, foi definido o mapeamento apresentado na **Tabela 11**.

Tabela 11 - Mapeamento semântico entre elementos das ontologias e dos diagramas de classes.

	Classes/relações ontológicas	Elemento do diagrama	Representação gráfica no diagrama
1	*subClassOf:Actor* e propriedade *i2-name*	Classe nomeada e com estereótipo "*Actor*".	
2	*subClassOf: DataObject; subClassOf: DataStore* e propriedade *i2-name*	Classe nomeada e com estereótipo "*data_object*" ou "*data_store*".	
3	*subClassOf:Artifact* e *ExtendedAttribute*	Atributos de dados e seus respectivos tipos (*int*, *String*, etc).	
4	*subClassOf:Activity; subClassOf: Actor;* relações *isPerformedBy, usesInput* e *producesOutput*	Associações entre classes e definição de métodos.	

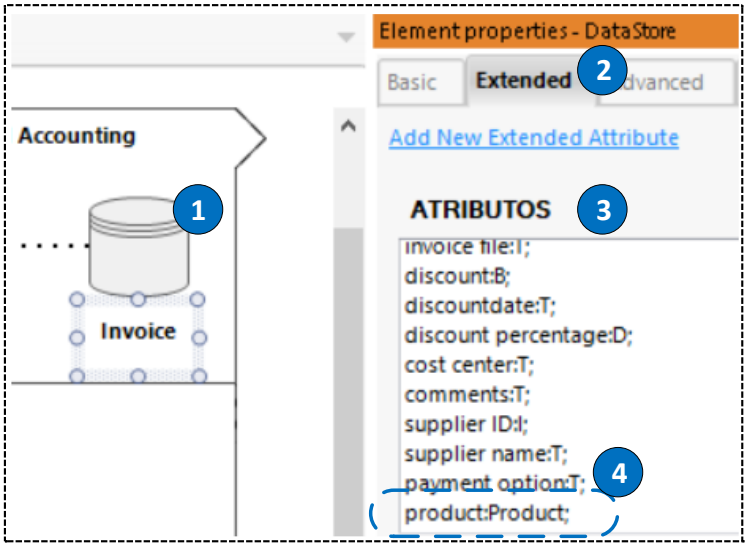
Fonte: elaborado pelo autor.

A expressividade dos diagramas de classes está limitada ao conhecimento representado na ontologia. Logo, não há como identificar elementos como herança, cardinalidade, composição e agregação, visto que essas informações não estão presentes na estrutura ontológica. Entretanto, as relações entre classes definidas pelo relacionamento do tipo "associação" (linha 4 da **Tabela 11**), quando duas classes estão conectadas, são identificadas, semanticamente, segundo as regras de mapeamento (RM) definidas a seguir:

- **RM1 - relações entre classes existentes na ontologia:** uma instância de uma subclasse de *Actor* está relacionada com uma instância de subclasse de artefato (*Data Store* ou *Data Object*) quando ambas as instâncias estão relacionadas com a mesma atividade. Essa regra é definida em termos de triplas da seguinte maneira: Activity "X" → *isPerformedBy* → Actor "A" e (Activity "X" *usesInput* (DataStore "D" ou DataObject "D") ou (Activity "X" *producesOutput* (DataStore "D" ou DataObject "D"))). Logo, as classes em UML estereotipadas como *Actor* se relacionam com classes estereotipadas como *DataStore* ou *DataObject*, por meio das instâncias das subclasses de *Activity*;
- **RM2 - relações entre classes definidas por atributos:** considerando que os atributos de dados no modelo de processos de negócio foram documentados, é possível definir tipos primitivos e relacionamentos entre classes existentes e novas classes;
- **RM3 - funcionalidades (métodos) das classes**: as subclasses ontológicas de *Actor* são mapeadas para atores no diagrama de classes. Logo, as atividades executadas por cada ator, com exceção das atividades do tipo "Manual", são consideradas métodos nas classes do diagrama. Essa regra é definida em termos de triplas da seguinte maneira: *Activity → isPerformedBy → Actor.*

A **Figura 26** exemplifica um aspecto importante do diagrama de classes gerado em relação aos atributos de dados das classes. O mapeamento desses atributos (**RM2**) é possível devido à documentação textual dos elementos gráficos no modelo de processos de negócio original. Essa documentação é realizada por meio do atributo estendido identificado como "ATRIBUTOS" (HNd9 e HNd10 - ver *Seção 4.3*, **Tabela 6**), usando uma sintaxe simples, no padrão *"nome_do_atributo: tipo_de_dado;"*. Por exemplo, na expressão, *"name:T; date:D;"* são documentados os atributos *name* do tipo *Text* e *date* do tipo *Date*. As demais abreviações para os tipos de dados padrão são: *B* (*boolean*), *I* (*integer*) *e F* (*float*). Caso seja colocado um nome de entidade (ex.: *"Product"),* será criada uma classe e estabelecido um relacionamento com ela.

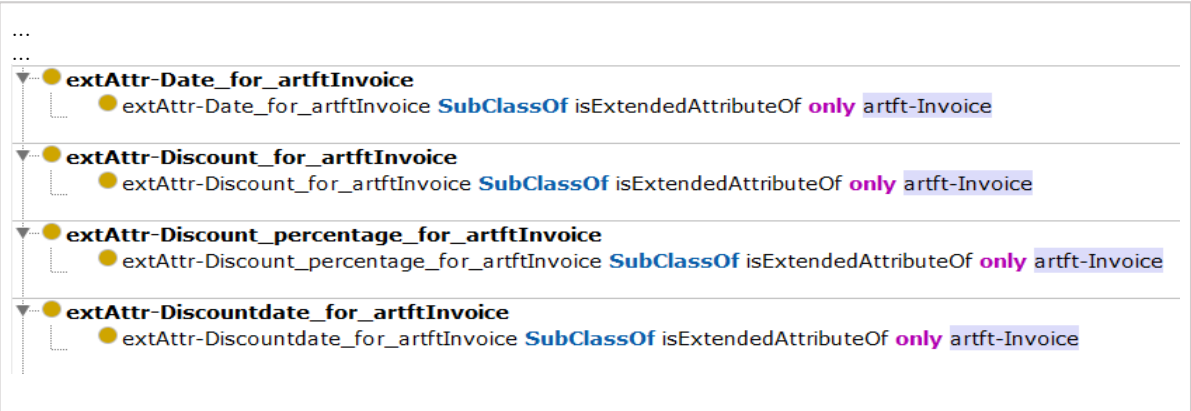
Figura 26 - Exemplo de texto em um atributo estendido, usando o sistema *Bizagi Modeler*.



Fonte: elaborado pelo autor.

Na **Figura 26** é mostrado, como exemplo, um elemento do tipo *DataStore* **(1)** em um modelo em BPMN, construído com a ferramenta *Bizagi Modeler* (*https://www.bizagi.com/en/products/bpm-suite/modeler*). A guia de atributos estendidos **(2)** do sistema modelador, permite a criação de atributos **(3)** segundo as necessidades do domínio. Em destaque **(4)**, está definido um atributo do tipo classe. Em consequência, na **Figura 27** é mostrada parte do conjunto de atributos estendidos documentados já mapeados na ontologia. Essa visualização foi feita usando a ferramenta *Protégé* v5.2.0 (*https://protege.stanford.edu*).

Figura 27 - Visualização de parte dos atributos estendidos na ontologia, usando o sistema *Protégé*.



Fonte: elaborado pelo autor.

Considerando o mapeamento apresentado na **Tabela 11**, o algoritmo para a geração do diagrama de classes é apresentado em linguagem natural na **Figura 28**.

Figura 28 - Algoritmo para geração do diagrama de classes.

```
recuperar todas as instâncias de DataObject e DataStore em DataList
para (cada objeto em DataList) {
  definir uma classe estereotipada em C
  recuperar as instâncias de DataAttribute de C em DA
  para (cada DA) {
    se (DA = tipo de dado primitive)
      adicionar DA em C
    senão, definir uma nova classe NC
  }
} //..atores e seus métodos (funcionalidades)
recuperar todas as instâncias de Actors em ActList
para (cada objeto em ActList){
   definir uma classe A estereotipada como "actor"
   para(cada instância de ActList){
      recuperar todas as instâncias de Activity em AtyList
      para(cada AtyList){
         recuperar Objetos e Depósitos de dados em Data
         para(cada Data){
            definir um método como o nome da instância de AtyList
            definir os parâmetros de entrada (usesInput)
            definir os parâmetros de saída (producesOutput)
         }
      }
   }
} //..relacionamento entre atores e objetos de dados
para (cada SC){
   recuperar todas as instâncias de UsesInput em UI
   para (cada UI)
      definir um relacionamento "Uses Input" entre SC e UI
}
para (cada each SC){
   recuperar todas as instâncias de ProducesOutput em PO
   para (cada PO)
      definir um relacionamento "Produces Output" entre SC e PO
}
```

Fonte: elaborado pelo autor.

De modo geral, o algoritmo apresentado na **Figura 28** pode ser expresso da seguinte maneira:

1) Todas as instâncias das classes *DataStore*, *DataObject* e *Actor* devem ser obtidas da ontologia por meio de uma consulta SPARQL;

2) Para cada instância da classe *DataStore* e da classe *DataObject, d*efinir uma classe UML estereotipada de acordo com sua superclasse;

    a) Recuperar todos os atributos estendidos da classe ontológica, segundo a relação *hasExtendedAttribute;*

b) Para cada atributo estendido:

i) Se for de um dos tipos padrão (*integer*, *boolean*, *float*, *text* or *date*), definir o atributo; senão

ii) Criar uma classe UML.

4) Para cada instância de *Actor*:

b) definir uma classe UML estereotipada como "*User*";

c) recuperar as instâncias da classe ontológica *Activitity* que possuem as relações *usesInput* ou *ProducesOutput*;

d) definir os métodos (funcionalidades) da classe de acordo com as relações *isPerformedBy* e *usesInput* (define parâmetros de entrada) e *producesOutput* (define o tipo da saída);

e) definir um relacionamento entre a instância corrente da classe *Actor* e as instâncias das classes *DataObject* e/ou *DataStore* de acordo com as relações *usesInput* e/ou *ProducesOutput*.

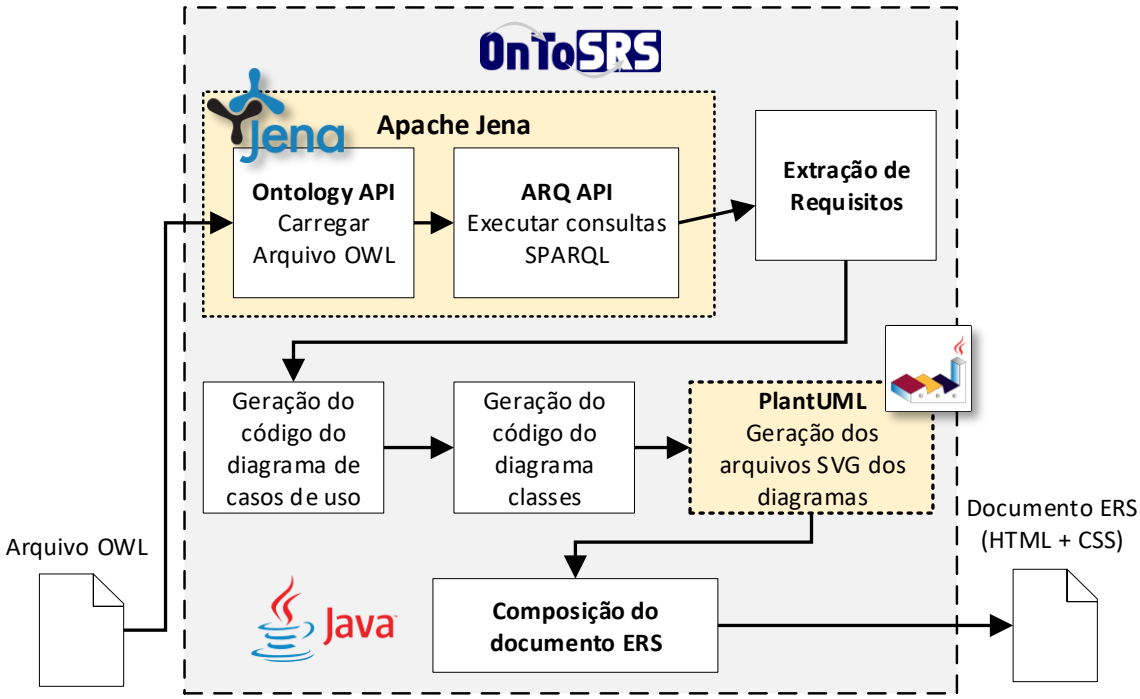
5.5 SISTEMA ONTOSRS

Os processos de geração do documento ERS e dos diagramas UML, apresentados respectivamente na *Seção 5.3* e na *Seção 5.4*, foram implementados no sistema OnToSRS (*Ontology to Software Requirements Specification*), conforme esquema da **Figura 29**. O sistema foi desenvolvido com a plataforma Java SE (linguagem Java 1.8), usando o *framework* Apache Jena (*https://jena.apache.org/*) para realização das consultas SPARQL (ver APÊNDICE B) e a ferramenta PlantUML (*http://www.plantuml.com*) para geração dos diagramas UML (ver APÊNDICE C). O código fonte do sistema OnToSRS está disponível no APÊNDICE D.

Como ilustrado na **Figura 29**, o sistema OnToSRS recebe como entrada um arquivo OWL gerado pelo sistema PM2ONTO v2.0. Depois, por meio das APIs do framework Apache Jena, carrega a estrutura ontológica e executa consultas semânticas para extrair as informações básicas do documento ERS. Feito isso, gera o código de criação dos diagramas de caso de uso e classe, usando as informações recuperadas da ontologia. Então, de acordo com a linguagem PlantUML, os arquivos em formato SVG (*Scalable Vector Graphics*) dos diagramas são gerados e a estrutura do documento ERS é composta,

segundo a **Tabela 11** (*Seção 5.3*). Por fim, a saída é o documento ERS e pode ser composto por um ou mais arquivos em HTML.

Figura 29 - Visão geral do sistema OnToSRS.



Fonte: elaborado pelo autor.

Convém observar que o *framework* Apache Jena é de domínio público (*freeware*), com fonte aberta (*open source*) e fornece meios para construção de ontologias nas linguagens RDF e OWL. O sistema OnToSRS usou as APIs "Ontology" e "ARQ" do referido *framework* para carregar a ontologia e posterior execução de consultas semânticas na linguagem SPARQL, respectivamente. Convém lembrar que, neste trabalho, as consultas em SPARQL são determinantes para a extração das informações da ontologia.

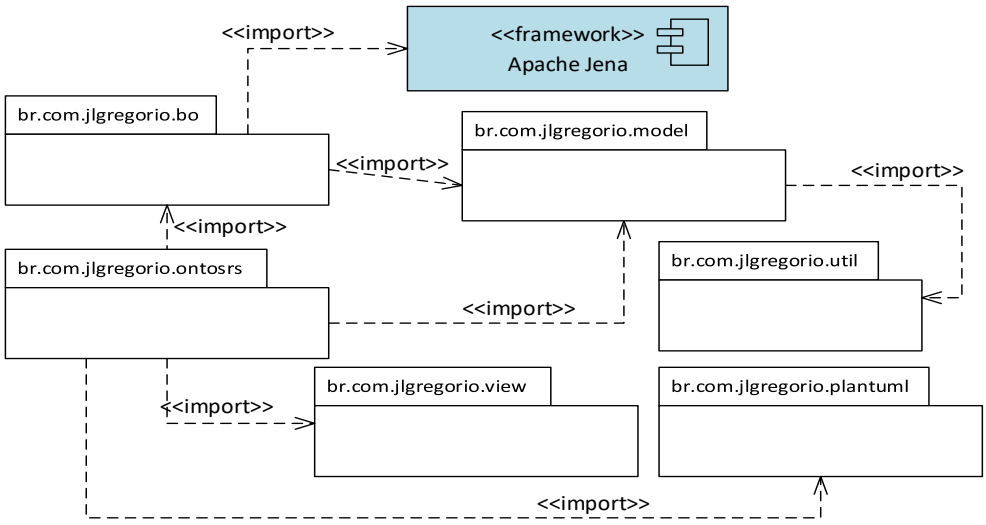
A ferramenta PlantUML, também de domínio público e com código aberto, é usada para a criação de diagramas UML usando uma linguagem descritiva de alto nível (ver APÊNDICE C). Neste trabalho foi usado o *plugin* PlantUML (*https://github.com/qjebbs/vscode-plantuml*) para uso da linguagem textual PlantUML no ambiente de desenvolvimento Microsoft Visual Studio Code (*https://code.visualstudio.com/*), para fins de teste. A natureza textual da linguagem PlantUML facilita a geração de diagramas a partir do processamento de informações oriundas de uma ou mais fontes. A renderização dos diagramas requer a instalação da biblioteca de ferramentas de código fonte aberto GraphViz (*http://gitlab.com/graphviz/graphviz*), que usa a linguagem de descrição gráfica DOT para exibir os diagramas. Além de editores *online*, como o PlantUML Editor (*http://plantuml-*

editor.kkeisuke.com), é possível encontrar *plugins* de integração para diversas ferramentas de edição de código fonte, a fim de testar a geração de diagramas.

A arquitetura do sistema OnToSRS é organizada em seis pacotes, ilustrados no diagrama de pacotes da **Figura 30**:

- ***br.com.jlgregorio.view*** - agrupa as interfaces gráficas do usuário. Como o sistema é bastante simples em termos de interface gráfica, esse pacote possui somente a classe *ViewMain.java*;
- ***br.com.jlgregorio.model*** - agrupa as classes de modelo do sistema, ou seja, as classes que modelam o domínio, que, neste caso, precisa compor um documento ERS;
- ***br.com.jlgregorio.util*** - agrupa as classes utilitárias do sistema, como os enumeradores, por exemplo, que determinam tipos de requisitos funcionais;
- ***br.com.jlgregorio.bo*** - agrupa as classes de negócio (BO: *business object*), que, neste contexto, encapsula a lógica de recuperação dos dados da ontologia por meio de consultas em SPARQL. As classes desse pacote são dependentes do *framework* Apache Jena;
- ***br.com.jlgregorio.plantuml*** - agrupa classes para a geração dos diagramas UML segundo a linguagem PlantUML;
- ***br.com.jlgregorio.ontosrs*** - possui a classe principal do sistema, responsável por iniciar a execução de OnToSRS.

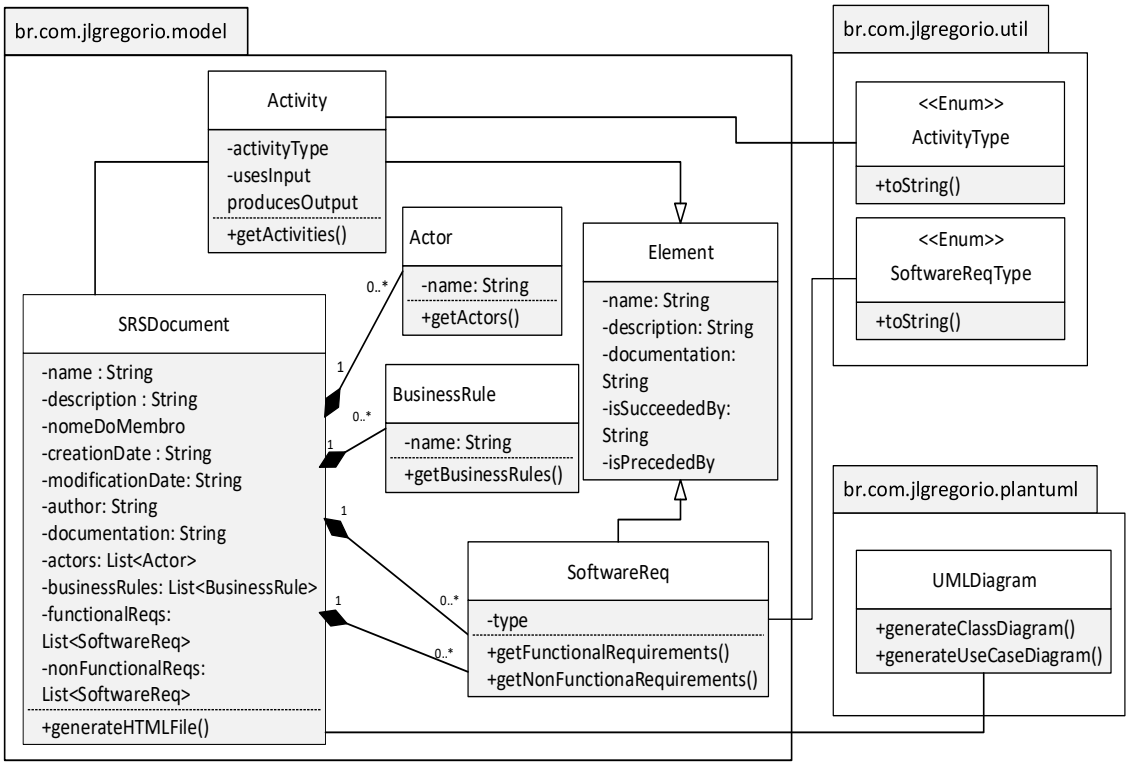
Figura 30 - Diagrama de pacotes do sistema OnToSRS.



Fonte: elaborado pelo autor.

Os diagramas de classes dos pacotes *model* (modelagem de domínio), útil (utilitários) e plantuml (criação dos diagramas) são mostrados parcialmente na **Figura 31**.

Figura 31 - Classes e relacionamentos (parcial) dos pacotes "model", "útil" e "plantuml".

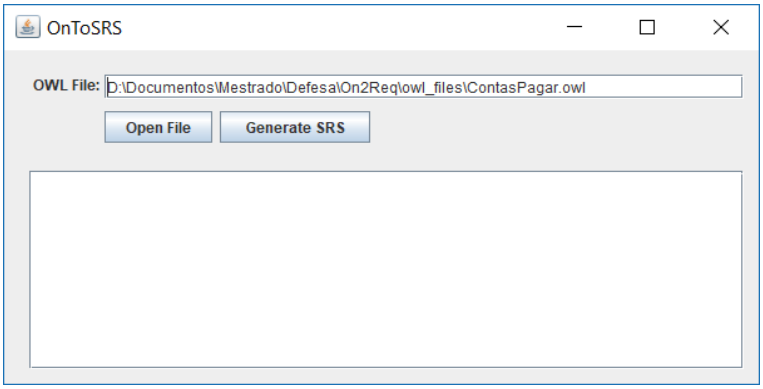


Fonte: elaborado pelo autor.

As classes do pacote "*model" SRSDocument*, *SoftwareReq*, *BusinessRule*, *Activity* e *Actor* possuem classes homônimas, pertencentes ao pacote "*bo",* responsável por encapsular as consultas em SPARQL, que recuperam dados da ontologia. A classe *UMLDiagram* possui a implementação dos algoritmos para a geração dos diagramas de classes e casos de uso. Assim, os objetos que compõem o documento ERS representam os elementos e os mapeamentos semânticos mostrados na **Tabela 9** (*Seção 5.3*).

A interface do usuário no sistema OnToSRS é composta por apenas dois botões: "open", para carregar o arquivo OWL, e "Generate SRS", para gerar o documento ERS (**Figura 32**). Após carregar o arquivo e clicar no botão "Generate SRS", será solicitado o local onde o(s) arquivo(s) será(ão) criado(s). Feito isso, os arquivos serão abertos no navegador padrão do sistema operacional.

Figura 32 - Interface gráfica do usuário do sistema OnToSRS.



Fonte: elaborado pelo autor.

5.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo foi apresentado o processo sistematizado para a geração do documento ERS, em conformidade com o padrão ISO/IEC/IEEE 29148:2018, a partir de ontologias de processos geradas pelo sistema PM2ONTO v2.0. Esse documento contempla atores, pré-condições, pós-condições, regras de negócio, requisitos funcionais e não funcionais, além de diagramas de casos de uso e classes. A extração das informações da ontologia é possível por meio da linguagem de consultas semânticas SPARQL e, principalmente, por meio de regras definidas para o mapeamento semântico entre os elementos da ontologia e os do documento ERS. A geração dos diagramas de classes e casos de uso requerem processamento adicional, pois não há uma correspondência semântica direta (“1 para 1”) entre os elementos da ontologia e dos diagramas. Convém destacar que muitas das regras de mapeamento semântico são definidas considerando a documentação textual do modelo de processos de negócio em BPMN e não somente a estrutura ontológica implícita. Essencialmente, a expressividade dos diagramas UML gerados está limitada ao conhecimento representado na ontologia.

O processo sistematizado apresentado foi implementado no sistema OnToSRS, desenvolvido na plataforma Java SE. O *framework* Apache Jena foi usado para a realização das consultas em SPARQL, enquanto que a ferramenta PlantUML foi usada para definição dos diagramas UML.

O documento ERS gerado atende os objetivos apresentados na *Seção 1.1*, com um mapeamento PIM para PIM bem-sucedido, considerando-se a arquitetura MDA. No próximo capítulo, serão apresentados testes e estudos de caso para validar todo o processo definido e implementado no sistema OnToSRS.

6 AVALIAÇÃO DO PROCESSO SISTEMATIZADO PARA ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE

Neste capítulo são apresentados estudos de caso para avaliação do processo sistematizado apresentado no *Capítulo 5*, que possibilita a geração automática de um documento de especificação de requisitos de software (ERS) a partir de uma ontologia de processos, que representa um modelo de processos de negócio em BPMN v2.0. O objetivo é avaliar se as informações no documento ERS são semanticamente equivalentes ao conteúdo representado na ontologia, gerada pelo sistema PM2ONTO v2.0. Convém lembrar que os modelos de processos de negócio devem satisfazer critérios de boas práticas de modelagem, com documentação de atributos estendidos, como apresentado nas *Seções 2.3* e *4.3*. Como mencionado na *Seção 5.2*, o sistema PM2ONTO precisa de ajustes em relação ao mapeamento de relações entre classes *Activity* e *Artifact*, requerendo que as ontologias sejam ajustadas manualmente para geração adequada dos diagramas de casos de uso e classes – o que foi feito para os testes e estudos de caso aqui apresentados.

Com o objetivo de verificar se o mapeamento "ontologia → documento ERS" é satisfatório, os testes realizados consideraram, inicialmente, a parte principal do documento ERS, que possui apenas as informações básicas do processo representado. Essa parte representa um mapeamento "1 para 1", ou seja, um elemento da ontologia pertinente ao escopo do documento ERS possui um elemento equivalente no documento ERS. Então, a geração dos diagramas UML contemplados neste trabalho (casos de uso e classe) foi testada e ajustada de forma incremental, corrigindo erros e inconsistências encontradas a cada teste.

Convém lembrar que a especificação da notação BPMN é muito extensa, sendo composta por mais de 100 elementos de modelagem. Logo, a estrutura ontológica gerada pelo sistema PM2ONTO não contempla todos os aspectos da especificação. Assim, na **Tabela 12**, são apresentados os elementos de BPMN e seus respectivos correspondentes ontológicos testados.

Tabela 12 - Elementos de BPMN e respectivos elementos ontológicos considerados.

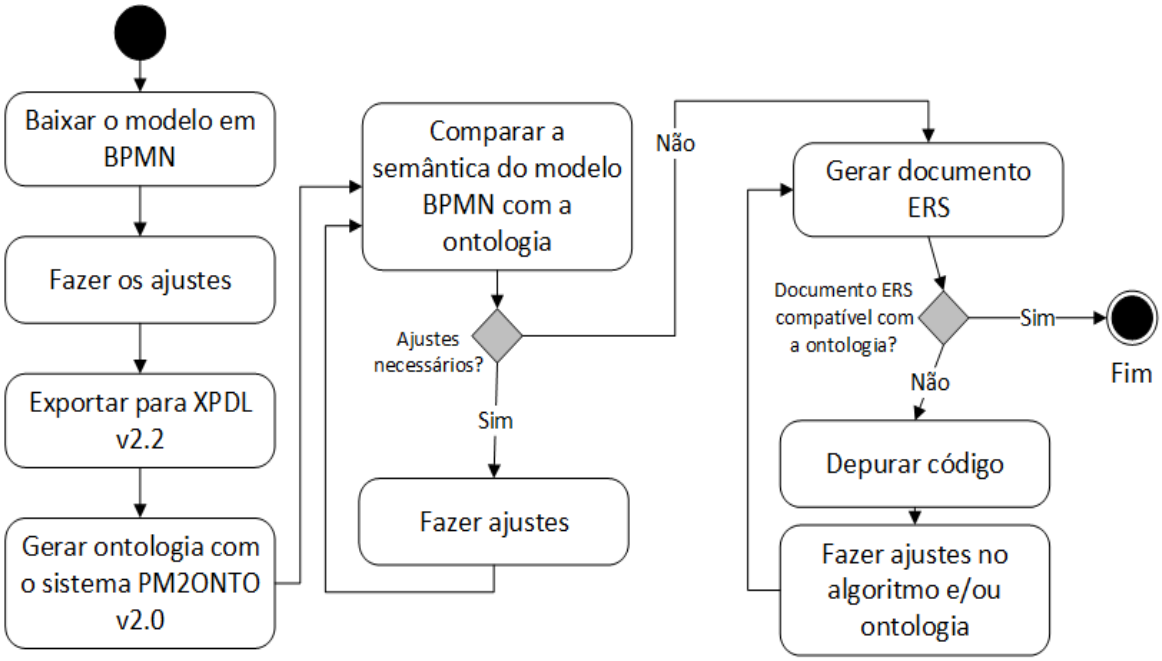
Elementos de BPMN	Elementos da ontologia	Elementos do documento ERS e/ou diagrama UML
Evento (Início, Intermediário e Fim)	*Event* (subclasses *Start*, *Intermediate* e *End*)	Precondições, caso de uso e pós-condições
Atividade (Usuário, *Script*, Recebimento, Envio e Regras de Negócio)	*Activity* (subclasses *Simple*, *User*, *Script*, *Receive*, *Send*, *Manual* e *BusinessRule*)	Casos de uso, métodos e requisites funcionais
Gateway (Exclusivo, Paralelo, Inclusivo)	*Gateway* (subclasses *Exclusive*, *Paralel* e *Inclusive*)	Relacionamentos entre casos de uso do tipo *extend* e *include*
Subprocesso simples	*Subprocess* (subclasse *Simple*)	Caso de uso (detalhado em outro documento ERS
Artefato (Objeto de dados e repositório de dados)	*Artifact* (subclasses *DataObject* e *DataStore*)	Classes
Participante do processo	Actor (subclasses)	Atores e classes
Atributo estendido	*ExtentedeAttribute* (REGRA, RNF e ATRIBUTOS)	Regra de negócio, requisito não funcional e atributos de dados
Relações entre elementos BPMN	**Relações da ontologia**	**Relações dos diagramas UML**
Fluxo de sequência	*isSucceededBy* e *isPrecededBy*	Relação entre casos de uso *include* ou *extend*)
Associação entre artefatos e atividades	*usesInput* e *producesOutput*	Relações entre classes
Elementos que fazem parte de um subprocesso	*isPartOfSubProcess*	Elementos que fazem parte de outro documento SRS
Associação entre participantes e atividades	*isPerformedBy* e *isExecutorOfActivity*	Relação entre Atores e casos de uso

Fonte: elaborado pelo autor.

Neste capítulo, são apresentados quatro estudos de caso realizados para avaliação dos documentos ERS gerados. O fluxo das atividades executadas nos três primeiros estudos de caso está ilustrado na **Figura 33**. No quarto estudo de caso, o fluxo de atividades é diferente, visto que é gerado um documento ERS baseado na integração entre duas ontologias de processos.

Assim, a *Seção 6.1* apresenta o primeiro estudo de caso, considerando o modelo de processos de negócio identificado como “Access Management” (Gerenciamento de acessos). Esse modelo trata de permissões de acesso em uma organização e foi obtido a partir do repositório público “Bizagi Xchange” (*https://www.bizagi.com/pt/comunidade/global-xchange*), da empresa desenvolvedora da ferramenta modeladora *Bizagi Modeler*. Trata-se de um modelo simples, com apenas quatro participantes e nenhum subprocesso.

Figura 33 - Atividades executadas três primeiros estudos de caso.



Fonte: elaborado pelo autor.

A *Seção 6.2* apresenta o segundo estudo de caso, considerando o modelo identificado como “Accounts Payable” (Contas a Pagar). Esse modelo também foi obtido a partir do repositório Bizagi Xchange, mas possui um subprocesso.

O terceiro estudo de caso está apresentado na *Seção 6.3*, considerando o modelo identificado como “Estágio Fatec Jales”, que modela o processo de formalização de estágio dos alunos da Faculdade de Tecnologia “Prof. José Camargo” (Fatec Jales). Esse modelo possui dois subprocessos e outros elementos que os modelos anteriores não possuem, como eventos intermediários e mais de uma piscina.

O estudo de caso da *Seção 6.4* visou avaliar como duas ontologias de processos podem ser integradas. Essa motivação se deu por considerar que informações sobre dois ou mais modelos de processos de negócio podem ser representadas ontologicamente, contribuindo para consultas, tomadas de decisão, análises dos processos, especificações, etc. A *Seção 3.6* apresentou fundamentos sobre a integração de ontologias, que foram consideradas no estudo de caso.

Face ao exposto, a *Seção 6.5* apresenta algumas considerações finais sobre este capítulo.

6.1 ESTUDO DE CASO 1: MODELO "ACCESS MANAGEMENT"

O modelo de processos de negócio "Access Management" foi disponibilizado pelo repositório público *Bizagi Xchange* e, segundo sua documentação, representa o processo de gerenciamento de acessos e permissões em uma organização alinhada aos padrões ITIL (*Information Technology Infrastructure Library*). O modelo original possui: 1 piscina, 4 arraias, 1 evento de início, 1 evento de fim, 4 participantes, 9 atividades (5 sem tipo definido, 1 do tipo Serviço e 3 do tipo Script), 2 *gateways* inclusivos e 2 *gateways* exclusivos. O modelo não possui nenhum subprocesso.

Conformidade com os critérios definidos para a geração da ontologia

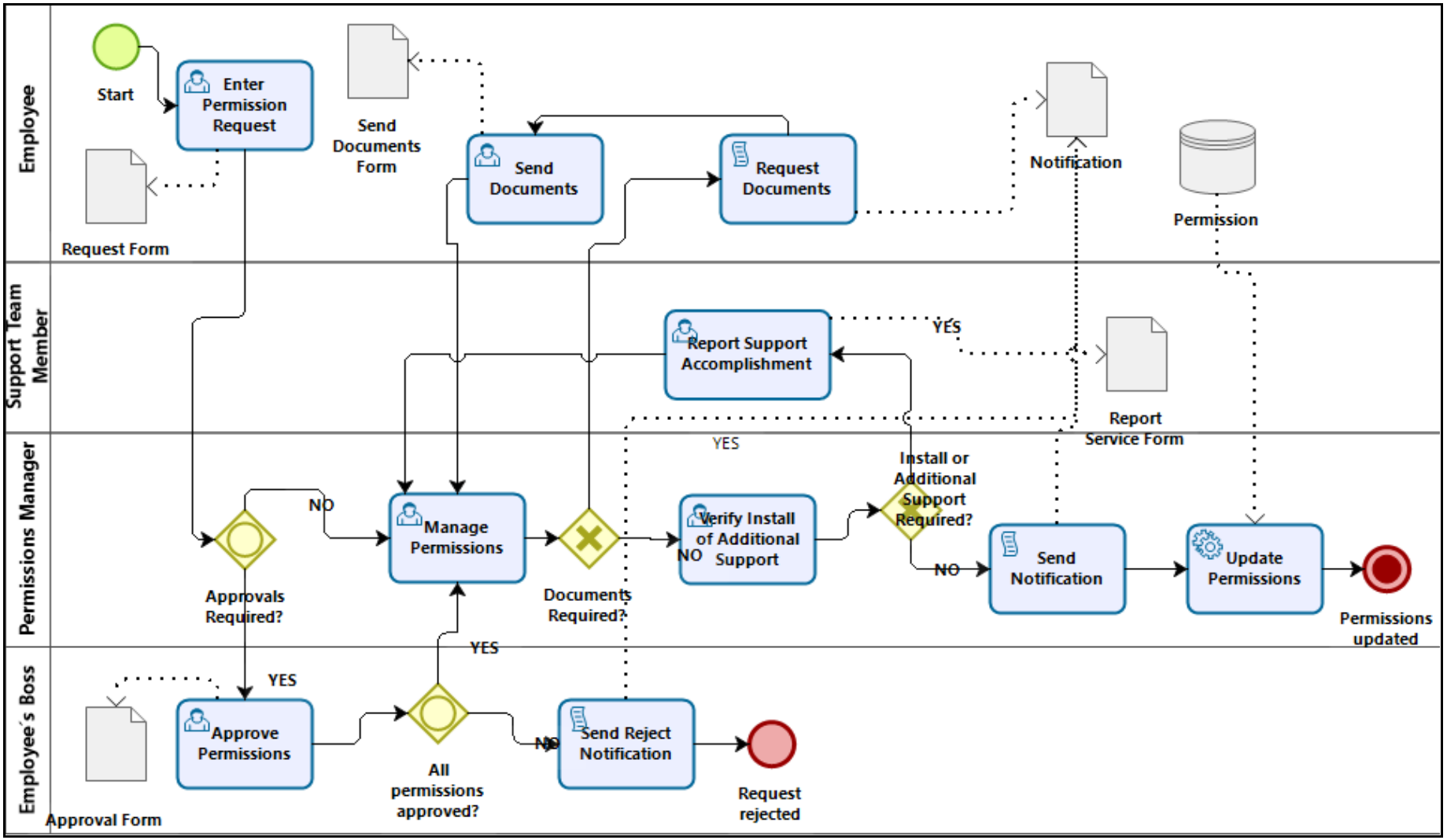
Inicialmente, foi avaliado se o modelo estava em conformidade com os critérios definidos por Figueiredo (2018), apresentados na *Subseção 2.3.3.* Foi necessário fazer ajustes no modelo de modo que ele satisfizesse os critérios C3, C5, C7 e C10. Assim:

- todas as atividades foram documentadas (C3);
- todas as atividades tiveram seus respectivos tipos definidos (C5);
- todas as atividades foram associadas a um ator (participante) (C7);
- os fluxos dos *gateways* foram devidamente nomeados (C10).

Adicionalmente, dois *gateways* encadeados foram ajustados, adicionando-se uma nova tarefa ("*Verify Install of Aditional Support*") com o objetivo de satisfazer a condição COND-1 (ver *Subseção 5.1*). Também foram adicionados novos elementos do tipo Artefato (*DataStore* e *DataObject*), que foram documentados com o atributo estendido "ATRIBUTOS" (HNd9 e HNd10 - ver *Seção 4.3*, **Tabela 6**), a fim de especificar os seus atributos de dados.

Objetivando documentar as regras de negócio, foram usadas as heurísticas de requisitos HR4 e HR6 (ver *Seção 4.3*, **Tabela 7**). Assim, foi adicionado o atributo "REGRA" (HR6) nas atividades "Enter Permission Request", "Request Documents", "Send Notification", "Update Permissions", "Approve Permissions" e "Send Request Notification". O valor do atributo "REGRA" para cada atividade foi determinado segundo a documentação do processo. Os requisitos não funcionais, documentados com o atributo estendido "RNF" (HR4), foram adicionados às atividades "Enter Permission Request" e "Update Permissions". É importante destacar que, mesmo após todos os ajustes, a semântica do modelo resultante, apresentado na **Figura 34**, preserva sua originalidade. Após os ajustes, o modelo passou a ser composto por:

Figura 34 - Modelo de processos de negócio “Access Management”.



Fonte: modelo ajustado após ser extraído do repositório público: *https://www.bizagi.com/en/community/process-xchange*.

- 10 atividades: 6 do tipo “Usuário”, 3 do tipo “Script” e 1 do tipo “Serviço”;
- 6 artefatos: 5 do tipo “Objeto de dados” e 1 do tipo “Repositório de dados”;
- A quantidade dos demais elementos permanece inalterada.

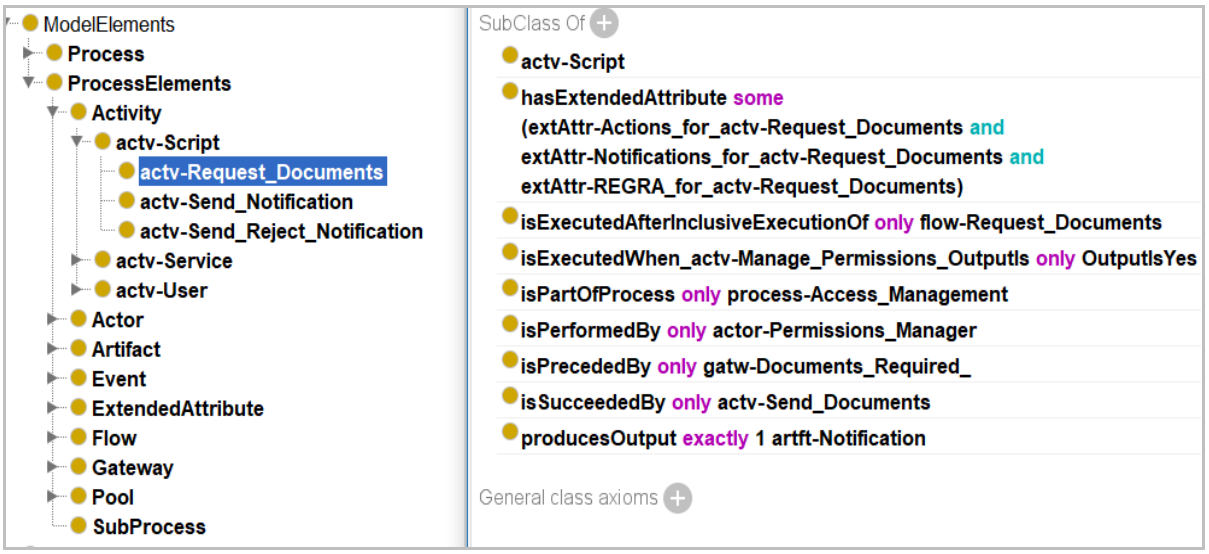
Geração e análise da ontologia

Após exportar o modelo para o formato XPLD v2.2, o arquivo gerado foi usado como entrada no sistema PM2ONTO v2.0, que gerou a ontologia representativa do modelo.

A análise da ontologia considerou suas classes, relações, axiomas e atributos estendidos, usando o editor de ontologias *Protégé* v5.2.0. A ontologia foi considerada adequada para o processo de geração do documento ERS, visto que todos os elementos correspondentes foram encontrados (mapeamento “1 para 1”).

Parte da ontologia gerada é apresentada na **Figura 35**, com as classes à esquerda e as relações da classe “actv-Request_Documents” à direita, a qual corresponde à atividade “Request Documents” no modelo de processos de negócio.

Figura 35 - Parte da ontologia representativa do modelo “Access Management”.



Fonte: elaborado pelo autor.

Geração e análise do documento ERS

O arquivo OWL gerado foi usado como entrada no sistema OnToSRS, gerando o documento ERS, em conformidade com o padrão ISO/IEC/IEEE 29148:2018, contemplando atores, precondições, pós-condições, regras de negócio, requisitos funcionais, requisitos não funcionais e diagramas UML (casos de uso e classes).

O mapeamento entre os elementos da ontologia e os elementos do documento ERS foi analisado e se mostrou de acordo com o mapeamento definido na **Tabela 9** (ver *Seção 5.3*). O mapeamento resultou em 4 atores, 1 precondição (evento de início), 2 pós-condições (eventos de fim) e 4 regras de negócio (atributo estendido "REGRA") no documento ERS correspondente, como pode ser visto na **Figura 36**.

Figura 36 – Parte do documento ERS gerado para o modelo "Access Management".

About the Software Requirements	
Actor(s):	**New Boss for employee**: Person who approves the special permissions of a specific employee **Support Team Member**: Person in charge of additional managements for the activation, deactivation or change of privileges of permissions. **Employee**: Person who opens the case **Permissions Manager**: Person who manage the activation, deactivation or change of privileges of a permission
Pre-conditions:	Enter request
Pos-conditions:	**1**:Request rejected **2**:Permissions updated
Business Rules:	**1**:Get parameter values **2**:This activity must be performed by the Boss of the user for whom the permissions are requested The activity is assigned to the person registered as Boss of the user for whom the permissions are requested. Update permissions status **3**:Create E-mail **4**:Add permissions

Fonte: elaborado pelo autor.

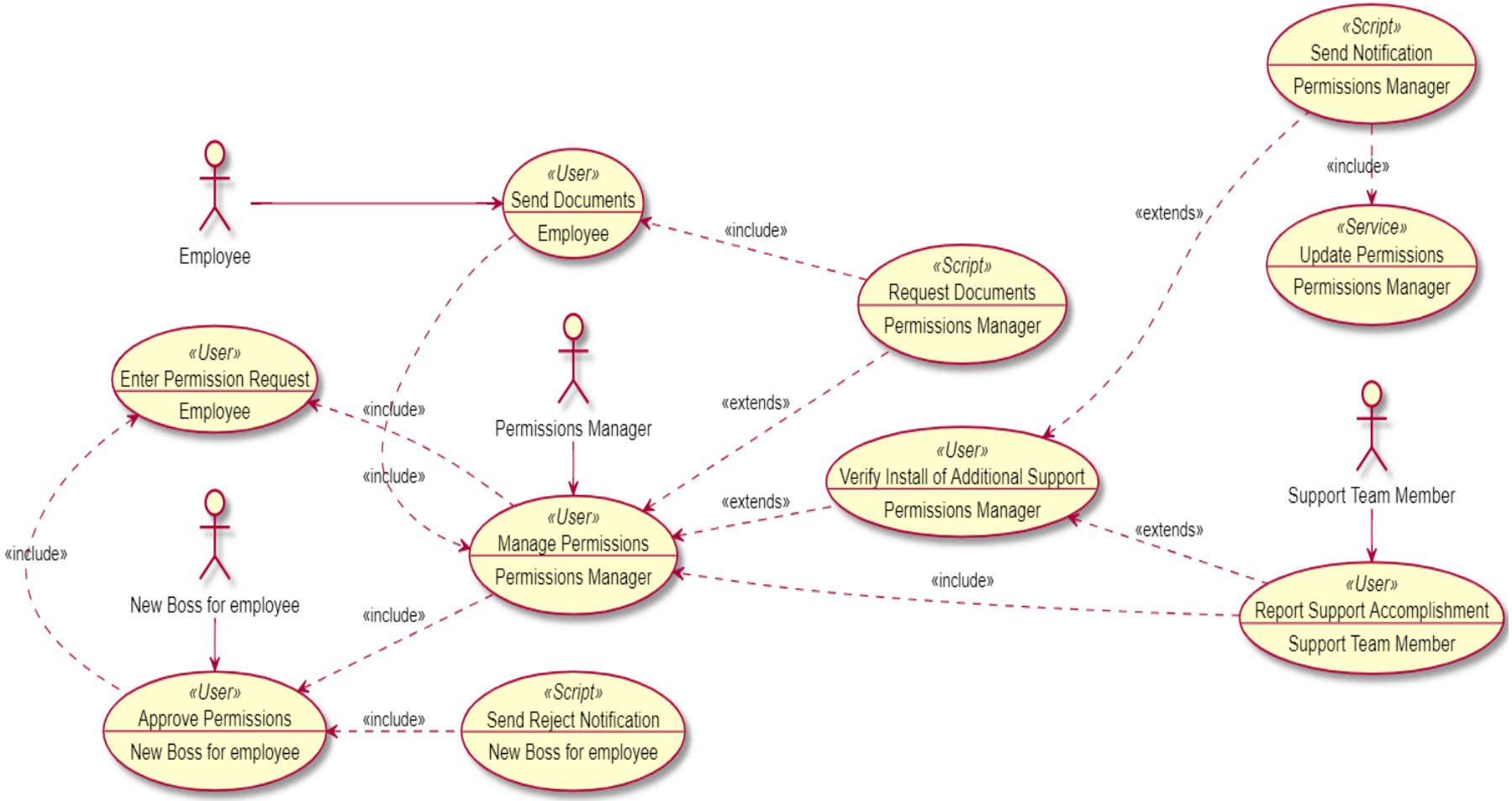
O mapeamento dos elementos da ontologia para os elementos dos diagramas pode ser visto na **Tabela 13**. Os diagramas de casos de uso e de classes gerados são mostrados nas **Figuras 37** e **38**, respectivamente.

Discussão dos resultados

O diagrama de casos de uso (**Figura 37**) mostrou uma representação adequada dos elementos do modelo de processos de negócio do ponto de vista semântico. Mesmo não sendo possível expressar com exatidão o fluxo dos processos, foram mantidos os relacionamentos entre atores e casos de uso, bem como entre os casos de uso em si (ideia de fluxo).

No modelo de processos de negócio, o participante "New Boss for employee" é o executor da atividade "Approve Permissions". Essa atividade é precedida e sucedida por *gateways* inclusivos, "Approval Required?" e "All Permissions Approved", respectivamente. O *gateway* "All Permissions Approved" é sucedido pelas atividades "Manage Permissions" e "Send Reject Notification". Cada uma dessas atividades é executada por diferentes participantes.

Figura 37 - Diagrama de casos de uso para o modelo “Access Management”.



Fonte: elaborado pelo autor.

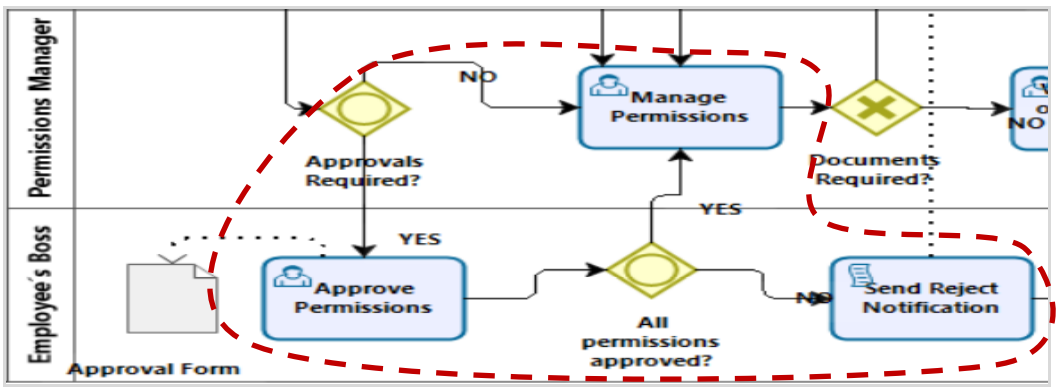
Tabela 13 – Mapeamento da ontologia para os diagramas UML - modelo “Access Management”.

Elemento da ontologia	Diagrama de casos de uso	Diagrama de classes
10 subclasses de Activity	10 casos de uso	10 métodos
4 subclasses de Actor	4 atores	4 classes com estereótipo “actor”
2 sub-casses de “gatw-Exclusive”	4 relacionamentos do tipo “extend” entre casos de uso	
2 subclasses de “gatw-Inclusive”	4 relacionamentos do tipo “include” entre casos de uso	
5 subclasses de “DataObject”		5 classes com estereótipo “data_object”
1 subclasse de “DataStore”		1 classe com estereótipo “data_store”
5 relações “producesOutput”		5 tipos de saídas de métodos
1 relação “usesInput”		1 parâmetro de entrada de método
43 atributos estendido “ATRIBUTOS”		43 atributos de dados

Fonte: elaborado pelo autor.

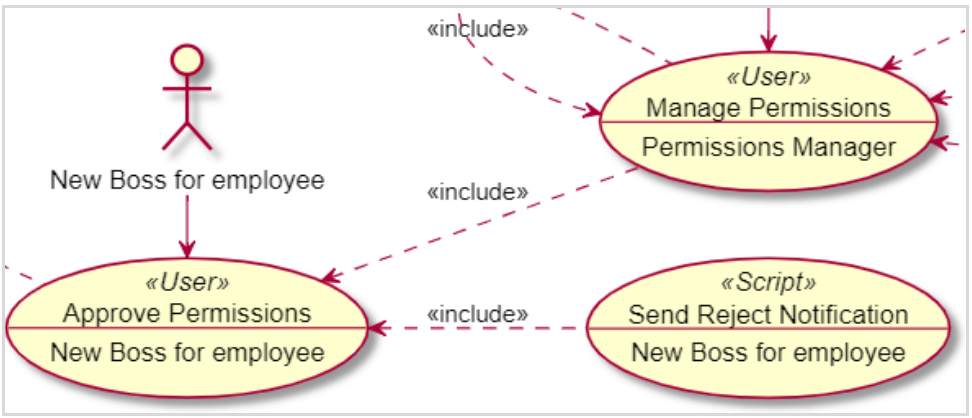
Logo, como esperado, essas atividades foram mapeadas para casos de uso no diagrama, vinculadas ao respectivo ator, com relacionamentos do tipo “*include*” entre eles. Para exemplificar, a **Figura 38** mostra um fragmento do modelo de processos de negócio, para o qual corresponde parte do diagrama de casos de uso com relacionamentos do tipo “include”, como apresentado na **Figura 39**. De modo similar, a **Figura 40** mostra um fragmento do modelo com “*gateway* exclusivo”, para o qual corresponde parte do diagrama de casos de uso com relacionamentos do tipo “extend”, apresentada na **Figura 41**.

Figura 38 - Fragmento do modelo “Access Management” com dois *gateways* inclusivos e três atividades.



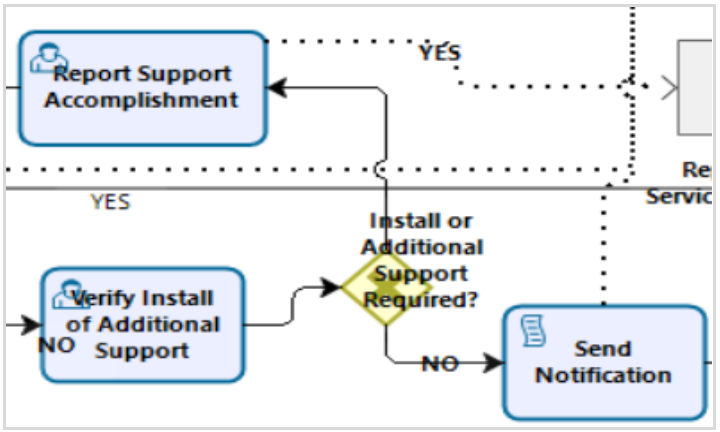
Fonte: elaborado pelo autor.

Figura 39 - Fragmento do diagrama de casos de uso correspondente ao fragmento na Figura 38.



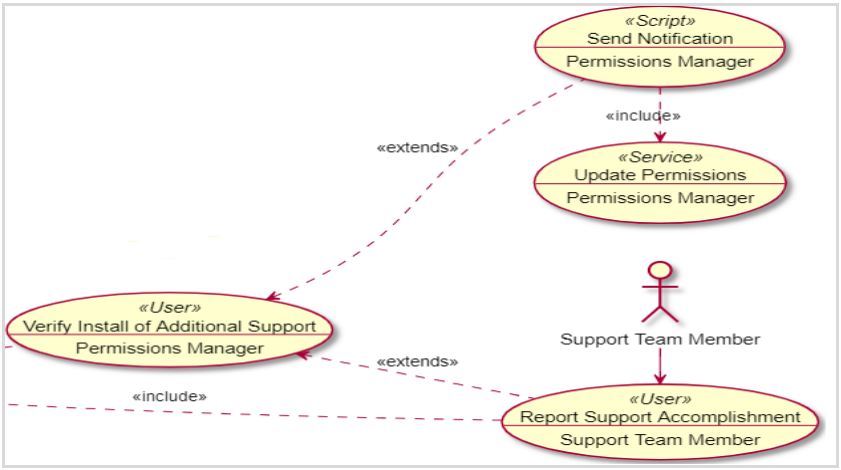
Fonte: elaborado pelo autor.

Figura 40 - Fragmento do modelo “Access Management” com um *gateway* exclusivo e três atividades.



Fonte: elaborado pelo autor.

Figura 41 - Fragmento do diagrama de casos de uso correspondente ao fragmento na Figura 40.



Fonte: elaborado pelo autor.

Na **Figura 37**, pode ser observado que os casos de uso possuem duas seções. A seção superior contém o nome do caso de uso e a seção inferior informa o respectivo ator. Essa estratégia foi adotada para deixar o diagrama mais legível, visto que expressar todas as relações diretas entre atores e casos de uso geraria muitas setas, prejudicando a legibilidade do diagrama.

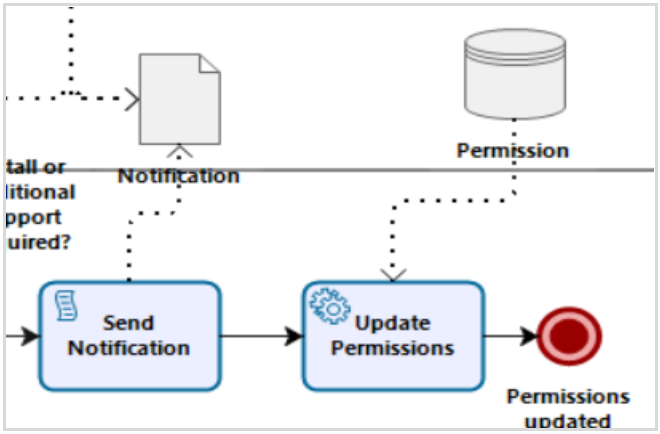
Em relação ao diagrama de classes gerado, também se mostrou adequado semanticamente à ontologia, visto que mapeou todos os elementos ontológicos esperados, ou seja, classes, atributos, métodos e relacionamentos.

De acordo com o que foi apresentado na *Subseção 5.4.2*, os atributos das classes incluíram os tipos de dados que foram mapeados do atributo estendido "ATRIBUTOS" no modelo de processos de negócio e depois convertidos em instâncias de classes ontológicas *ext-Attr*. As classes com estereótipo "actor" foram mapeadas das subclasses ontológicas *Actor*. As classes de dados foram mapeadas das subclasses ontológicas de *Artifact*, *DataStore* e *DataObject*. Os relacionamentos entre as classes foram mapeados a partir das relações *usesInput* e *producesOutput* e do atributo estendido "ATRIBUTOS", que permite a definição de relacionamentos. As **Figuras 42** e **43** evidenciam esses aspectos, além de mostrarem a correspondência semântica entre o modelo de processos de negócio e o diagrama de classes.

Na **Figura 42**, é mostrado um fragmento do modelo considerado nesse estudo de caso, destacando-se as atividades "Send Notification" e "Update Permissions". Ambas são executadas pelo participante "Permissions Manager". Quando a classe "Permissions Manager" foi mapeada para o diagrama de classes, uma classe homônima foi criada e os métodos foram definidos de acordo com as atividades e os parâmetros de entrada e saída definidos pelos relacionamentos ontológicos *usesInput* e *producesOutput*, respectivamente. A parte correspondente do diagrama de classes (**Figura 43**) manteve consistência com a parte do modelo apresentada.

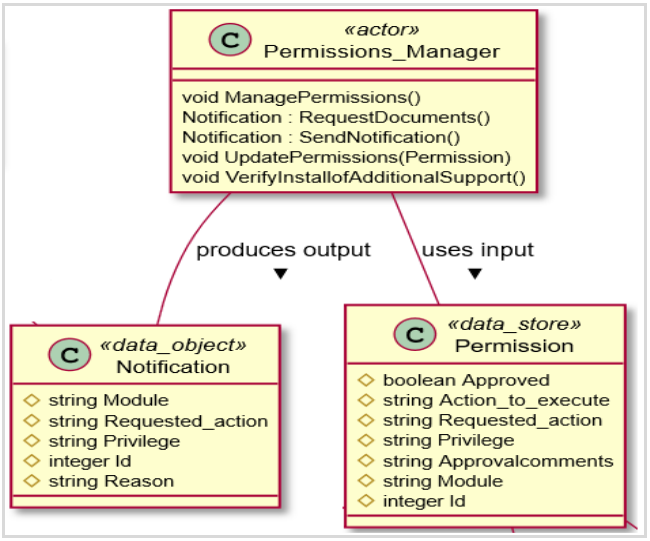
Convém destacar que, entre os atributos de dados, em todos os elementos *DataStore* dos estudos apresentados, há um atributo "id", referente ao "identificador" da classe. Do ponto de vista de classes em UML, esse não seria um atributo usável na modelagem, mas foi necessário inseri-lo, para possibilitar, em trabalhos futuros, a geração automática de código e integração com sistemas de gerenciamento de banco de dados. Além disso, todas as classes possuem estereótipos, objetivando enriquecer semanticamente o diagrama e relacioná-lo à ontologia de origem.

Figura 42 - Fragmento do modelo “Access Management” com duas atividades e dois objetos de dados.



Fonte: elaborado pelo autor.

Figura 43 - Fragmento do diagrama de classes correspondente ao fragmento na Figura 42.



Fonte: elaborado pelo autor.

O APÊNDICE E traz o modelo “Access Management” em BPMN v2.0 na versão original e com os ajustes adequados à aplicação do processo sistematizado, o arquivo em OWL da ontologia representativa do modelo, bem como o documento ERS completo.

6.2 ESTUDO DE CASO 2: MODELO “ACCOUNTS PAYABLE”

O modelo “Accounts Payable” (Contas a Pagar) foi disponibilizado gratuitamente no repositório público *Bizagi Xchange* e, segundo sua documentação, representa o processo de recepção, validação e aprovação de uma fatura. Ele foi selecionado como o segundo estudo

de caso por ter um subprocesso, gerando dois documentos ERS, um para o processo principal e outro para o subprocesso em questão, como apresentado na *Seção 5.3*. Observa-se que o documento final ERS é composto pela união desses dois documentos gerados.

O processo principal do modelo possui: 1 evento de início, 2 eventos de fim, 5 atividades (4 sem tipo definido, 1 do tipo Serviço), 3 *gateways* exclusivos e 4 raias. O subprocesso do modelo possui: 1 evento de início, 2 eventos de fim, 3 atividades (1 de tipo indefinido, 1 do tipo *Script* e 1 do tipo *Manual*) e 1 *gateway* exclusivo.

Conformidade com os critérios definidos para a geração da ontologia

Inicialmente, foi avaliado se o modelo estava em conformidade com os critérios requeridos, como apresentado na *Subseção 2.3.3*. Foi necessário fazer ajustes no modelo de modo que ele satisfizesse os critérios C3, C5 e C7. Assim:

- todas as atividades foram devidamente nomeadas e documentadas (C3);
- todas as atividades tiveram seus respectivos tipos definidos, seguindo a documentação original do processo (C5);
- todas as atividades foram associadas aos seus respectivos participantes (C7).

Assim como no estudo mostrado na *Seção 6.1*, também foram adicionados, ao modelo, os atributos estendidos “RNF” (HR4) e “REGRA” (HR6) segundo as heurísticas de requisitos (ver *Seção 4.3*, **Tabela 7**) definidas por Nogueira (2017). O atributo “REGRA” foi adicionado nas atividades “Validate Invoice”, “Justify the Rejection” e “Inform Supplier”. O atributo “RNF” foi adicionado nas atividades “Validate Invoice”, “Update Financial ERP. Objetos e repositórios de dados foram adicionados seguindo a documentação original/oficial, objetivando enriquecer o modelo, possibilitando a extração do diagrama de classes. Esses elementos de dados também foram documentados com o atributo estendido “ATRIBUTOS” (HNd9 e HNd10 - ver *Seção 4.3*, **Tabela 6**), objetivando especificar seus respectivos atributos de dados. Todos esses ajustes preservaram a semântica original do modelo. Após os ajustes, o processo principal (**Figura 44.a**) do modelo passou a ser composto por:

- 2 eventos: 1 do tipo “Início, 1 do tipo “Fim”;
- 5 atividades: 4 do tipo Usuário, 1 do tipo Serviço;
- 3 *gateways* do tipo “Exclusivo”;
- 4 artefatos: 2 do tipo “Objeto de dados”, 2 do tipo “Repositório de dados”;

- 4 Participantes;
- 4 raias.

O subprocesso “Return Invoice to Supplier” (**Figura 44.b**), após os ajustes, passou a ser composto por:

- 3 eventos:1 do tipo “Início”, 2 do tipo “Fim”;
- 3 atividades: 1 do tipo “Usuário”, 1 do tipo “Script”, 1 do tipo “Manual”;
- 2 artefatos do tipo “Objetos de dados”;
- 1 gateway do tipo “Exclusivo”.

Geração e análise da ontologia

Após exportar o modelo para o formato XPLD v2.2, o arquivo gerado foi usado como entrada no sistema PM2ONTO v2.0, que gerou a ontologia representativa do modelo.

Todos atores estão relacionados ao subprocesso “Return Invoice to Supplier” por meio da relação *isPartOfSubProcess*. Entretanto, no modelo de processos de negócio considerado (**Figura 44**), apenas o ator “Financial Assistant” faz parte desse subprocesso. Assim, na ontologia, essa relação foi excluída nas classes *actor-Accountant*, *actor-Administration_Manager* e *actor-Receptionist*. Com essas alterações, foi verificado que as demais relações entre as classes ontológicas expressavam coerência semântica com o modelo considerado. Então, o arquivo OWL, da ontologia, foi usado como entrada no sistema OnToSRS.

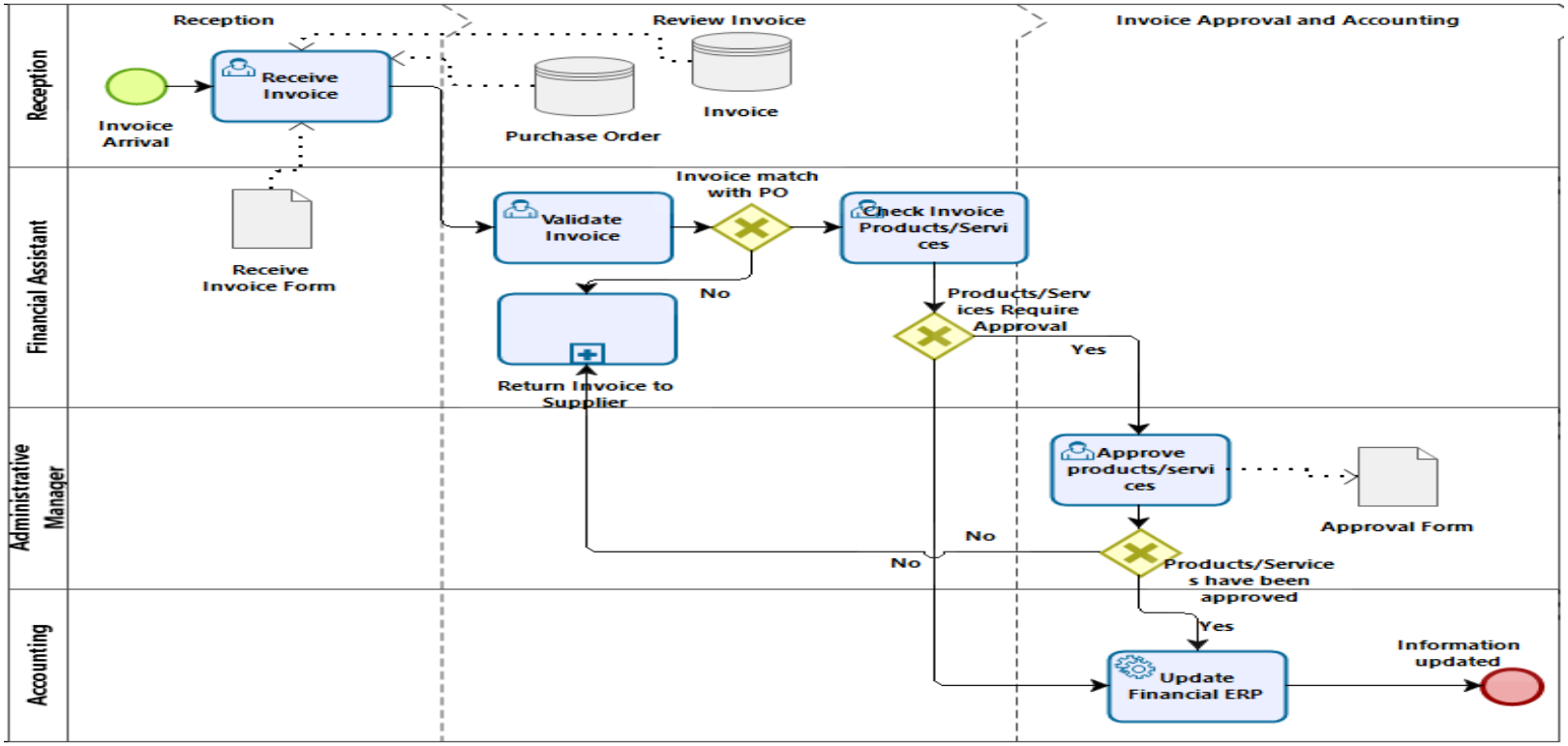
Geração e análise do documento ERS

Nesse estudo de caso, foram gerados dois documentos ERS, um para o processo principal e um para o subprocesso, considerando que o documento ERS final é a composição desses dois documentos. Foi possível perceber que a estratégia de gerar um documento ERS para o processo principal e um para cada subprocesso presente na ontologia possibilitou documentos simples, de fácil leitura e compreensão.

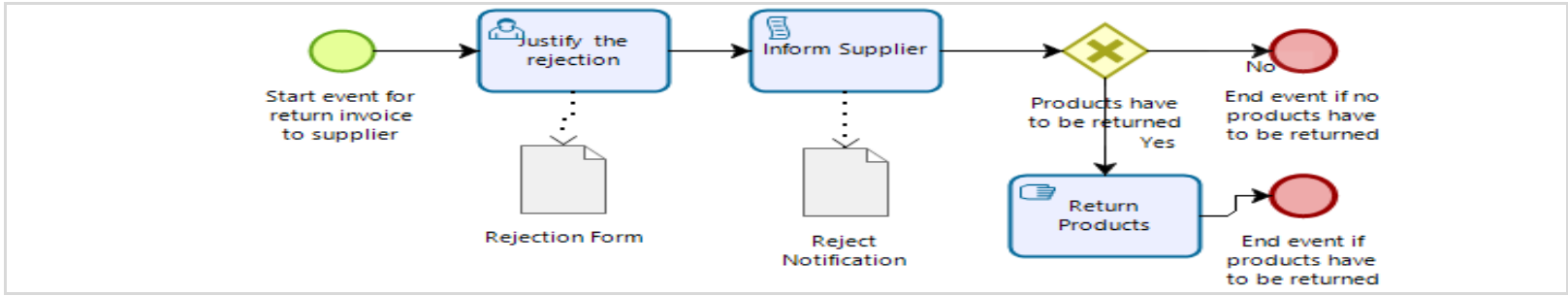
As informações básicas da ontologia foram as mesmas para ambos os documentos ERS, a saber: “Name”, “Description”, “Creation Date”, “Modification Date”, “Author” e “Documentation”.

Figura 44 - Modelo de processos de negócio “Accounts Payable”: (a) processo principal; (b) Subprocesso “Return Invoice to Supplier”.

(a)



(b)



Fonte: repositório público: *https://www.bizagi.com/en/community/process-xchange*.

Por outro lado, as seguintes informações foram específicas para cada documento ERS gerado: “Actors”, “Pre-conditions”, “Post-conditions”, “Business Rules”, “Functional Requirements”, “Non-functional requirements”.

A análise dos dois documentos ERS gerados verificou que a semântica original foi preservada, observando-se que todos os elementos relativos ao processo principal constavam no correspondente documento ERS, bem como todos os elementos do subprocesso constavam no documento ERS correspondente.

Foi constatado que cada documento ERS teve seus próprios diagramas de classe e de caso de uso, como esperado.

No documento ERS correspondente ao processo principal foram mapeados:

- 4 atores (“Receptionist”, “Financial Assistant”, “Administration Manager” e “Accountant”;
- 1 precondição (subclasse de *evt-Start* → *evt-Invoice_Arrival*);
- 1 pós-condição (subclasse de *evt-End* → *evt-Information_Updated*;
- 1 regra de negócio (atributo estendido “REGRA” da subclasse de *Activity* → *actv-Validate_Invoice*;
- 5 requisitos funcionais, referentes à descrição de cada atividade;
- 2 requisitos não funcionais, referentes ao atributo estendido “RNF” presentes na atividade “Update Financial ERP”.

No documento ERS correspondente ao subprocesso “Return Invoice to Supplier” foram mapeados:

- 1 ator (“Financial Assistant”);
- 1 precondição (subclasse de evt-Start);
- 2 pós-condições (duas subclasses de evt-End);
- 1 regra de negócio (atributo estendido “REGRA” da subclasse de Activity → actv-Inform_Supplier);
- 2 requisitos funcionais, referentes à descrição de cada subclasse de Activity diferente de actv-Manual (atividades manuais).

Discussão dos resultados

As informações constantes nos dois documentos ERS foram geradas de acordo com o esperado. Os diagramas UML também foram gerados de maneira satisfatória, visto que

preservaram os aspectos semânticos do modelo de processos de negócio representado na ontologia.

A **Tabela 14** apresenta a correspondência dos elementos da ontologia para os diagramas de casos de uso e de classes. É importante ressaltar que, com exceção dos *gateways* (subclasse ontológica *Gateway*), todos os demais elementos apresentam um relacionamento “1 para 1”.

Tabela 14 - Mapeamento da ontologia para os diagramas UML - modelo “Accounts Payable”.

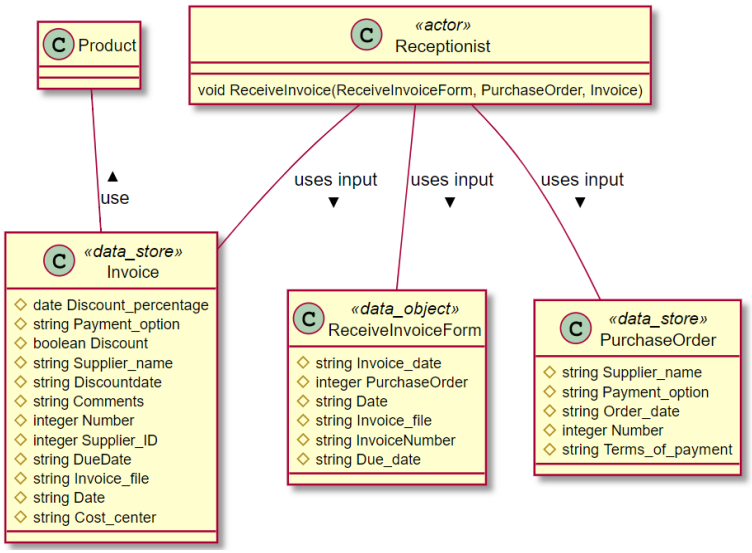
Elemento da ontologia	Diagrama de casos de uso	Diagrama de classes
Processo principal		
5 subclasses de *Activity*	5 casos de uso	
4 subclasses de *Actor*	4 atores	4 classes com estereótipo “actor”
3 subclasses de *gatw-Exclusive*	6 relacionamentos do tipo “extend” entre casos de uso	
2 subclasses de *DataObject*		2 classes com estereótipo “data_object”
2 subclasses de *DataStore*		2 classes com estereótipo “data_store”
1 relação *producesOutput*		1 tipo de saída de métodos
3 relações “*usesInput*”		3 parâmetros de entrada de métodos
28 atributos estendidos “ATRIBUTOS” de tipos “padrão”		28 atributos de dados
1 atributo estendido do tipo classe (*Seção 5.4.2* – Regra de Mapeamento RM2)		1 classe e seu relacionamento com a classe que contém o atributo estendido
Subprocesso “Return Invoice to Supplier”		
3 subclasses de *Activity* (1 do tipo manual é descartada)	2 casos de uso	
1 subclasse de *Actor*	1 ator	1 classe com estereótipo “actor”
1 *gateway* exclusivo	Não mapeado devido sua ligação com uma atividade do tipo “Manual”	
2 subclasses de *DataObject*		2 classes com estereótipo “data_object”
2 relações *producesOutput*		2 tipos de saída de método
10 atributos estendidos “ATRIBUTOS” de tipos “padrão”		10 atributos de dados

Fonte: elaborado pelo autor.

O conteúdo e organização dos diagramas UML gerados mostraram-se adequados à representação dos elementos do modelo de processos de negócio. Foi verificado que, no diagrama de casos de uso, todos os atores e casos de uso com suas respectivas relações foram gerados corretamente.

Em relação ao diagrama de classes, a **Figura 45** mostra um fragmento desse diagrama gerado para o processo principal. A classe “Product” não possui correspondente na ontologia nem no modelo de processos de negócio de origem. Essa classe é criada devido à regra de mapeamento **RM2**, apresentada na *Subseção 5.4.2*, que faz o mapeamento de atributos de dados que não fazem parte de tipos primitivos (*string*, *integer*, etc) para classes e seus respectivos relacionamentos.

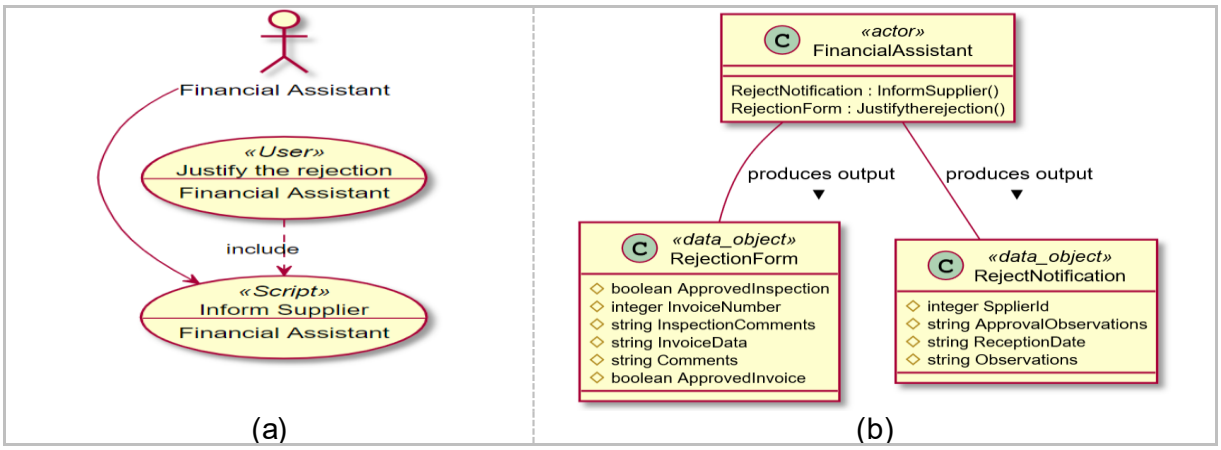
Figura 45 - Diagrama de classes para o modelo de processo “Accounts Payable” - processo principal.



Fonte: elaborado pelo autor.

Por sua vez, no diagrama de casos de uso do subprocesso “Return Invoice to Supplier”, mostrado na **Figura 46a**, o ator “Financial Assistant” é o responsável pelo caso de uso “Inform Supplier”. Como esse caso de uso está ligado ao caso de uso “Justify the Rejection” com um relacionamento do tipo “include”, isso significa que todas as vezes que o primeiro caso de uso for executado, o segundo será realizado, mantendo a coerência semântica com o modelo de origem.

Figura 46 - Diagramas UML gerados para o subprocesso “Return Invoice to Supplier”: (a) diagrama de casos de uso; (b) diagrama de classes.



Fonte: elaborado pelo autor.

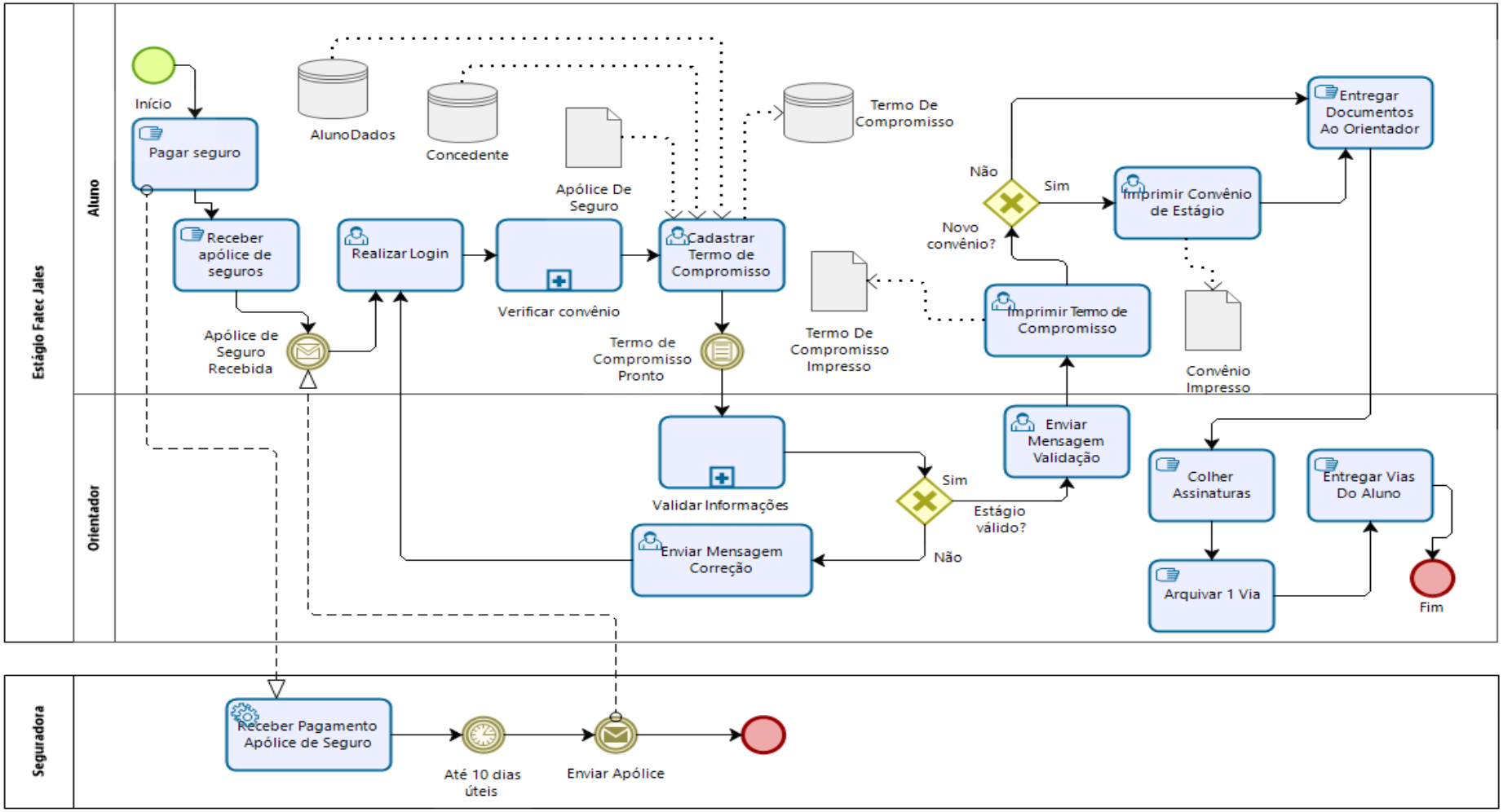
Por fim, na **Figura 46b**, a classe “Financial Assistant” (classe ontológica *actv-Financial_Assistant*) possui dois métodos que correspondem às subclasses ontológicas *Activity* (*actv-Rejection_Form* e *actv-Reject_Notification*). O relacionamento entre o ator e seus métodos é possível por meio tripla *Actor* → *isPerformedBy* → *Activity* presente na ontologia. Os tipos das saídas dos métodos são definidos de acordo com a tripla *Activity* → *producesOutput* → *Artifact*. Assim é possível mapear essas relações para o digrama de classes, mantendo a semântica do modelo de processos de negócio de entrada.

O APÊNDICE F traz o modelo “Acounts Payable” em BPMN v2.0 na versão original e com os ajustes adequados à aplicação do processo sistematizado, o arquivo em OWL da ontologia representativa do modelo, bem como o documento ERS completo.

6.3 Estudo de caso 3: modelo “Estágio Fatec Jales”

O estudo de caso apresentado nesta seção é referente a um modelo de processos de negócio identificado como “Estágio Fatec Jales. Esse modelo é a representação do processo de formalização de estágios supervisionados que todos os alunos dos cursos superiores da Faculdade de Tecnologia de Jales “Prof. José Camargo” (Fatec Jales) devem realizar. O modelo foi construído pelos docentes que foram orientadores de estágio da Fatec em 2019, sem contar com nenhum sistema de software para o processo. Na **Figura 47**, é apresentado o modelo principal, onde pode ser observado que há mais dois subprocessos.

Figura 47 - Modelo de processos de negócio “Estágio Fatec Jales” - processo principal.



Fonte: elaborado pelo autor.

Conformidade com os critérios definidos para a geração da ontologia.

Inicialmente, foi avaliado se o modelo estava em conformidade com os critérios requeridos, como apresentado na *Subseção 2.3.2*. Foi necessário fazer ajustes no modelo de modo que ele satisfizesse os critérios C5, C7 e C10, observando-se que após os ajustes, o modelo continuou com o mesmo número de elementos. Assim:

- todas as atividades tiveram seus respectivos tipos definidos (C5);
- todas as atividades foram associadas a um ator (participante) (C7);
- os fluxos dos *gateways* foram devidamente nomeados (C10).

Além dos critérios citados, as heurísticas de requisitos e de negócios também foram utilizadas. A atividade “Realizar Login” (processo principal) foi documentada com os atributos estendidos “RNF” (HR4) e “REGRA” (HR6). A atividade “Cadastrar Concedente” (subprocesso “Verificar convênio”) foi documentada com o atributo estendido “REGRA”. Ademais, todos os objetos e repositórios de dados foram documentados com o atributo estendido “ATRIBUTOS” (HNd9 e HNd10 - ver *Seção 4.3*, **Tabela 6**). Após os ajustes, o processo principal (**Figura 47**) do modelo passou a ser composto por:

- 4 eventos: 2 do tipo “Início” e 2 do tipo “Fim”;
- 2 piscinas;
- 3 raias;
- 12 atividades: 6 do tipo “Manual”, 5 do tipo “Usuário”, 1 do tipo “Serviço”;
- 2 subprocessos do tipo “Simples”;
- 5 artefatos: 3 do tipo “Repositório de dados” e 2 do tipo “Objeto de dados”;
- 2 *gateways* do tipo “Exclusivo”;
- 3 eventos intermediários: 2 do tipo “Mensagem” e 1 do tipo “Condicional”.

Após os ajustes, o subprocesso “Validar Informações” passou a ser composto por:

- 3 eventos: 1 do tipo “Início” e 2 do tipo “Fim”;
- 3 atividades: 1 do tipo “Manual” e 2 do tipo “Usuário”;
- 3 artefatos: 3 do tipo “Repositório de dados”;
- 1 *gateway* do tipo “Exclusivo”.

Após os ajustes, o subprocesso “Firmar Convênio”” passou a ser composto por:

- 2 eventos: 1 do tipo “Início” e 1 do tipo “Fim”;

- 4 atividades: todas do tipo “Usuário”;
- 2 artefatos: ambos do tipo “Repositório de Dados”;
- 2 *gateways* do tipo “Exclusivo”.

Geração e análise da ontologia

Após exportar o modelo para o formato XPLD v2.2, o arquivo gerado foi usado como entrada no sistema PM2ONTO v2.0, que gerou a ontologia representativa do modelo. Foram, então, identificadas algumas inconsistências que precisaram ser corrigidas neste caso, mas que serviram de requisitos para uma nova versão do sistema PM2ONTO, além dos problemas já mostrados na *Seção 5.2*:

- A classe *actor-Orientador* possui uma relação com a classe *subProcess-Verificar_Convenio*. Essa relação teve que ser excluída, pois, segundo o modelo de processos de negócio, o participante “Orientador” não faz parte do referido subprocesso;
- Nenhum relacionamento do tipo *usesInput* e *producesOutput* foi criado nos subprocessos. Assim, foi necessário editar a ontologia, adicionando os devidos relacionamentos entre as subclasses de *Activity* e *DataStore* e/ou *DataObject*.

Geração e análise do documento ERS

Os documentos ERS gerados foram considerados semanticamente adequados. Convém destacar que foram gerados três documentos ERS, que compõem o ERS final: 1 para o processo principal e 1 para cada subprocesso. É importante observar que a piscina identificada como “Seguradora” no modelo da **Figura 47** foi ignorada, visto que suas atividades não possuem participantes. Na **Tabela 15** é mostrado o resumo do mapeamento entre os elementos ontológicos e dos diagramas UML.

Discussão dos resultados

As informações constantes nos dois documentos ERS foram geradas de acordo com o esperado. Mais uma vez foi observado que cada elemento da ontologia foi mapeado para, no mínimo, um elemento do documento ERS.

Os diagramas UML também foram gerados de maneira satisfatória, visto que preservaram os aspectos semânticos do modelo de processos de negócio representado na ontologia, usando as todas as regras de mapeamento apresentadas na *Seção 5.4*.

Tabela 15 - Mapeamento da ontologia para os diagramas UML – modelo "Estágio Fatec Jales" (*continua...*)

Elemento da ontologia	Diagrama de casos de uso	Diagrama de classes
Processo principal		
7 subclasses de *Activity*	7 casos de uso	
2 subclasses de *Actor*	2 atores	2 classes com estereótipo "actor"
2 subclasses de *gatw-Exclusive*	3 relacionamentos do tipo "extend" entre casos de uso	
3 subclasses de *DataObject*		3 classes com estereótipo "data_object"
3 subclasses de *DataStore*		3 classes com estereótipo "data_store"
3 relações *producesOutput*		3 tipos de saídas de métodos
3 relações "*usesInput*"		3 parâmetros de entrada de métodos
58 atributos estendidos "ATRIBUTOS" de tipos "padrão"		58 atributos de dados
9 atributos estendido do tipo classe (*Seção 5.4.2* - Regra de Mapeamento RM2)		2 classes (Cidade e Aluno) e relacionamentos com a classe que contém o atributo estendido. 7 relacionamentos com classes existentes.
Subprocesso "Firmar Convênio"		
4 subclasses de *Activity* (do tipo manual é descartada)	4 casos de uso	
1 subclasse de *Actor*	1 ator	1 classe com estereótipo "actor"
2 *gateways* do tipo "Exclusivo"	4 relacionamentos do tipo "extend" entre casos de uso	
2 subclasses de *DataStore*		2 classes com estereótipo "data_store"
1 relação *producesOutput*		1 tipos de saída de método
3 relações *usesInput*		3 parâmetros de entrada de métodos
23 atributos estendidos "ATRIBUTOS" de tipos "padrão"		23 atributos de dados
2 atributos estendido do tipo classe (*Seção 5.4.2* - Regra de Mapeamento RM2)		1 classe (Cidade) e seu relacionamento; 01 relacionamento entre Concedente e Convênio
Subprocesso "Validar Informações"		
2 subclasses de *Activity* (do tipo manual é descartada)	2 casos de uso	
1 subclasse de *Actor*	1 ator	1 classe com estereótipo "actor"

(*...continuação*) **Tabela 15** - Mapeamento da ontologia para os diagramas UML – modelo "Estágio Fatec Jales"

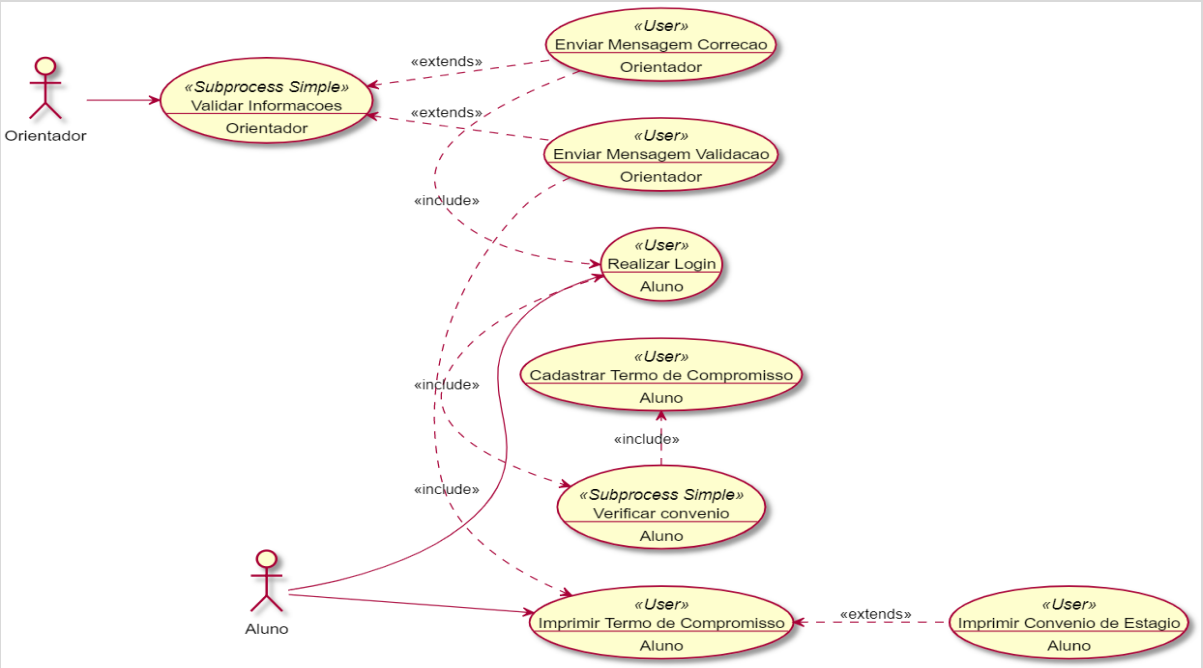
1 *gateway* do tipo "Exclusivo"	1 relacionamento do tipo "extend" entre casos de uso	
3 subclasses de *DataStore*		3 classes com estereótipo "data_store"
3 relações *usesInput*		3 parâmetros de entrada de métodos
35 atributos estendidos "ATRIBUTOS" de tipos "padrão"		35 atributos de dados
2 atributos estendido do tipo classe (*Seção 5.4.2* - Regra de Mapeamento RM2)		2 classes (Cidade e Curso) e seus relacionamentos com as classes Concedente e Aluno

Fonte: elaborado pelo autor.

Mais uma vez a ontologia gerada pelo sistema PM2ONTO v2.0 mostrou-se adequadamente representativa do modelo de processos de negócio considerado.

Na **Figura 48** é possível identificar todos os casos de uso correspondentes às subclasses ontológicas *Activity* que não são do tipo "Manual", além dos relacionamentos com seus respectivos atores e entre os próprios casos de uso.

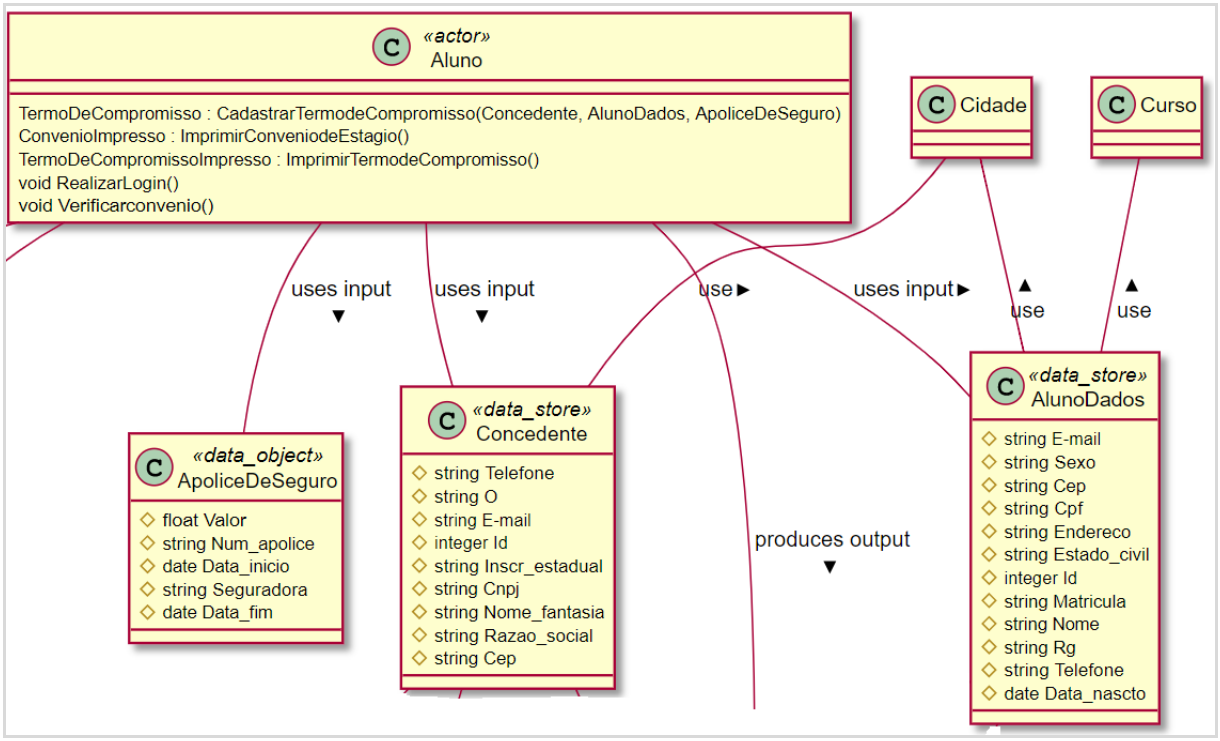
Figura 48 - Diagrama de casos de uso para o modelo "Estágio Fatec Jales" - processo principal.



Fonte: elaborado pelo autor.

Na **Figura 49** é mostrada uma parte do diagrama de classes, com destaque às classes “Curso” e “Cidade”, pois elas foram mapeadas pela regra de mapeamento **RM2** (*Seção 5.4*), onde um atributo de tipo não primitivo se torna uma classe com seu respectivo relacionamento. Na **Figura 49** também pode ser observado que a classe “Aluno”, com estereótipo “actor,” possui 5 métodos, que correspondem às atividades do participante “Aluno” no modelo de processos de negócio. O método “Cadastrar Termo de Compromisso” retorna um tipo “TermoDeCompromisso” (*Activity → producesOutput → Artifact*) e possui três parâmetros de entrada que são os relacionamentos do tipo usesInput (*Activity → usesInput → Artifact*), a saber: “ApóliceDeSeguro”, “AlunoDados” e “Concedente”. Esses relacionamentos no diagrama UML são semanticamente idênticos aos elementos da ontologia e, principalmente, ao modelo de processos de negócio considerado.

Figura 49 - Fragmento do diagrama classes para o modelo “Estágio Fatec Jales” - processo principal.



Fonte: elaborado pelo autor.

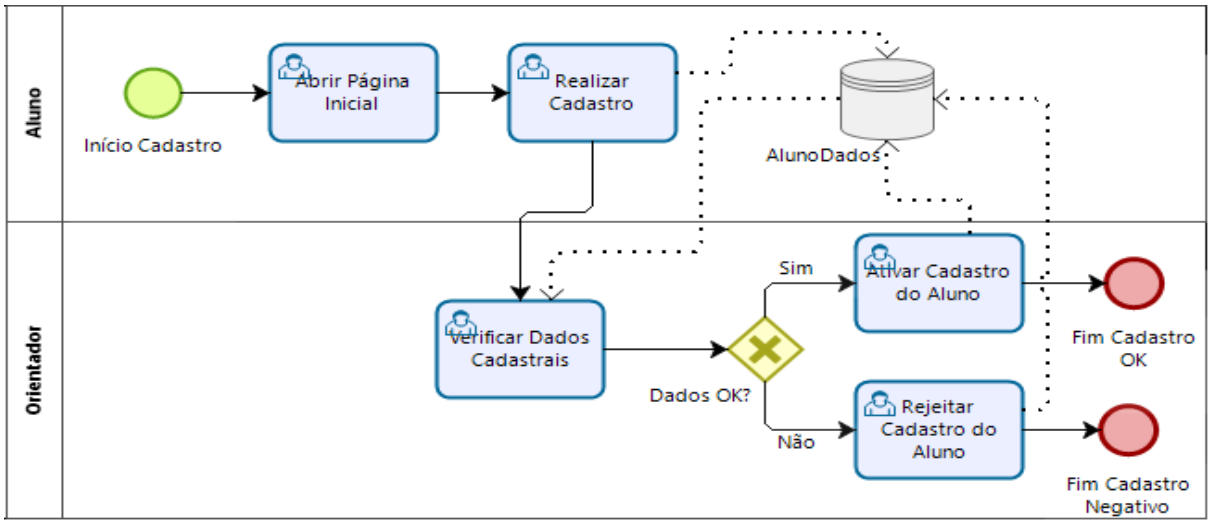
O APÊNDICE G traz o modelo “Estágio Fatec Jales” em BPMN v2.0 completo já com os ajustes adequados à aplicação do processo sistematizado, o arquivo em OWL da ontologia representativa do modelo, bem como o documento ERS final completo.

6.4 ESTUDO DE CASO 4: INTEGRAÇÃO ENTRE ONTOLOGIAS

Esta seção objetiva avaliar a integração de dois modelos de processos de negócio em BPMN v2.0 por meio da integração das suas respectivas ontologias representativas (ver *Seção 3.6*).

Para isso, um dos modelos considerado foi modelo identificado como “Estágio Fatec Jales”, usado para o estudo de caso da *Seção 6.3*. Um segundo modelo foi utilizado, identificado com o “Estágio Fatec Jales Cadastro” (**Figura 50**), com as atividades para cadastramento dos alunos para estágios. Observa-se que esses dois modelos estão relacionados a um mesmo tema, de modo que o segundo poderia ser integrado com o primeiro modelo.

Figura 50 - Modelo de processos de negócio “Estágio Fatec Jales Cadastro”.



Fonte: elaborado pelo autor.

Esse segundo modelo faz uso de elementos já presentes no primeiro modelo, como, por exemplo:

- participantes “Aluno” e Orientador;
- repositório de dados “AlunoDados”;
- evento de início “Início”.

Observa-se que os eventos de fim “Fim Cadastro OK” e “Fim Cadastro Negativo”, bem como o *gateway* “Dados OK?” são elementos que constam somente no segundo modelo.

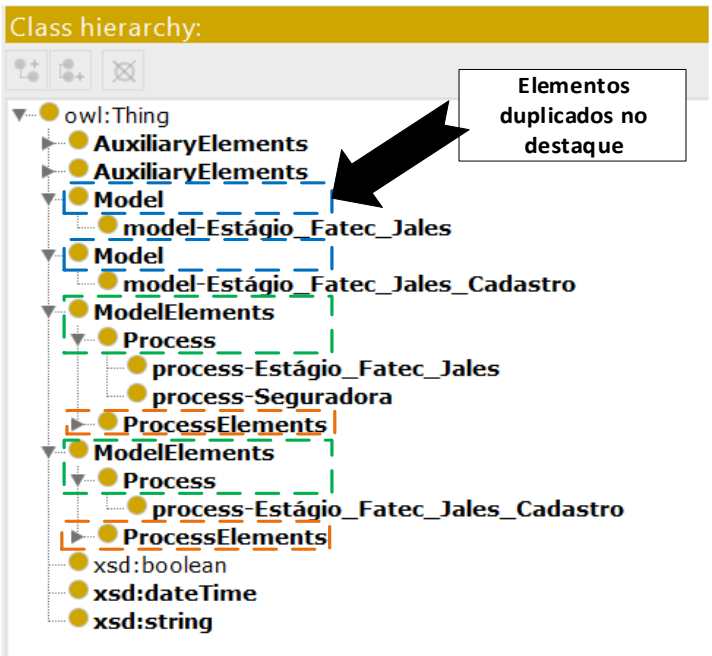
Geração e análise da ontologia

Após exportar os dois modelos para o formato XPLD v2.2, os arquivos gerados foram usados como entrada no sistema PM2ONTO v2.0, que gerou as respectivas ontologias representativas dos modelos, sendo que a do modelo “Estágio Fatec Jales” já foi analisada na *Seção 6.3*. Assim, considerar como primeira ontologia a que representa o modelo “Estágio Fatec Jales” e a segunda como sendo a representativa do modelo “Estágio Fatec Jales Cadastro”.

Então, os arquivos OWL foram importados e integrados, de acordo com o processo de fusão (*merging*), apresentado na *Seção 3.6*. Para isso, foi utilizado o editor de ontologias *Protégé* v5.2.0, de modo a unir as duas ontologias, onde a primeira ontologia “recebeu” a segunda ontologia, preservando a segunda e modificando a primeira.

O problema dessa abordagem é que são preservados elementos duplicados na ontologia modificada. Na **Figura 51**, é mostrada parte da hierarquia de classes da primeira ontologia após ser fundida com a segunda ontologia. Como pode ser observado, os elementos da hierarquia são duplicados.

Figura 51 – Fragmento das classes da primeira ontologia após a fusão, com elementos duplicados.



Fonte: elaborado pelo autor.

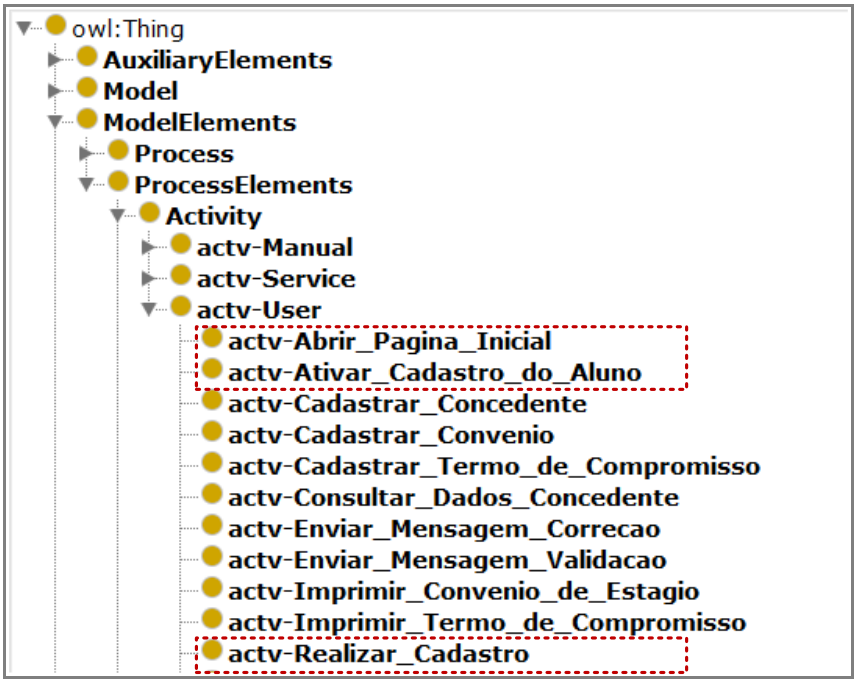
Assim, o processo de alinhamento (*matching*) poderia ser mais adequado nesse cenário, uma vez que permitiria identificar a ontologia de origem dos elementos duplicados (ver *Seção 3.6*). Porém, com o objetivo de evitar fazer o *matching* manualmente, antes de

fazer o *merging* das ontologias foram realizadas as seguintes modificações na segunda ontologia, devido aos elementos duplicados:

- todos os IRIs (*Internationalized Resource Identifiers*) dos recursos foram renomeados usando o mesmo nome do IRI da primeira ontologia ("EstagioFatecJalesMerged"). No editor *Protégé*, isso é possível usando o menu/comando "*Refactor*/*Rename multiple entities...*";
- todas as informações de cabeçalho da ontologia também foram editadas para "EstagioFatecJalesMerged", usando um editor de textos.

Assim, a fusão entre as duas ontologias foi realizada usando o menu/comando "*Refactor*/*Merge* do editor *Protégé*, com as duas ontologias "abertas" na mesma janela. Observa-se que, embora a primeira ontologia tenha sido modificada, uma cópia foi feita com o objetivo de preservá-la originalmente. Na **Figura 52**, as classes em destaque são oriundas da segunda ontologia, o que significa que a fusão (*merging*) foi realizada com sucesso.

Figura 52 - Fragmento das classes da primeira ontologia após a fusão, com destaque às novas classes da segunda.



Fonte: elaborado pelo autor.

No entanto, foi necessário analisar se as relações preservavam a semântica do domínio em questão. Após a verificação dos novos elementos, foi constatado que a relações estão definidas corretamente.

Além de cinco novas subclasses de *actv-User*, referentes às atividades do segundo modelo de processos de negócio (*act-Abrir_Pagina_Inicial*, *act-Realizar_Cadastro*, *act-Verificar_Dados_Cadastrais*, *act-Ativar_Cadastro_Do_Aluno* e *act-Rejeitar_Cadastro_Do_Aluno*), foram criadas:

- uma subclasse de *gatw-Exclusive* (*gatw-Dados_OK)*;
- uma subclasse de *evt-Start* (*evt-Start_Inicio_Cadastro*);
- duas subclasses de *evt-End* (*evt-End_Fim_Cadastro_OK* e *evt-End_Fim_Cadastro_Negativo*).

Convém ressaltar que, após o processo de fusão (*merging*), todas essas novas classes criadas tiveram seus IRIs modificados, perdendo a informação sobre sua origem, não sendo possível identificar que pertenciam à segunda ontologia. Além disso, os novos elementos possuem relação *isPartOfProcess* com a subclasse de *Process* identificada como *process-Estágio_Fatec_Jales,* não sendo gerada uma subclasse específica para o novo processo. Isso significa que todas as referências do modelo de processos de negócio que originou esses novos elementos foram realmente perdidas.

Geração e análise do documento ERS

A primeira ontologia modificada pela fusão foi então usada como entrada para o sistema OnToSRS a fim de verificar o resultado do documento ERS.

Considerando os documentos ERS da primeira ontologia original (como na *Seção 6.3*) e dessa ontologia já modificada pelo processo de fusão, foi observado que o processo principal da ontologia original recebeu novos elementos e houve as atualizações na estrutura do documento ERS. Entretanto não foram devidamente mapeados todos os elementos, como pode ser visto na **Figura 53**, cabendo as seguintes considerações sobre o documento ERS gerado após a fusão:

- ***Informações básicas*** - as informações básicas do cabeçalho do documento foram mantidas. É necessário pensar numa estratégia para inserir as informações da nova ontologia inserida;
- ***Preconditions*** - a nova precondição, referente ao evento de início (subclasse de "evt-Start") não foi mapeada. Isso acontece porque o algoritmo de geração do documento ERS considera apenas um evento de início para o modelo;

Figura 53 - Informações do documento ERS adicionadas (em destaque) após a fusão das ontologias.

Actor(s):	**Orientador**: Professor responsavel pela formalizacao do estagio **Aluno**: Aluno que ira iniciar o estagio
Pre-conditions:	Aluno precisa estar matriculado em pelo menos uma disciplina.
Post-conditions:	**1**:Aluno cadastrado no sistema de estágio. **2**:O cadastro do aluno é rejeitado pelo Oritentador **3**:Convênio de concessão de estágio firmado;Estágio formalizado;
Business Rules:	**1**:O aluno precisa estar cadastrado no sistema e com a matrícula ativa em pelo menos uma disciplina. **2**:O aluno precisa estar regularmente matriculado em pelo menos uma disciplina;
Functional Requirements	**1**:Enviar mensagem com as inconsistências dos documentos **2**:Aluno imprime termo de compromisso - 3 vias. **3**:Rejeitar Cadastro do Aluno **4**:Enviar mensagem validando informações. **5**:Aluno imprime convênio de estágio - 2 vias **6**:Orientador deve verificar no sistema e na secreataria a validade dos dados. **7**:O aluno deve abrir a página inicial do sistema. **8**:O aluno deve realizar o seu cadastro **9**:Aluno realiza login no sistema **10**:Cadastrar termo de compromisso - Anexo 3 - Documentos de estágio **11**:Ativar o cadastro do aluno
Non-Functional Requirements	**1**:A operação de realizar login não pode demorar mais do que 15 segundos.

Fonte: elaborado pelo autor.

- ***Post-conditions*** - foram adicionadas novas pós-condições referentes às classes ontológicas que representam os novos eventos de fim do processo: “Aluno cadastrado no sistema de estágio” e “O cadastro do aluno é rejeitado pelo Orientador”;
- ***Business Rules*** - a regra de negócio do novo modelo, representada pelo atributo “REGRA” documentado na atividade “Verificar Dados Cadastrais” no modelo de origem foi adicionado ao documento;
- ***Functional Requirements*** - as descrições das novas subclasses de *Activity*, que representam as atividades do tipo “Usuário” do novo modelo, foram incluídas da lista de requisitos funcionais.

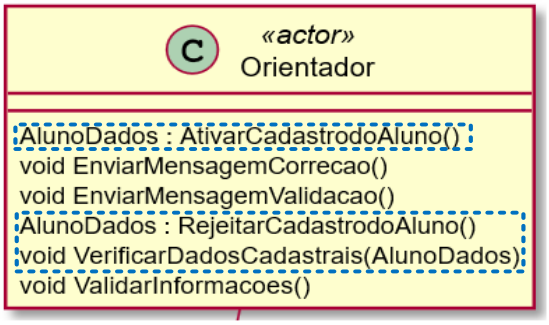
Discussão dos resultados

As informações constantes no documento ERS foram apresentadas como esperado. Em relação aos diagramas UML que foram gerados, apresentaram os novos elementos adicionados na ontologia após a fusão com a segunda ontologia, como esperado.

Na **Figura 54**, é mostrada parte do diagrama de classes, onde a classe “Orientador” aparece modificada, contendo novos métodos que correspondem às novas atividades

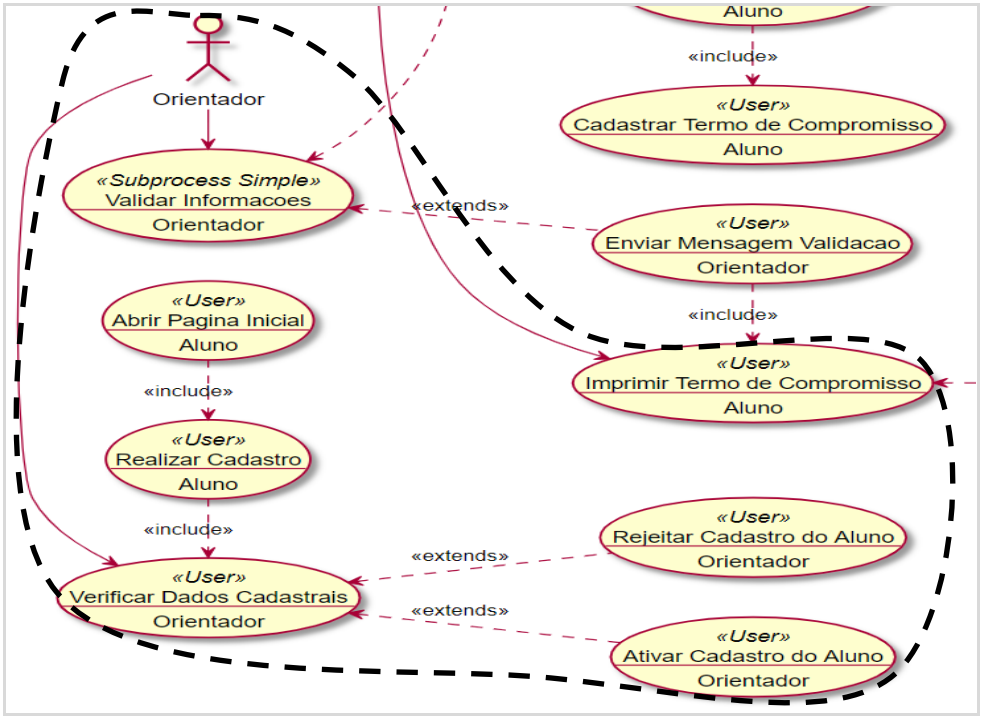
representadas na segunda ontologia. A classe “Aluno” também apresentou modificações. Na **Figura 55**, é mostrado uma parte do diagrama de casos de uso com destaque aos novos elementos.

Figura 54 - Classe “Orientador” com os novos métodos em destaque.



Fonte: elaborado pelo autor.

Figura 55 – Fragmento do diagrama de casos de uso, com destaque aos novos elementos.



Fonte: elaborado pelo autor.

Entretanto, houve perda de informações sobre a nova precondição (evento de início), sobre a documentação (descrição do processo ou modelo) e, principalmente, sobre a origem das novas informações. Isso pode comprometer questões de rastreabilidade dos requisitos, impactando na colaboração durante o processo de modelagem. Assim, é necessário

desenvolver uma estratégia para integrar diferentes ontologias, de modo que as novas informações sejam devidamente referenciadas, com o objetivo de saber sua origem.

O APÊNDICE H traz os modelos de processos de negócio considerados, os arquivos usados para fazer a fusão (*merging*) e documento ERS completo gerado sobre a primeira ontologia modificada.

6.5 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada a avaliação do processo sistematizado, implementado no sistema OnToSRS, para a geração de documentos ERS a partir de ontologias de processos geradas pelo sistema PM2ONTO v2.0. Convém lembrar que as informações básicas do documento ERS são mapeadas de maneira direta, por meio da execução de consultas SPARQL em determinadas propriedades da ontologia. Já os diagramas UML necessitam de processamento adicional, visto que a recuperação dos dados via SPARQL deve seguir a estrutura do metamodelo Meta2Onto, proposto por Figueiredo (2018). Isso significa que não é possível extrair um diagrama de casos de uso ou de classes de maneira direta, sendo necessária a aplicação dos algoritmos apresentados na *Seção 5.4*.

Apesar do diagrama de casos de uso não possibilitar expressão de todos os aspectos do modelo de processos de negócio, ele manteve a semântica e a pragmática dos modelos. O diagrama de classes mostrou resultados satisfatórios, pois mostrou que foi possível representar as classes, suas associações, métodos e atributos, sendo que essas informações são suficientes para gerar um modelo PSM, considerando o padrão MDA.

Os estudos de caso apresentados buscaram usar modelos de processos de negócio que evidenciassem aspectos importantes dos processos, principalmente para o contexto da Engenharia de Requisitos. É importante enfatizar que as boas práticas de modelagem de processos de negócio e a documentação textual (ver *Capítulo 2*) foram, de fato, determinantes para a construção de modelos expressivos, considerando as qualidades semântica e pragmática. Nos estudos de caso, ficou evidente que a representatividade da ontologia gerada depende das boas práticas de modelagem, que foram apresentadas na *Seção 2.3,* além das heurísticas de negócios e requisitos, apresentadas na *Seção 4.3*.

Os estudos de caso apresentados nas *Seções 6.2*, *6.3* e *6.4* possibilitaram perceber os benefícios da representação ontológica dos modelos de processos de negócio, pois elas permitiram a complementação de informações, visando expressar aspectos que não são contemplados nos modelos. Com o estudo de caso apresentado na *Seção 6.4*, foi possível

fazer uma integração entre dois modelos de processos de negócio por meio da fusão (*merging*) entre duas ontologias distintas de mesmo domínio. Isso é relevante, considerando que a integração entre diferentes sistemas e estruturas ontológicas pode trazer mais flexibilidade ao tratamento e uso de modelos de processos de negócio representados. Como destacado na *Seção 6.4*, o processo de *merging* (fusão) foi realizado manualmente, o que causou perda de informações. Nesse sentido, é necessário propor métodos para que o a ontologia atualizada consiga expressar, de maneira clara, a origem das novas informações. Dessa maneira, será possível gerar um documento ERS considerando a rastreabilidade dos requisitos.

Frente aos resultados obtidos com a composição dos documentos ERS´s gerados, foi possível perceber que constituem valiosas fontes de informações organizadas, de modo a contribuir com os processos de Engenharia de Requisitos para a automação dos processos. A versão do documento servirá de apoio aos profissionais de TI, preocupados em compreender o ambiente de negócio (domínio), possibilitando complementação à medida que for se estreitando a interação com as equipes de negócio.

7 CONCLUSÃO E CONSIDERAÇÕES FINAIS

Este trabalho apresentou um processo sistemático para a geração de um documento de especificação de requisitos de software a partir de ontologias representativas de modelos de processos de negócio na notação BPMN v.2.0. O processo, apresentado no *Capítulo* 5, foi implementado no sistema OnToSRS (*Ontology to Software Requirements Specification*), o qual extrai informações da ontologia de processos (entrada do sistema) e gera, automaticamente, documento(s) denominado(s) "Especificação de Requisitos de Software" (ERS), segundo o padrão ISO/IEC/IEEE 29148:2018. Caso o modelo de processos de negócio tenha "n" subprocessos, o sistema OnToSRS gera "n+1" documentos ERS parciais, sendo um relativo ao processo principal do modelo. Esses ERS parciais (ou simplesmente ERS) compõem o documento ERS final, relativo ao modelo de processos de negócio como um todo. Um documento ERS é composto por atores, precondições, pós-condições, regras de negócio, requisitos funcionais, requisitos não funcionais e diagramas da linguagem UML (casos de uso e classes).

O mapeamento entre os elementos da ontologia e os elementos do documento ERS considera a semântica desses elementos, sendo 1:1 (um para um). Assim, cada elemento do documento ERS tem um correspondente na ontologia. A geração dos diagramas UML, por outro lado, requer processamento adicional, visto que são mapeados a partir de dois ou mais elementos da ontologia. Todos os aspectos desses mapeamentos foram explorados nos estudos de caso apresentados no *Capítulo 6*.

O documento ERS gerado contribui para que a equipe de desenvolvimento de software entenda o domínio do processo no ambiente de negócio e tenha informações de apoio para dar continuidade aos processos da Engenharia de Requisitos, complementando o documento. Essa aproximação da equipe de desenvolvimento de software com a equipe de negócios propicia automação dos processos de modo mais aderente às necessidades dos usuários.

Para o desenvolvimento do processo sistemático apresentado, as pesquisas sobre Engenharia de Requisitos foram importantes para entender trabalhos que geram documentos de especificação de requisitos de software diretamente de modelos de processos de negócio em BPMN, como Nogueira (2017), por exemplo (*Capítulo 4*). Esses trabalhos definem mapeamentos entre diferentes modelos com alto nível de abstração, objetivando adequar esses modelos a diferentes perfis de usuários, sob a perspectiva MDA (*Model-Driven Architecture)*, como neste trabalho.

É importante mencionar que modelos de processos de negócio têm sua navegação e legibilidade comprometidas à medida que ficam mais extensos. Assim, a representação ontológica ajuda na compreensão e navegação do modelo, evidenciando também as interdependências entre seus elementos. Considerando que a representação do conhecimento implícito é complexa e não tem padrão para os modelos de processos de negócio, a representação ontológica contribui para evidenciar esse conhecimento. Assim, as ontologias contribuem para a conceituação e organização da informação implícita e desestruturada que se encontra presente nos processos de negócio e que deve ser explorada. Desse modo, pode-se estruturar o conhecimento implícito presente nos processos de negócio, possibilitando a compreensão desse conhecimento por máquina.

Para que a estrutura ontológica possibilite a extração de informações de modo adequado à geração automática do texto e diagramas UML do documento ERS, os modelos de processos de negócio devem ser documentados e estarem em conformidade com boas práticas de modelagem especificadas neste trabalho. Para este trabalho, foram considerados os doze critérios definidos por Figueiredo (2018), além de algumas das heurísticas de requisitos e heurísticas de negócios propostas por Nogueira (2017), apresentadas na *Seção 4.3*. Isso aumenta a qualidade semântica e pragmática do modelo de processos de negócio, sendo determinante para não haver perdas na representação ontológica do modelo.

As ontologias em OWL usadas como entrada para o sistema OnToSRS são geradas pelo sistema PM2ONTO v2.0, que estendeu a v1.0 de Figueiredo (2018), inserindo documentação de atributos de dados na ontologia. Isso possibilitou que os diagramas de classes mostrem atributos de dados e seus respectivos tipos. Ressalta-se que, para o desenvolvimento dessa extensão, foi importante o código fonte estar aberto e ter a possibilidade de interação com o autor do sistema original.

Como as ontologias são usadas para representação do conhecimento, sendo úteis na formalização de domínios, constituem uma alternativa significativa para representação de modelos de processos de negócio de modo mais expressivo em diversos níveis e contextos.

Isso contribui para o compartilhamento dos modelos, bem como com a integração com outros modelos de processos de negócio. Nessa direção, o *Capitulo 6* apresenta um estudo de caso que analisa a integração de dois modelos de processos de negócio por meio da integração de suas respectivas representações ontológicas, geradas pelo sistema PM2ONTO v2.0.

Trabalhos futuros

Com a mesma abordagem utilizada neste trabalho, o documento ERS gerado pelo sistema OnToSRS, pode ser complementado, automaticamente, com mais informações textuais e gráficas. Por exemplo, os diagramas de classes podem ser expressar aspectos de herança, cardinalidade e polimorfismo. Para isso, as ontologias podem ser complementadas, adicionando-se o conhecimento necessário, manual ou automaticamente. Observa-se que a ontologia de processos considerada neste trabalho é decorrente do trabalho de Figueiredo (2018), que não contempla todos os elementos gráficos de BPMN v2.0. Além disso, outros diagramas de UML também poderiam ser considerados em trabalhos futuros, como diagrama de atividades, pacotes, máquina de estados e outros.

Outra direção de trabalho futuro é a extração de requisitos de software a partir de outras estruturas ontológicas, como, por exemplo, estruturas que formalizam corpos de conhecimento como os guias SWEBoK (*Software Engineering Body of Knowledge*) e REBoK (*Requirements Engineering Body of Knowledge*). Ontologias específicas de um domínio também podem ser consideradas, como de Medicina ou Direito, por exemplo. Outra estrutura ontológica que pode ser considerada como fonte para a extração de requisitos é a "BPMN Ontology", proposta por Rospocher, Ghidini e Serafini (2014). Essa ontologia formaliza a especificação de BPMN v2.0 usando a linguagem OWL DL, seguindo a semântica oficial da notação BPMN.

Convém observar que a escolha da linguagem OWL, ao invés de RDF, propicia ontologias com maior grau de completude, uma vez que possibilita a representação dos aspectos formais da especificação de BPMN em sua totalidade, além de outros aspectos que o engenheiro ontológico julgar necessário. Ademais, a linguagem OWL permite a utilização de mecanismos de inferência (*reasoners*) por meio de axiomas, possibilitando o processamento e não apenas a representação dessas informações, o que pode ser explorado em trabalhos futuros. Nesse sentido, modos de se inferir novos conhecimentos acerca do modelo de processos de negócio podem ser definidos, bem como modos de análises mais completas sobre a interdependência entre os elementos de modelagem.

A integração entre diferentes ontologias permite integrar diferentes modelos de processos de negócio e, consequentemente, a extração de requisitos de software alcança outra dimensão. Contudo, é necessário considerar aspectos adicionais na geração da ontologia, de modo a permitir o mapeamento adequado para o documento ERS. Somente com os aspectos considerados neste trabalho, o estudo de caso apresentado na *Seção 6.4* mostrou que houve perda de informações com a integração das ontologias representativas dos dois modelos de processos de negócio considerados.

Na mesma linha de integração de ontologias, um trabalho futuro poderia ser a definição de um processo sistematizado para fazer o "*merging*" entre diferentes ontologias, de maneira a preservar informações sobre as ontologias integradas. Isso possibilitaria a rastreabilidade dos requisitos, modelagem colaborativa e a integração entre diferentes sistemas, modelos e ambientes.

Outra linha de trabalho pode ser direcionada a considerar modelos de processos de negócio com a notação BPMN e com a notação DMN (*Decision Model and Notation*), usada para modelagem de decisões e regras de negócio. Adicionalmente, poderiam ser considerados modelos que também utilizassem a notação CMMN (*Case Management Model and Notation*), utilizada para gerenciamento de casos específicos. Assim como BPMN, as notações DMN e CMMN também são especificadas pelo consórcio OMG.

Em outra direção, a geração automática de código poderia ser considerada em trabalhos futuros, implementando um mapeamento PIM (*platform-independent model*) para PSM (*platform-specific model*), na abordagem MDA. Nesse sentido, as ontologias para geração de modelos PSM poderiam ser integradas usando *frameworks* que possibilitem a geração de código fonte. Para isso, seria necessário definir um conjunto de regras, que poderiam ser baseados nas boas práticas de modelagem de processos de negócio apresentadas no *Capítulo 2*. Assim, o sistema OnToSRS poderia ser integrado a ferramentas já disponíveis que usam a abordagem MDA para apoiar o desenvolvimento de software.

REFERÊNCIAS

ABPMP BRAZIL – ASSOCIATION OF BUSINESS PROCESS MANAGEMENT PROFESSIONALS. **BPM CBOK** – Guia para o Gerenciamento de Processos de Negócio – Corpo Comum de Conhecimento. Association of Business Process Management, 2013. 441p.

AVDEENKO, T. V.; PUSTOVALOVA, N. V. The Ontology-Based Approach to Support the Requirements Engineering Process. In: INTERNATIONAL SCIENTIFIC-TECHNICAL CONFERENCE. 13., 2016. Novosibirsk. **Proceedings…** Novosibirsk: IEEE Computer Society, 2016. p.513-518.

BALDAM, R.; VALLE, R.; ROZENFELD, H. **Gerenciamento de Processos de Negócio BPM**: Uma referência para implantação prática. Rio de Janeiro: Elsevier, 2014. 777p.

BARTUSSEK, W.; BENSE, H.; HOPPE T.; HUMM B. G.; REIBOLD A.; SCHADE, U.; SIEGEL M.; WALSH P. Introduction to Semantic Applications. In: HOPPE, T.; REIBOLD, A.; HUMM, B. (ed.). **Semantic Applications:** Methodology, Technology, Corporate Use. Springer Verlag, 2018. p. 1-12.

BÍBLIA, A. T. Isaías. In BÍBLIA. Português. **Bíblia de estudo King James 1611**. Niterói: BV Films Editora, 2017.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML - Guia do Usuário**. 2. ed. Rio de Janeiro: Elsevier, 2005.

BORGER, E. Approaches to modeling business processes: a critical analysis of BPMN, workflow patterns and YAWL. **Software & Systems Modeling**, v. 3, p. 305-318, 2012.

BOUZIDI, A.; HADDAR, N.; ABDALLAH, M. B.; HADDAR K. Deriving Use Case Models from BPMN Models. In: International Conference on Computer Systems and Applications (AICCSA), 14., 2017. **Proceedings...** Hammamet: 2017, IEEE. p. 238-243.

CHRISTEL, M. G.; KANG, K. C. **Issues in Requirements Elicitation**. 1992. Disponível em: <https://resources.sei.cmu.edu/asset_files/TechnicalReport/1992_005_001_16478.pdf>. Acesso em: jan. 2018.

CHUNG, S. Object-Oriented Programming with DevOps. In: Annual Conference on Information Technology Education - SIGITE, 18., 2017, New York. **Proceedings...** New York: ACM Press, 2017. p. 65-65.

CORREIA, A.; ABREU, F. B. Adding preciseness to BPMN models. In: ENTERPRISE INFORMATION SYSTEMS CONFERENCE: ALIGNING TECHNOLOGY, ORGANIZATIONS AND PEOPLE, 4., 2012, Algarve. **Proceedings…** Algarve: Elsevier, 2012. p. 407-417.

DE NICOLA, A.; MISSIKOFF, M. A lightweight methodology for rapid ontology engineering. Communications of the ACM, v. 59, n. 3, p. 79–86, 25 fev. 2016.

DERMEVAL, D.; VILELA J.; BITTENCOURT, I. I.; CASTRO, J.; ISOTANI, S.; BRITO, P.; SILVA, A. Applications of ontologies in Requirements Engineering: a systematic review of the literature. **Requirements Engineering**, London, v.21, n.04, p.405-437, jan. 2015.

DI MARTINO, B.; ESPOSITO, A.; NACCHIA, E.; MAISTO, S., A. Semantic annotation of BPMN: current approaches and new methodologies. In: INTERNATIONAL CONFERENCE ON INFORMATION INTEGRATION AND WEB-BASED APPLICATIONS AND SERVICES, 17., 2015, Brussels. **Proceedings…** New York: ACM, 2015. p. 1-14.

DITORO, L. **Creating Value With BPMN**: A Business Users Perspective. 2017. Business Process Managament Institute BPMI. Disponível em: <http://www.bpminstitute.org/resources/articles/creating-

value-bpmn-business-users-perspective>. Acesso em: abr. 2018.

DITTRICH, Y.; NØRBJERG, TELL, P.; BENDIX, L. Researching Cooperation and Communication in Continuous Software Engineering. In: International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 11., 2018, Gothenburg. **Proceedings...** Gothenburg: IEEE, 2018. p.87-90.

EUZENAT, J.; SHVAIKO, P. **Ontology Matching**. Springer: Heidelberg, 2013. 512p.

FERNÁNDEZ, D. M.; WAGNER, S.; KALINOWSKI, M.; FELDERER M.; MAFRA, P.; VETRO A.; CONTE T.; CHRISTIANSSON T. M.; GREER, D.; LASSENIUS, C.; MÄNNISTÖ, T.; NAYABI, M.; OIVO, M.; PEZENSTADLER, B.; PFAHL, D.; PRIKLADINICK, R.; RUHE, G.; SCHEKELMANN, A.; SEN, S.; SPINOLA, R.; TUZCU, A.; DE LA VARA, J. L.; WIERINGA, R. Naming the pain in requirements engineering: Contemporary problems, causes, and effects in practice. **Empirical Software Engineering – An International Journal**, v. 22, n. 5, p. 2298-2338. Disponível em: <https://arxiv.org/abs/1611.10288>. Acesso em fev. 2018.

FIGUEIREDO, L. R. **Mapeamento de modelos de processos de negócio para ontologias, incluindo sistema de consultas**. 2018. 140f. Dissertação (Mestrado em Ciência da Computação) Universidade Estadual Paulista, Rio Claro, 2018.

FIGL, K. Comprehension of Procedural Visual Business Process Models: A Literature Review. **Business & Information Systems Engineering**. v. 59, p. 41-67, 2017.

GRIMM, S.; HITZLER, P.; ABECKER, A. Knowledge Representation and Ontologies. In: STUDER, R.; GRIMM, S.; ABECKER, A. (Ed.) **Semantic Web Services**. Berlin, Heidelberg: Springer, 2007. p. 52-105.

GUARINO, N. Formal ontology and Information Systems. In: FORMAL ONTOLOGY IN INFORMATION SYSTEMS. 1. 1998. Trento. **Proceedings…** Amsterdam: IOS Press, 1998. p.3-15.

HAISJACKL, C.; SOFFER, P.; LIM, S. Y.; WEBER, B. How do humans inspect BPMN models: an exploratory study. Software and System Modeling Journal. v. 16, p. [s.i], out. 2016. Disponível em <https://link.springer.com/article/10.1007/ s10270-016-0563-8>. Acesso em: mar. 2018.

HALLER, A.; MARMOLOWSKI, M.; OREN, E.; GAALOU, W. **A Process Ontology for Business Intelligence**. Galway, Ireland. DERI – Digital Enterprise Research Institute. 01 abr. 2008.

HEREDIA, L. R. **Transformação de Modelos de Processos de Negócio em BPMN para Modelos de Sistema utilizando Casos de Uso da UML**. 2012. 126f. Dissertação (Mestrado em Ciência da Computação) – Faculdade de Informática, Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2012.

HOPPE, T.; TOLKSDORF, R. Guide for Pragmatical Modelling of Ontologies in Corporate Settings. In: HOPPE, T.; REIBOLD, A.; HUMM, B. (ed.). **Semantic Applications**: Methodology, Technology, Corporate Use. Springer Verlag, 2018. p. 13-30.

INAYAT, I.; MORAES, L.; DANEVA, M.; SALIM, S.S. A Reflection on Agile Requirements Engineering: Solutions Brought and Challenges Posed. In: INTENATIONAL CONFERENCE ON AGILE SOFTWARE DEVELOPMENT. 16., 2015. Helsinki. **Proceedings…** New York: ACM, 2015. p.6.1-6.7.

IEEE - INSTITUTE OF ELECTRICAL AND ELECTRONIC ENGINEERS COMPUTER SOCIETY. **SWEBOK v3.0 - Guide to Software Engineering Body of Knowledge**. [s.l.] IEEE Computer Society, 2014.

ISO - INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO/IEC/IEEE/29148:2018. Systems and Software Engineering - Life cycle processes - Requirements Engineering**. ISO/IEC/IEEE, 2018.

ISOTANI, S.; BITTENCOURT, I. I. Ontologias e Representação do Conhecimento. In: _____. **Dados Abertos Conectados**. Núcleo de Informação e Coordenação do Ponto Br. 2015. Disponível em: <http://ceweb.br/livros/dados-abertos-conectados/capitulo-3/>. Acesso em: abr. 2018.

KLUZA K.; JOBCZYK, K.; WISNIEWSKI, P. A.; SUCHENIA, A. Comparison of Selected Modeling Notations for Process, Decision and System Modeling. In: FEDERATED CONFERENCE ON COMPUTER SCIENCE AND INFORMATION SYSTEMS. 11. 2017. Prague. **Proceedings…** Prague: IEEE, 2017. p.1095-1098.

KROGSTIE, J. **Quality in business process modeling**. Cham: Springer International Publishing Switzerland, 2016. 250p.

LIMA, A. S. **UML 2.3 - Do Requisito à Solução**. São Paulo: Editora Érica, 2011.

LESHOB, A. Towards a Business-Pattern Approach for UML Models Derivation from Business Process Models. In: IEEE INTERNATIONAL CONFERENCE ON E-BUSINESS ENGINEERING. 13., 2016. Macau. **Proceedings...** Macau: IEEE, 2016. p. 244-249.

MENDLING, J.; REIJERS, H. A.; VAN DER AALST, W. M. P. **Seven process modeling guidelines (7PMG)**. Information and Software Technology, v. 52, n. 2, p. 127-136. [S:l.]: Elsevier, 2010.

MIKSA, K.; SABINA, P.; FRIESEN, F.; RAHMANI, T.; LEMCKE, J.; WENDE, C.; SRDJAN, Z.; ABMANN, U.; BARTHO, A. Case Studies for Marrying Ontology and Software Technologies. In: PAN, J. Z.; STAAB, S.; ABMANN, U.; EBERT, J.; ZHAO, Y. (ed). **Ontology-Driven Software Development**. Springer Heidelberg, 2013. p. 69-94.

MIZOGUCHI, R. Tutorial on Ontological Engineering - Part 3: Advanced Course of Ontological Engineering. **New Generation Computing**, v. 22, n. 2, p. 198–220, 2004.

MOHAMED, K. A.; ELLATIF, M. A.; FARHAN, M.S. Using ontology-based concept maps for requirements engineering: A case study. In: INTERNATION COMPUTER ENGINEERING CONFERENCE. 13., 2017, Cairo. **Proceedings…** Cairo: IEEE Computer Society, 2017, p. 366-371.

NOGUEIRA, F. A. **Levantamento e Especificação de Requisitos de Software utilizando Modelos de Processos de Negócio**. 2017. 139f. Dissertação (Mestrado em Ciência da Computação) Universidade Estadual Paulista, Rio Claro, 2017.

NGUYEN, T. T. A.; CONRAD, S. Ontology Matching using multiple similarity measures. In: INTERNATIONAL JOINT CONFERENCE ON KNOWLEDGE DISCOVERY, KNOWLEDGE ENGINEERING AND KNOWLEDGE MANAGEMENT. 7., 2015, Lisbon. **Proceedings…** Lisbon: IEEE Computer Society, 2015, p.603-611.

OCA, M. I.; SNOECK, M. **Pragmatic guidelines for business process modeling**. Technical report KU Leuven, 2014. 70p.

ODEH, Y. BPMN in Engineering Software Requirements: An Introdutory Brief Guide. In: INTERNATIONAL CONFERENCE ON INFORMATION MANAGEMENT AND ENGINEERING. 9. 2017. Nova Iorque. **Proceedings…** New York: ACM, 2017. p.11-16.

OMG – OBJECT MANAGEMENT GROUP. **Business Process Model and Notation**. Object Management Group, 2013. Disponível em <http://www.omg.org/spec/BPMN/2.0.2/PDF>. Acesso em: jan. 2018.

_____. **Model Driven Architecture (MDA) MDA Guide rev. 2.0**. Object Management Group, 2014. Disponível em <http://www.omg.org/cgi-bin/doc?ormsc/14-06-01.pdf>. Acesso em: set. 2018.

_____. **OMG Unified Modeling Language (OMG UML) Version 2.5**. Object Management Group, 2015. Disponível em <http://www.omg.org/spec/UML/2.5/PDF>. Acesso em: jan. 2018.

PARK, G.; FELLIR, F.; HONG, J.; GARRIDO, J. L.; NOGUERA, M.; CHUNG, L. Deriving Use Cases from Business Processes: A Goal-Oriented Transformational Approach. In: SYPOSIUM ON APPLIED COMPUTING, 32., Marrakech, Morocco. **Proceedings...** New York: ACM, 2017. p. 1288-1295.

PENZENSTADLER, B.; FERNANDEZ, D.M.; RICHARDSON, D.; CALLELE, D. KRZYSTOF, W. The requirements engineering body of knowledge (REBoK). In: IEEE International Requirements

Engineering Conference (RE), 2013, 21. Rio de Janeiro. **Proceedings...** Rio de Janeiro: IEEE Computer Society, 2013., jul. 2013

RHAZALI, Y.; HADI, Y.; MOULOUDI, A. Transformation Method CIM to PIM: From Business Process Models Defined in BPMN to Use Case and Class Models Defined in UML. **International Journal of Computer and Information Engineering**. v. 8, n. 8, p. 1467-1471, 2014.

SAITO, S.; IIMURA, Y.; AOYAMA, M. REO: Requirements Engineering Ontology – Spectrum Analysis of Requirements Engineering Knowledge and its Practical Application. In: ANNUAL INTERNATIONAL COMPUTERS, SOFTWARE & APPLICATIONS CONFERENCE. 39., 2015, Taichung. **Proceedings…** Taichung: IEEE Computer Society, 2015, p. 62-70.

SOMMERVILLE, I. Processos de software. In: **Engenharia de Software**. 9. ed. Rio de Janeiro: Pearson Prentice Hall, 2011. p. 529.

_____. Engenharia de Requisitos. In: **Engenharia de Software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011. p. 529.

SOWA, J. F. **Building, Sharing and Merging Ontologies**. 2009. Disponível em: <http://www.jfsowa.com/ontology/ontoshar.htm> Acesso em: abr. 2018.

SHUNXIN, L.; LEIJUN, S. Requirements Engineering Based on Domain Ontology. In: INTERNATIONAL CONFERENCE OF INFORMATION SCIENCE AND MANAGEMENT ENGINEERING, 1, 2010, Xi'an. **Proceedings…** Xi'an: IEEE Computer Society, 2010. Vol 01. p.120-122.

TERNAI, K.; TÖRÖK, M.; VARGA, K. Corporate Semantic Business Process Management. In: GÁBOR, A.; KŐ, A. **Corporate Knowledge Discovery and Organizational Learning**: The Role, Importance, and Application of Semantic Business Process Management. Springer International Publishing, 2016. p. 33-57. (Knowledge Management and Organizational Learning, v. 2).

TSAGKANI, C.; TSALGATIDOU, A. Abstracting BPMN Models. In: Panhellenic Conference on Informatics 2015. 19. 2015. Athens. **Proceedings…** New York: ACM, 2015. p.243-244.

VAN DER AALST, W. M. P. **Business Process Management: a comprehensive survey**. ISRN Software Engineering. 2013. Disponível em: <http://dx.doi.org/10.1155/2013/507984>. Acesso em: out. 2017.

WfMC - WORKFLOW MANAGEMENT COALITION. **Process Definition Interface**: XML Process Definition Language. WfMC-TC-1025. Version 2.2. 2012. Disponível em: <http://www.xpdl.org>. Acesso em: mar. 2018.

W3C – WORLD WIDE WEB CONSORTIUM. **OWL 2 Ontology Language Primer (Second Edition)**. 2012. Disponível em: <https://www.w3.org/2012/pdf/REC-owl2-primer-20121211.pdf>. Acesso em: abr. 2018.

_____. **SPARQL 1.1 Overview**. 2013. Disponível em: <https://www.w3.org/TR/sparql11-overview/>. Acesso em: mai. 2018.

APÊNDICE A - PRINCIPAIS ELEMENTOS DA NOTAÇÃO BPMN

As **Tabelas 16** a **21** mostram os elementos da notação BPMN v2.0, de acordo com a especificação do consórcio OMG (2013), agrupados nas categorias: atividades, eventos, *gateways*, estruturas básicas, artefatos e subprocessos.

Tabela 16 - Atividades e seus respectivos tipos.

ATIVIDADES	DESCRIÇÃO
	**Simples:** tipo genérico de atividade, empregado normalmente nos estágios iniciais do desenvolvimento do processo, sem que haja a necessidade de discriminação em detalhes.
	**Serviço:** atividade associada a algum tipo de serviço ou sistema, e que não necessita de interferência humana.
	**Usuário:** atividade realizada por uma pessoa auxiliada por um sistema.
	**Recebimento de mensagem:** atividade alimentada por participantes externos.
	**Envio de mensagem:** atividade relacionada ao envio de mensagem a um participante externo.
	***Script*:** atividade executada por meio de um mecanismo de processos de negócio. A definição do *script* ocorre a partir de um modelador, em uma linguagem que possa ser interpretada pelo motor de processos.
	**Manual:** atividade exclusivamente realizada por humanos.
	**Regra de negócio:** atividade que fornece um mecanismo para que o processo defina informações para a execução de um motor de regras de negócio para a obtenção dos cálculos realizados por esse motor de regras de negócio.

Fonte: Figueiredo (2018), OMG (2013).

Tabela 17 - Eventos e seus respectivos tipos.

EVENTOS DE INÍCIO	EVENTOS INTERMEDIÁRIOS	EVENTOS DE FIM	DESCRIÇÃO
			**Nenhum:** elemento que representa o início, a continuidade ou o fim de um processo.
			**Mensagem:** Indica o recebimento ou envio de uma mensagem
			**Tempo:** indica uma parada definida por um tempo determinado para iniciar/continuar o processo
			**Condição:** elemento que determina o aguardo de uma condição ou expressão até que a mesma seja verdadeira para dar início/continuidade ao processo.
			**Sinal:** demonstra que em determinado ponto do fluxo haverá o recebimento ou o envio de um sinal.
			**Múltiplo:** determina a existência de diversos eventos que disparam ou continuam o processo. Todavia, somente um é necessário. Para o caso de evento de fim, indica a existência de sequências ao finalizar o processo, sendo que todas ocorrerão.
			**Múltiplo paralelo:** indica que existem vários eventos e que todos precisam ocorrer em paralelo para que se inicie ou continue o processo.
			***Link*:** interliga atividades de um mesmo processo, indicando recebimento ou envio do *link*, deixando o diagrama mais limpo.
			**Compensação:** indica a necessidade de uma compensação, ou seja, alguma atividade que será desfeita no processo.
			**Escalar:** indica a necessidade de escalabilidade de um processo, comunicando uma falha de execução de alguma atividade do subprocesso ao seu processo pai.
			**Exceção:** denota a criação de um erro no fim de um processo.
			**Cancelamento:** é utilizado dentro de um subprocesso de transação, indicando que a mesma deve ser cancelada. Efetua o disparo de um evento intermediário, receptor de cancelamento, na fronteira do subprocesso.

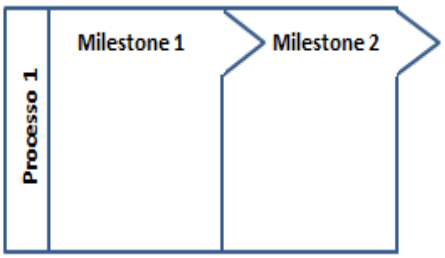
Fonte: Figueiredo (2018), OMG (2013).

Tabela 18 - *Gateways* e seus respectivos tipos.

GATEWAYS	DESCRIÇÃO
	***Gateway* exclusivo:** é utilizado para a situação em que existe apenas um caminho a ser tomado após uma decisão. Pode ser usado como ponto de convergência ou divergência.
	***Gateway* paralelo:** é utilizado para indicar que todos os caminhos devem ser seguidos simultaneamente, sem que haja a necessidade de se tomar uma decisão.
	***Gateway* inclusivo:** utilizado quando há a possibilidade do seguimento de vários caminhos para uma decisão ser tomada.
	***Gateway* baseado em eventos:** só existe um caminho a ser seguido, e isso depende do evento intermediário que ocorrer primeiro.
	***Gateway* exclusivo baseado em eventos:** semelhante ao *gateway* baseado em eventos, mas nesse caso cada ocorrência de um evento posterior inicia uma nova instância do processo.
	***Gateway* paralelo baseado em eventos:** semelhante ao *gateway* baseado em eventos, porém só há o seguimento do fluxo quando ocorre um evento.
	***Gateway* complexo:** utilizado para decisões com base em expressões complexas e situações que os outros gateways não conseguem contemplar.

Fonte: Figueiredo (2018), OMG (2013).

Tabela 19 - Estruturas básicas da notação BPMN.

ESTRUTURAS BÁSICAS	DESCRIÇÃO
	**Piscina ou *Pool*:** utilizado para representação de um ator em um processo.
	**Raia ou *Lane*:** subdivisão de uma piscina, empregada para organizar e categorizar os participantes e suas atividades dentro de um processo.
	**Etapa ou *Milestone*:** utilizado para indicar as diferentes etapas de um processo, exibindo assim uma mudança de fase.

Fonte: Figueiredo (2018), OMG (2013).

Tabela 20 - Artefatos em BPMN.

ARTEFATOS	DESCRIÇÃO
	**Objeto de dados:** disponibiliza informações a respeito de documentos, dados e outros objetos. É usado e atualizado conforme necessidade durante o processo.
	**Depósito de dados:** fornece um mecanismo de resgate ou atualização de dados e/ou informações que serão persistidos além do escopo do processo.
	**Grupo:** agrupamento de elementos do diagrama que não afeta o mesmo. É empregado para documentação ou análise.
	**Anotação:** possibilita a inserção de informações adicionais no diagrama, com o objetivo de facilitar a leitura do diagrama pelo usuário.

Fonte: Figueiredo (2018), OMG (2013).

Tabela 21 - Subprocessos e seus respectivos tipos.

SUBPROCESSOS	DESCRIÇÃO
	**Subprocesso:** empregado em uma atividade que contém um conjunto de outras atividades. Os subprocessos são dependentes de um processo, mas possuem fluxo próprio.
	**Subprocesso reutilizável:** realiza o encapsulamento da chamada a um processo global dentro do processo de negócio. A ativação desse subprocesso transfere o controle processual para o processo global.
	**Transacional:** o comportamento do subprocesso é controlado por meio de um protocolo de transação. Permite que todas as suas atividades sejam concluídas com sucesso ou canceladas e compensadas.
	***Ad hoc*:** subprocesso que é formado por um grupo de atividades que não têm obrigatoriamente relações sequenciais. Esse subprocesso é concluído após a execução de todas as atividades.
	***Loop*:** indica a repetição de um subprocesso até que se atinja a condição estabelecida anteriormente. Nesse caso não se sabe o número de vezes em que será repetido o processo.
 (Paralelo e Sequencial, respectivamente)	**Múltiplas instâncias:** os subprocessos podem ser repetidos de maneira paralela ou sequencial. A quantidade de vezes em que esses subprocessos serão repetidos é conhecida antes do início dos mesmos.

Fonte: Figueiredo (2018), OMG (2013).

APÊNDICE B - LINGUAGEM SPARQL E FRAMEWORK APACHE JENA

Como citado na *Subseção 5.5*, o *framework* Apach Jena (*https://jena.apache.org*) foi utilizado no desenvolvimento do sistema OnToSRSR com o objetivo de realizar consultas em SPARQL. Na **Figura 56** é mostrado um exemplo de código-fonte para carregar um arquivo OWL. N **Figura 57** é mostrado um código-fonte em que é criada uma consulta que recupera uma lista de classes *act-User* e suas respectivas propriedades e relações.

Figura 56 - Carregando um arquivo OWL com Apache Jena.

```
1  String SOURCE =
   "http://www.semanticweb.org/ontologiaDeProcesso/Contas_a_pagar";
2  String NS = SOURCE + "#";
3  OntModel ont = ModelFactory.createOntologyModel(OntModelSpec.OWL_DL_MEM);
4  ont.read(FileManager.get().open("Contas_a_pagar.owl", "RDF/XML");
```

Fonte: elaborado pelo autor.

Parte da saída do código mostrado na **Figura 57** é mostrada na **Figura 58**, em que são evidenciadas as propriedades *id*, *name*, *description* e *documentation*. Em *Type* e *SubType*, que representam respectivamente o tipo e o subtipo da classe também são mostrados (axioma *subClassOf*).

Mais informações sobre o framework Apache Jena podem ser obtidas no site oficial: https://jena.apache.org/.

Figura 57 - Código para execução de uma consulta simples com Apache Jena.

```
01  StringBuilder queryStr = new StringBuilder("PREFIX rdf:
    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> ");
02  queryStr.append("PREFIX owl: <http://www.w3.org/2002/07/owl#> ");
03  queryStr.append("PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> ");
04  queryStr.append("PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> ");
05  queryStr.append("PREFIX j.0: <").append(NS).append("> ");
06  queryStr.append("SELECT ?id ?name ?description ?documentation ?type ?sub_type ");
07  queryStr.append("WHERE { ");
08  queryStr.append("  ?x  rdfs:subClassOf  j.0:actv-User ");
09  queryStr.append("  BIND(\"actv-User\" AS ?sub_type) ");
10  queryStr.append("  j.0:actv-User ");
11  queryStr.append("  rdfs:subClassOf ?type . ");
12  queryStr.append("  ?x j.0:i1-id ?id ;");
13  queryStr.append("  j.0:i2-name  ?name ;");
14  queryStr.append("  j.0:i3-description    ?description ;");
15  queryStr.append("  j.0:i4-documentation  ?documentation");
16  queryStr.append("}");
17  //..instancia o objeto Query para ser passado ao objeto QueryExecutionFactory
18  Query query = QueryFactory.create(queryStr.toString());
19  //..executa
20  try (QueryExecution queryExec = QueryExecutionFactory.create(query, ont)) {
21      ResultSet rs = queryExec.execSelect();
22      while (rs.hasNext()) { //..enquanto houver elementos no resulSet, faça...
23          QuerySolution sol = rs.nextSolution();
24          Literal id, name, description, documentation;
25          String type = null; String subtype = null;
26          //..pega os dados da consulta e atribui a variáveis
27          id = sol.getLiteral("id");
28          name = sol.getLiteral("name");
29          description = sol.getLiteral("description");
30          documentation = sol.getLiteral("documentation");
31          if(sol.getResource("type") != null)
32              type = sol.getResource("type").getLocalName();
33          RDFNode subType = sol.get("sub_type");
34          //..mostra o resultado na saída do programa.
35          System.out.println("---------------------------------------\n");
36          System.out.println("Id: " + id + "\n");
37          System.out.println("Name: " + name + "\n");
38          System.out.println("Description: " + description + "\n");
39          System.out.println("Documentation: " + documentation + "\n");
40          System.out.println("Type: " + type + "\n");
41          System.out.println("Subtype: " + subType.toString() + "\n");
42          }
43      } catch (Exception ex) {
44              System.out.println("Erro na consulta: \n" + ex.getMessage());
45      }
```

Fonte: elaborado pelo autor.

Figura 58 - Saída da execução do código mostrado na Figura 57.

```
Id: d50d98b6-fd37-4904-943f-548c0ed7fd64

Name: Check Invoice Products/Services

Description: The financial assistant must check the prod

Documentation: The financial assistant must check the pr

Type: Activity

Subtype: actv-User

----------------------------------------

Id: 7e032e15-06c3-4313-a9aa-3d92332d8302

Name: Justify  the rejection

Description: indicate the reason for rejection

Documentation: indicate the reason for rejection

Type: Activity

Subtype: actv-User
```

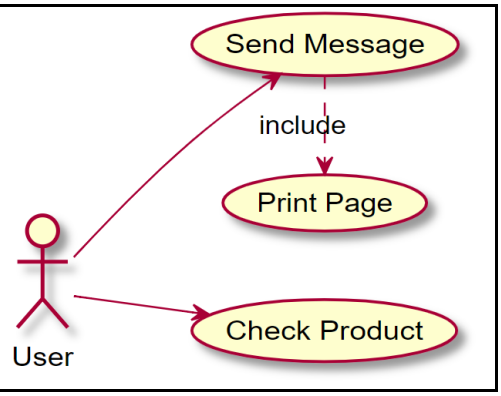
Fonte: elaborado pelo autor.

APÊNDICE C – FERRAMENTA PLANTUML

Como mostrado na Seção 5.5, a ferramenta PlantUML (*htttp://www.plantuml.com*) é uma ferramenta usada para a criação de diagramas UML por meio de uma linguagem descritiva de alto nível. Na **Figura 59** é mostrado um exemplo de código em PlantUML para a definição de um simples diagrama de casos de uso. Na linha 2 é conFigurada a direção que o diagrama irá fluir. A *string "@startuml"* (linha 1) define o início do diagrama, a *string "@enduml"* (linha 7) define o fim do diagrama. Entre essas duas declarações, são definidos os atores (linha 3) usando um nome simples. Os relacionamentos entre atores e casos de uso são definidos unindo as *strings* desses dois elementos usando um ou mais hifens seguido de um sinal de maior ou menor, dependendo do sentido do relacionamento (linhas 4 e 5). Os relacionamentos entre os casos de uso (linha 6), do tipo *include* ou *extend*, é realizado usando um ponto seguido de um sinal de maior ou menor, dependendo da direção, entre os nomes dos casos de uso.

Figura 59 - Exemplo de diagrama de casos de uso genérico usando a linguagem PlantUML.

1	**@startuml**
2	left to right direction
3	actor User
4	User --> (Send Message)
5	User --> (Check Product)
6	(Send Message) .> (Print Page) : include
7	**@enduml**

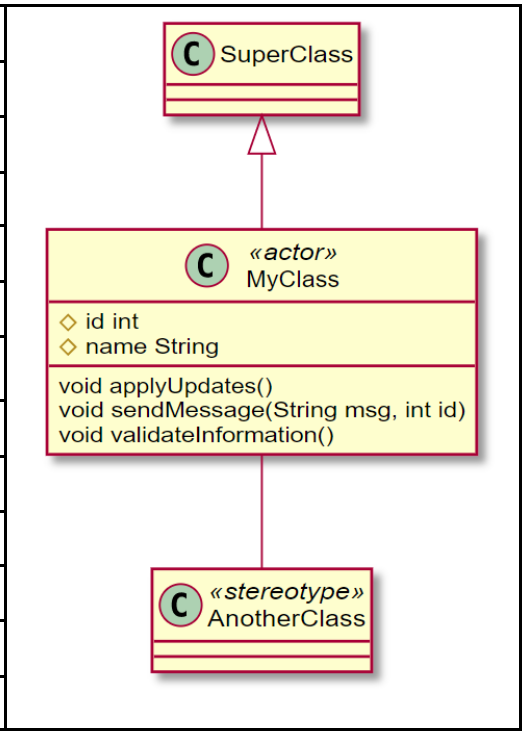


Fonte: elaborado pelo autor.

Na **Figura 60** é mostrado um exemplo de diagrama de classes com duas classes relacionadas. A declaração das classes e seus elementos lembra uma linguagem de programação orientada a objetos, como Java ou PHP. Na linha 2 há a declaração de uma classe simples, sem atributos ou métodos. Da linha 3 até a linha 9 é declarada uma classe com estereótipo e seus atributos e métodos. Na linha 10 há a declaração de outra classe. O relacionamento de herança entre as duas primeiras classes é realizado unindo os respectivos nomes usando sinal de maior, barra vertical e hífen (linha 11). Uma associação simples entre classes é mostrada na linha 12.

Figura 60 - Exemplo de diagrama de classes genérico usando a linguagem PlantUML.

1	**@startuml**
2	class SuperClass
3	class MyClass << actor >> {
4	#id int
5	#name String
6	{method} void applyUpdates()
7	{method} void sendMessage(String msg, int id)
8	{method} void validateInformation()
9	}
10	class AnotherClass <<stereotype>>
11	SuperClass <|-- MyClass
12	MyClass -- AnotherClass
13	**@enduml**



Fonte: elaborado pelo autor.

APÊNDICE D – SISTEMA ONTOSRS: USO E CÓDIGO *(CD-R anexo)*

Este Apêndice contém o código fonte em Java da ferramenta OnToSRS, desenvolvida para automatizar o processo sistemático apresentado. Além disso, este Apêndice contém instruções para a conFiguração de todos os recursos necessários para execução da ferramenta.

O conteúdo deste Apêndice encontra-se em formato digital, em CD-R anexo a esta dissertação, em pasta denominada “APENDICE D – Sistema ONTOSRS”.

APÊNDICE E – DOCUMENTO ERS DO MODELO "ACCESS MANAGEMENT" (*CD-R anexo)*

Este Apêndice contém o documento ERS referente ao modelo de processos de negócio intitulado "Accounts Payable" gerado pelo estudo e caso apresentado na *Seção 6.1*. Ademais, também foram disponibilizados o arquivo BPMN, criado e editado com a ferramenta de modelagem Bizagi Modeler, e os arquivos OWL gerados pelo sistema PM2ONTO v2.0.

O conteúdo deste Apêndice encontra-se em formato digital, em CD-R anexo a esta dissertação, em uma pasta denominada "APÊNDICE E – Estudo de caso 01".

.

APÊNDICE F – DOCUMENTO ERS DO MODELO "ACCOUNTS PAYABLE" (*CD-R anexo)*

Este Apêndice contém o documento ERS referente ao modelo de processos de negócio intitulado "Accounts Payable" gerado pelo estudo e caso apresentado na *Seção 6.2*. Ademais, também foram disponibilizados o arquivo BPMN, criado e editado com a ferramenta de modelagem Bizagi Modeler, e os arquivos OWL gerados pelo sistema PM2ONTO v2.0.

O conteúdo deste Apêndice encontra-se em formato digital, em CD-R anexo a esta dissertação, em uma pasta denominada "APÊNDICE F – Estudo de caso 02".

APÊNDICE G – DOCUMENTO ERS DO MODELO "ESTÁGIO FATEC JALES" (*CD-R anexo)*

Este Apêndice contém o documento ERS referente ao modelo de processos de negócio intitulado "Estágio Fatec Jales" gerado pelo estudo e caso apresentado na *Seção 6.3*. Ademais, também foram disponibilizados o arquivo BPMN, criado e editado com a ferramenta de modelagem Bizagi Modeler, e os arquivos OWL gerados pelo sistema PM2ONTO v2.0.

O conteúdo deste Apêndice encontra-se em formato digital, em CD-R anexo a esta dissertação, em uma pasta denominada "APÊNDICE G – Estudo de caso 03".

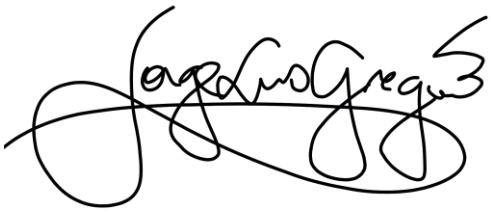
APÊNDICE H – DOCUMENTO ERS DO MODELO "ESTÁGIO FATEC JALES CADASTRO" (*CD-R anexo)*

Este Apêndice contém o documento ERS referente ao modelo de processos de negócio intitulado "Estágio Fatec Jales", após ser realizada a operação de *merging*, gerado pelo estudo e caso apresentado na Seção 6.4. Ademais, também foram disponibilizados o arquivo BPMN, criado e editado com a ferramenta de modelagem Bizagi Modeler, e o arquivo OWL gerados pelo sistema PM2ONTO v2.0.

O conteúdo deste Apêndice encontra-se em formato digital, em CD-R anexo a esta dissertação, em uma pasta denominada "APÊNDICE H – Estudo de caso 04".

Autorizo a reprodução xerográfica para fins de pesquisa

Rio Claro, 05/09/2019



_____

Assinatura