Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Motor de Workflow Semântico para Processos de Negócios Não Estruturados

Gabriel Lins e Nóbrega

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Edison Ishikawa

Brasília
2021

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

Nóbrega, Gabriel
NG118mm Motor de Workflow Semântico para Processos de Negócios Não Estruturados / Gabriel Nóbrega; orientador Edison Ishikawa. - Brasília, 2021.
60 p.

Monografia (Graduação - Ciência da Computação) -- Universidade de Brasília, 2021.

1. BPMN. 2. Processos Não Estruturados. 3. Modelagem de Processos. 4. Ontologia. 5. Anotação Semântica. I. Ishikawa, Edison, orient. II. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Motor de Workflow Semântico para Processos de Negócios Não Estruturados

Gabriel Lins e Nóbrega

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Edison Ishikawa (Orientador)
CIC/UnB

Prof. Dr.a Vanessa Tavares Nunes
CIC/UnB

Prof. Dr. Fernando Chacon de Albuquerque
CIC/UnB

Prof. Dr. Marcelo Grandi Mandelli
Coordenador do Bacharelado em Ciência da Computação

Brasília, 16 de novembro de 2021

Dedicatória

Deus quer,
O homem sonha
A obra nasce.
- Fernando Pessoa

Dedico essa obra à minha família. Sem eles, nada disso seria possível. Por tudo, mutíssimo obrigado.

Agradecimentos

Primeiramente à minha família: pelos momentos de amor, compreensão, reflexão e apoio, muito obrigado!

Aos meus amigos: pelas risadas que nunca acabam, pelos momentos de catarse, pelas reflexões e discussões absolutamente inúteis que tanto gostamos, e pela amizade sempre presente, obrigado.

À minha namorada: pelo companheirismo inabalável, obrigado por estar sempre ao meu lado. Você é a luz no fim do túnel, meu porto seguro.

Ao meu orientador, Prof. Edison Ishikawa, pelas inúmeras oportunidades e diversos trabalhos realizados ao longo da graduação, meu eterno agradecimento. Você moldou meu futuro mais do que qualquer outro professor, e me fez um homem melhor por isso. Que os próximos estudantes tenham a sorte de trabalhar com você como eu tive, e que futuros professores sigam seus passos.

Por fim, gostaria de agradecer a Universidade de Brasília. Por ser um local em que excelência é o esperado e não a exceção, obrigado. Por prover um ambiente seguro para discutir, aprender, conviver, e viver, obrigado. Por lutar pelo ensino, pela educação e pelo corpo docente, muito obrigado.

Como sabemos, é pelo ensino que se liberta.

Este trabalho foi patrocinado pela Fundação de Apoio à Pesquisa do Distrito Federal (FAP-DF), no âmbito do projeto "Sistemas de informações organizacionais flexíveis baseados em processos de negócios com orientação semântica contextual"; outorga SEI 00193-00000096 / 2019-78.

Apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

O presente trabalho de conclusão de curso tem como objetivo analisar, investigar, contribuir e implementar um motor de workflow semântico capaz de modelar processos de negócio não estruturados, e, a partir do auxilio de inferências semânticas, melhorar a corretude, extensibilidade e manutenibilidade de tais processos de negócio. Para isso, a metodologia foi separada nas seguintes: pesquisa bibliográfica e fundamentação teórica, implementação do software em questão, e inspeção e validação manual dos artefatos produzidos.

Para isso, o motor semântico contará com informações, descrições e relacionamentos relacionados ao domínio do processo, bem como do próprio domínio de processos de negócios. A partir disso, ele será capaz de inferir quais atividades devem ser executadas por quais atores, além de definir possíveis fluxos e modificar atributos de atividades, buscando evitar situações-problema frequentemente encontradas no contorno de domínios e permitir maior grau de autonomia aos usuários do sistema em questão.

Palavras-chave: BPMN, processos não estruturados, modelagem de processos, ontologia, semântica, trabalho de conclusão de curso

Abstract

The present work, in the form of a graduation thesis, seeks to analise, investigate, contribute and implement a semantic workflow engine capable of modelling unstructured business processes and, using semantic inferences, improve said processes correctness, extensibility and maintenability. To achieve that, the metodology comprises of the following: related work and theory fundamentals, modelling and implementation, and manual inspection / validation of produced artifacts.

To do so, the semantic motor must rely on information, description and relationships associated to the process domain, aswell as of the business processes domain in itself. With those, it will be able to infer which activities must be executed by which actors, also defining possible execution flows and modifying activity attributes, avoiding problems frequently encountered on domain's boundaries, allowing greater autonomy for each of its users.

Keywords: BPMN, non-structured processes, process modelling, ontology, semantic, graduation thesis

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AIIM Association for Intelligent and Information Management.

API Application Programming Interface.

BDI Belief-Desire-Intention.

BPM Business Process Management.

BPMN Business Process Model Notation.

DSR Design Science Research.

IA Inteligência Artificial.

UBP Unstructured Business Process.

Capítulo 1

Introdução

Englobando o que deve ser feito e como deve ser feito, processos de negócio são frequentemente descritos como ativos centrais de organizações modernas [1]. Processos de negócio são, essencialmente, atividades e etapas necessárias a serem executadas para que determinado objetivo seja atingido. É um bloco fundamental para a gestão e automação de processos, e a formalização de processos garante que ele ocorra de forma sistemática, de forma indiferente ao agente executor e à ruídos externos.

Sendo assim, afetando diretamente o planejamento e a execução do funcionamento de uma empresa, é evidente que o sucesso ou o fracasso dessas depende de seus processos. Sendo assim, o planejamento e a execução de tais processos assumem papeis cruciais dentro de organizações modernas, e é onde o presente trabalho pretende atuar: desenvolvendo um sistema capaz de integrar processos de negócios e ontologias, melhorar a qualidade, a manutenção, a flexibilidade, e a extensibilidade de processos de negócio. Utilizando todo o conhecimento externo do domínio relacionado aos processos em questão, realizar inferências sobre as atividades a serem executadas, definindo atores e demais atributos associados (como, por exemplo, ordem e data de execução).

1.1 Definição do Problema

Embora sejam frequentemente utilizados na indústria e na academia - e, essencialmente, em todas as organizações modernas -, processos de negócios ainda contam com um vasto leque de problemas associados [2]. Entre eles, se destacam:

- Utilização humana;
- Flexibilidade;
- Manutenibilidade;

- Extensibilidade.

Na frente da utilização humana, a maioria dos problemas provém da falibilidade do indivíduo. Não é incomum que erros de compreensão ocorram, afetando o planejamento, ou erros de execução, como associação incorreta de processos. Ainda, ambiguidade nos rótulos de cada processo e atividade também são problemas que ainda não foram resolvidos, e, por envolver diretamente o processo de decisão humano, dificilmente serão.

Por outro lado processos dificilmente são flexíveis e extensíveis (edição e adição de atividades podem gerar problemas em diversos pontos no fluxo de execução) ou manuteníveis (se múltiplos processos contam com uma atividade que deve ser modificada, ela deve ser modificada em todos os processos), o presente trabalho estudará mecanismos para melhorar tais problemas. Para isso, será trabalhada principalmente uma classe de processos de negócio: processos de negócio não estruturados.

Sendo assim, a seguinte hipótese de pesquisa é formulada: é possível, a partir da implementação de um sistema de gestão de processos com motor semântico, auxiliar na execução e no planejamento de processos de negócio de forma a melhorar a corretude, extensibilidade, e manutenibilidade de tais processos? Para tal, ao motor semântico deverá ser garantido definições e descrições de regras, contextos, e todo o conhecimento semântico necessário tanto do domínio do processo, quanto do domínio de processos de negócio e seu funcionamento generalizado.

1.2 Objetivos

Nesta seção, serão descritos os objetivos gerais e específicos do trabalho sob a perspectiva de funcionalidades esperadas e de utilização do mesmo por usuários.

1.2.1 Objetivos Gerais

Entre os objetivos gerais do presente trabalho, se encontram:

1. Melhorar o desenho, o planejamento, e a utilização de processos;
2. Melhorar a corretude dos processos, reduzindo erros;
3. Permitir maior grau de autonomia aos usuários, de forma que eles possam focar no que de fato é importante para a organização a qual pertencem, e não em minúcias e em casos de contorno da utilização.

Com isso, espera-se ser capaz de responder, com acurácia, a hipótese proposta na seção anterior, bem como todos os problemas descritos anteriormente.

1.2.2 Objetivos Específicos

Como objetivos específicos, podemos citar a investigação, análise, contribuição e implementação de um motor de workflow semântico capaz de lidar com processos de negócio não estruturados, bem como uma ontologia apropriada para o estudo de caso a ser desenvolvido.

Ademais, podemos granularizar a melhoria de desenho e de planejamento sob a ótica da melhoria de extensibilidade e manutenibilidade. Para isso, precisamos mostrar que a solução proposta é capaz de, a partir de uma única alteração em uma definição semântica, modificar corretamente os processos de negócios e suas atividades já descritas, e que a edição de um conceito gera efeitos consistentes e confiáveis nas atividades já propostas.

Sobre a melhoria de execução de processos, espera-se que, ao deixar que o motor de inferência produza novas atividades (e que elas sejam semanticamente corretas), o grau de descrição do processo aumente. Com isso, interpretações pessoais dos atores (e não verificadas semanticamente) serão reduzidas ao mínimo, bem como erros desta origem. Ainda, na eventualidade de erros, a flexibilidade do processo (concedida pela natureza não estruturada destes) deve permitir sua redefinição para corrigir o erro.

Além disso, o fluxo de inferência semântica deve ocorrer automaticamente e apenas quando for próprio, de forma a evitar re-inferências repetitivas que não produziriam resultados diferentes das anteriores.

Por fim, para atingir os objetivos gerais e específicos, deverá ser realizada uma validação teórica e prática, com um estudo de caso suficientemente abrangente, contendo testes significativos, capazes de evidenciar a capacidade operacional do sistema proposto.

1.3 Organização do Trabalho

O restante do presente trabalho se organiza da seguinte forma:

- No capítulo 2, é apresentada a fundamentação teórica acerca dos assuntos abordados, apresentando conceitos necessários para o compreendimento integral do projeto proposto. Ainda, diversos trabalhos recentemente produzidos são apresentados;
- No capítulo 3 é discutida a metodologia a ser utilizada ao longo do presente trabalho, bem como a definição do estudo de caso e dos artefatos produzidos;
- No capítulo 4, a fim de criar um alicerce empírico, são apresentados trabalhos relacionados e motores de workflow frequentemente utilizados;
- No capítulo 5, apresenta-se a descrição da solução proposta, sob o ponto de vista de modelagem de *software*;

- No capítulo 6 são apresentados os artefatos produzidos, e discutidas questões relativas ao seu funcionamento;
- No capítulo 7, encerra-se o trabalho abrindo espaço para continuação do problema apresentado e para o aprimoramento da tecnologia desenvolvida, com propostas de continuação e melhorias diversas.

Capítulo 2

Fundamentação Teórica

Nesta seção, com o intuito de criar uma base teórica para o trabalho, serão apresentados conceitos-chave para o entendimento geral do sistema proposto, bem como trabalhos recentes relacionados ao problema abordado. Para isso, serão descritos conceitos relacionados à processos de negócios estruturados e não estruturados, além de definições associadas à ontologias e seus modelos de inferência, agentes, objetivos, desejos e intenções. Por fim, associando processos de negócio com o motor semântico, o conceito de modelagem semântica.

2.1 Processos de Negócio

Processos de negócio são a combinação de um conjunto de atividades, de uma mesma organização, cuja estrutura descreve uma dependência entre atividades, de forma que o conjunto tenha um mesmo objetivo em comum. [3] Assim, a modelagem de tais processos, bem como a análise deles, é capaz de descrever compreensivamente o funcionamento de organizações e de seus principais grupos de execução. Ainda, a formalização de processos garante que ele ocorra de forma sistemática, independente do ator que a executa e de suas peculiaridades individuais, de forma agnóstica também à ruídos externos.

Sendo assim, processos de negócio são, essencialmente, atividades e etapas necessárias a serem executadas para que determinado objetivo seja atingido. Naturalmente, se tornou uma parte integral do funcionamento de empresas modernas [2], fundamental para a gestão e automação de processos.

A gestão de processos de negócio (do inglês, Business Process Management (BPM)) é tido, atualmente, como um facilitador para que organizações mantenham sua vantagem competitiva [4], se tornando um excelente princípio para gestão de empresas e instituições. Em sua pesquisa, Hung (2006) examinou o BPM com base em dois construtos: alinhamento de processos e pessoas envolvidas (também frequentemente identificadas por

atores), testando empiricamente a relação entre a existência e a execução de processos e o desempenho organizacional. Os resultados demonstram que, naturalmente, órgãos que possuem o alinhamento processos-atores estão positivamente associados com o aumento de desempenho.

Portanto, ao englobar "o que deve ser feito"e "como deve ser feito", processos de negócio se tornam partes integrais e principais de organizações atuais, interferindo positivamente no planejamento e na execução do operacional de uma empresa.

2.1.1 Processos Estruturados

Refletindo, dentro da área de processos de negócio, processos com etapas facilmente predizíveis e frequentemente repetitivas, existem aqueles em que são classificados como "processos estruturados". Nele, as atividades nos processos possuem uma dependência lógica entre a ordem de execução de cada atividade, além da dependência de um mesmo objetivo final em comum.

Processos de negócio estruturados é uma área bem estudada na academia, e seus conceitos e conhecimentos são maduros. Para tal, observa-se a existência de um notação padrão já aceita na industria e na comunidade científica: Business Process Model Notation (BPMN) [5].

Tipicamente, devido ao fato de os atores já os utilizarem e concordarem com as atividades descritas, processos estruturados são processos já bem compreendidos pela organização em voga. Ainda, levando em consideração o requerimento lógico entre as etapas e atividades envolvidas, tais processos são comumente projetados "da forma que estarão", na visão do analista de negócio, e implementados "da forma que estão", *hardcoded* nos sistemas informacionais utilizados pela empresa. Portanto, a modificação destes (seja uma edição simples ou uma extensão complexa) é relativamente complicada, podendo se tornar demasiadamente laboriosa e inviável.

BPMN

BPMN, como descrito anteriormente, é uma notação padrão para descrever processos estruturados. De acordo com sua especificação [6], existem 5 grandes classes associadas à notação:

1. Piscinas, representando uma entidade ou um grupo;
2. Raias, representando uma entidade ou um subgrupo;
3. Eventos, representando um evento associado ao processo;
4. Atividades, representando uma atividade;

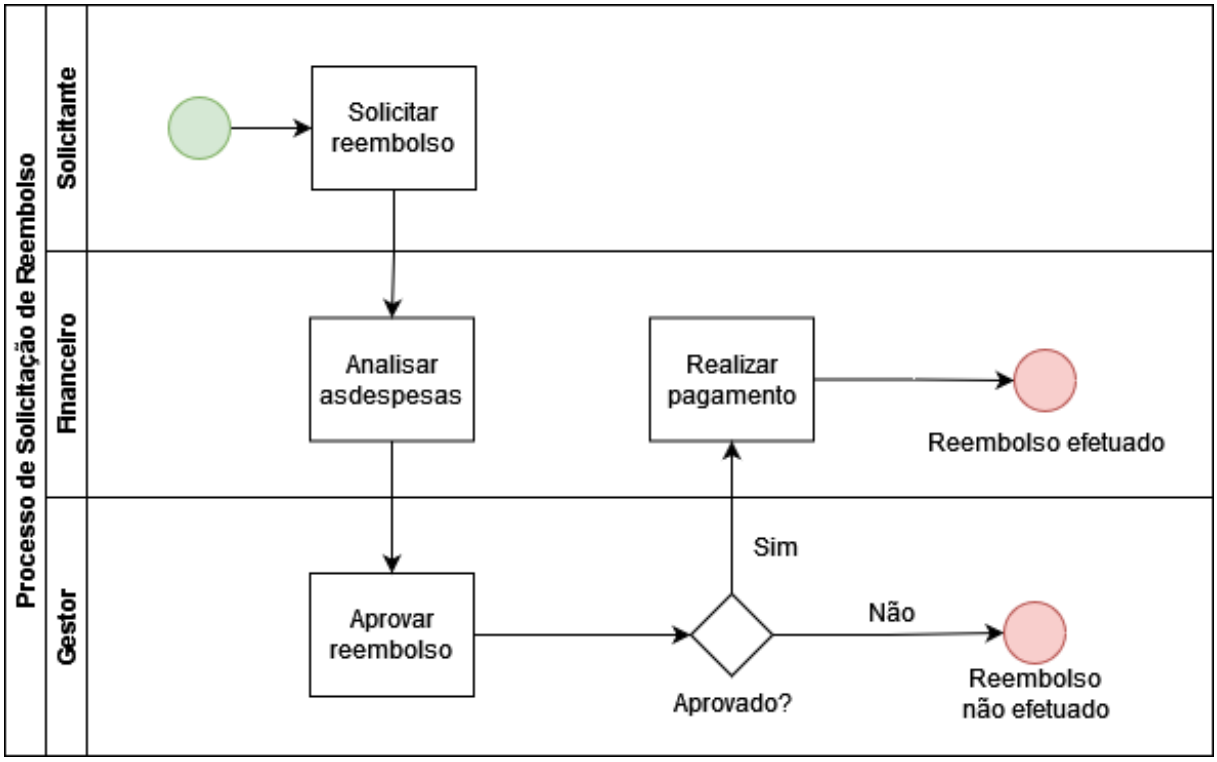

Figura 2.1: Exemplo de processo estruturado - fluxograma de requisiçaõ de reembolso.

5. Fluxos, representando um fluxo direcional entre objetos de um processo;

Para isso, implementações da notação tendem a abstrair tais classes de forma que possam se beneficiar de heranças múltiplas, facilitando a manutenção e a favorecendo a extensibilidade da implementação.

2.1.2 Processos Não Estruturados

Devido à natureza descritiva dos recursos necessários em um processo, dos agentes envolvidos, e dos procedimentos a serem realizados, existem processos cujas etapas podem ser descritas em tempo de projeto, ou *design-time*. Outros processos, porém, dependem estritamente de eventos ocorrendo em tempo real, cujo planejamento e descrição não podem ser previstos anteriormente [7], caracterizados por baixa previsibilidade e alta flexibilidade. A estes processos é dado o nome de processos não estruturados (do inglês, Unstructured Business Process (UBP)).

Ainda, é necessário notar que, nos últimos anos, o foco da indústria em processos não estruturados aumentou consideravelmente. De acordo com um relatório técnico da AIIM[8], publicado em 2014, em 51% das empresas entrevistadas, mais da metade de seus processos de negócio são classificados como não estruturados. Um exemplo deste

tipo de processo pode ser facilmente observado em soluções de problemas que envolvem alto conhecimento, como, no geral, suporte técnico de sistemas informacionais. Embora possível, listar todas as causas de todos os problemas em conjunto com suas possíveis soluções é um trabalho pouco factível que até hoje conta com soluções em tempo real auxiliadas por atores humanos.

Em processos deste tipo, uma forma de representação comum é a de nuvem [9]. Para representar processos não estruturados dessa maneira, é necessário que suas atividades estejam representadas em "*bags*", ou conjuntos únicos, em que as atividades nele contidas contenham uma mesma característica em comum. O conjunto dessas *bags* forma, em união, a representação em nuvem.

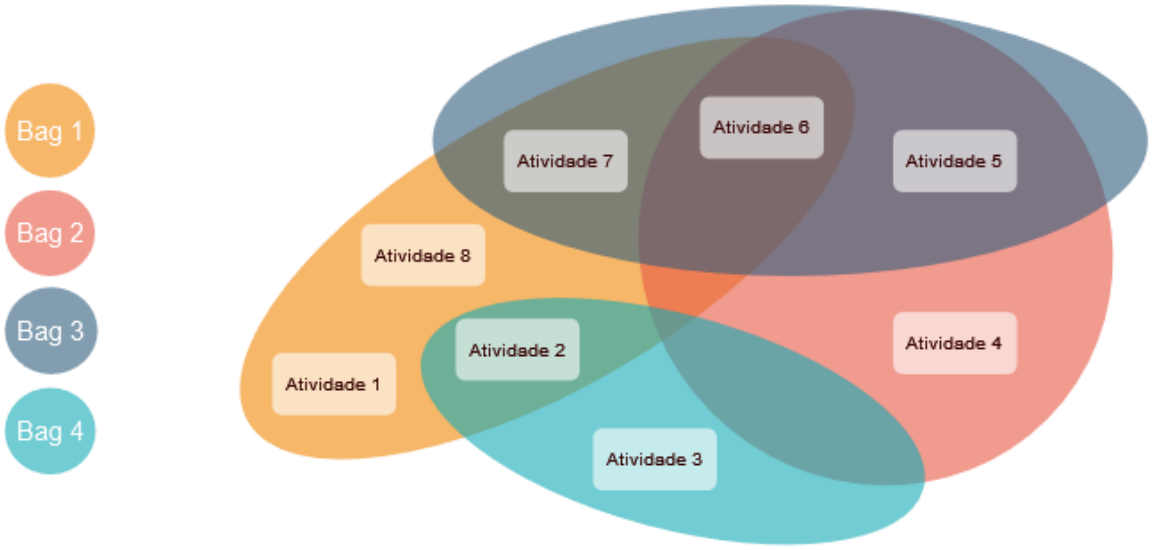

Figura 2.2: Exemplo de processo não estruturado - representação em nuvem.

Bag of Activities

Utilizado anteriormente, o conceito de *bag of activities* é crucial para a compreensão de processos não estruturados. Embora não exista uma estrutura fixa, com ordenamento lógico e requerimentos bem-definidos, ainda existem possíveis agrupamentos dentre as diversas atividades envolvidas em um UBP. Sendo assim, é possível agrupar tais atividades com base em uma característica única - seja o ator, o objetivo, ou um atributo semelhante -, de forma que atividades semelhantes em algum quesito sejam agrupadas em conjunto. O agrupamente de tais atividades é chamado de *bag*, e é definição de uma *bag of activities*.

No presente trabalho, os agrupamentos serão feitos com base nas informações de domínio, descritas na forma de ontologias, descritas na sub-seção seguinte. Assim, as *bags* terão, tipicamente, atributos semelhantes de ordem semântica, possibilitando que o motor de inferência do sistema proposto aja corretamente.

2.1.3 Processos Não Estruturados com Segmentos Pré-definidos

Por fim, existem processos que possuem características de processos estruturados e de processos não estruturados. Tipicamente, possuem atividades explicitas e bem descritas, comumente delimitadas por políticas empresariais, normas ou regulamentações, embora o restante do processo seja majoritariamente não estruturado, intensivo em conhecimento [7].

Alguns exemplos são investigações de cláusulas de seguro e corretagem de imóveis, cujas operações são individualizadas e possuem processos distintos caso-a-caso, mas ainda são regulamentadas e devem seguir um conjunto de normas em determinadas etapas. Assim, o processo é majoritariamente não-estruturado, mas existem etapas bem definidas, repetitivas e sistemáticas. Uma interpretação visual do espectro pode ser observada na Figura 2.3.

2.2 Espectro de Classes de Processos

A partir das classes de processos definidas anteriormente, é possível observar um espectro de gestão de processos com base nos seguintes fatores:

1. Facilidade de modelagem;
2. Previsibilidade;
3. Repetitividade;
4. Flexibilidade;
5. Grau de conhecimento necessário;

Pelo espectro, é possível perceber relações diretas entre a classificação de processos e facilidade de modelagem, previsibilidade e repetitividade. Relações inversamente proporcionais são observadas nos fatores restantes, flexibilidade e grau de conhecimento necessário. Assim, processos não estruturados com segmentos pré-definidos tipicamente contam com menos repetitividade, menos previsibilidade e mais dificuldade de modelagem do que processos puramente estruturados. O mesmo pode ser dito para processos estritamente não estruturados, mesmo quando comparados com outras classes no meio do espectro.

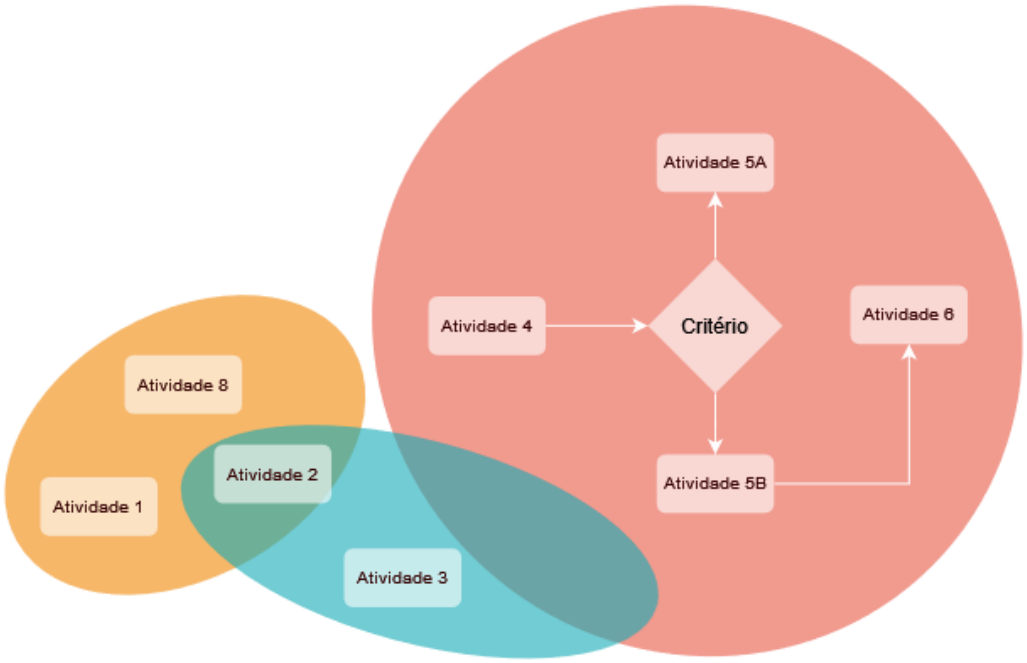

Figura 2.3: Exemplo de processo não estruturado com segmento pré-definido.

Uma interpretação visual do espectro pode ser observada na Figura 2.4.

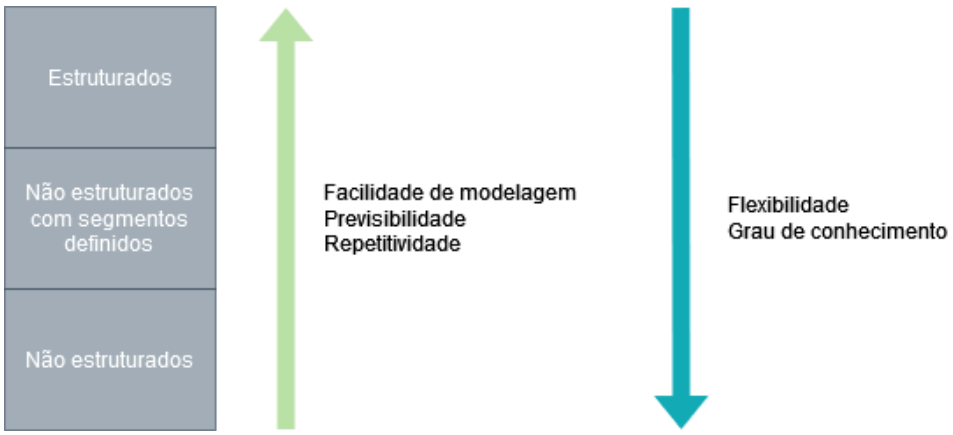

Figura 2.4: Espectro de classes de processo.

2.3 Ontologia

O termo ontologia tem um longo histórico de utilização na filosofia, frequentemente utilizado para discutir problemas relacionados à realidade. No contexto da gestão de processos de gestão de conhecimento, ontologia é referenciado para descrever o conhecimento obtido a respeito de determinado domínio. Sendo assim, ontologias tipicamente contém informações à respeito de entidades, relacionamentos, axiomas, descrições semânticas, e até instâncias de classes [10]. Pode ser visto como uma linguagem própria, capaz de descrever de sua própria maneira o domínio em questão.

A partir disso, ontologias são capazes de prover as seguintes funcionalidades:

1. Compartilhamento de conhecimento: o uso de uma única base de conhecimento a respeito de um mesmo domínio possibilita que agentes e processos diferentes contenham as mesmas informações semânticas e lógicas, possibilitando a comunicação entre eles;

2. Inferência lógica: com base na ontologia, é possível inferir e deduzir logicamente artefatos de alto-nível, sejam eles atributos, relacionamentos, instâncias, ou quaisquer outros tipos de objetos presentes na ontologia utilizada. Ainda, é possível resolver problemas complexos, e verificar a corretude semântica das soluções produzidas, garantindo interpretações corretas do contexto utilizado;

3. Reutilização de conhecimento: a partir de ontologias já utilizadas amplamente na industria e na academia, é possível produzir novas, mais complexas, mais abrangentes ontologias, por sua vez com mais relacionamentos e melhores capacidades de inferência.

2.3.1 Base do Conhecimento

Uma base de conhecimento consiste em um agrupamento de conhecimento representado mediante uma técnica adequada ao sistema em questão, a respeito de determinado domínio [11]. Essas informações podem ser carregadas e utilizadas posteriormente na solução de problemas, por meio de ferramentas de Inteligência Artificial (IA) ou sistemas especialistas, como motores de inferência. Neste contexto, uma ontologia para sistemas baseados em conhecimento é uma especificação para os objetos, conceitos, outras entidades e o relacionamento entre eles que possam existir em alguma área de interesse. Portanto, quanto maior o grau de corretude e completude da sua base de conhecimento, melhor será a saída do sistema.

2.3.2 Motor de Inferência

Para realizar as deduções e induções, existe o motor de inferência [12, 13]. A partir dele, é possível obter deduções e induções lógicas com base em dados contextuais e em definições do usuário, utilizando os conceitos definidos na ontologia.

Ainda, é possível verificar a consistência e corretude de informações contextuais de alto-nível a partir de informações de baixo nível (indução), bem como o contrário (dedução), permitindo que informações contextuais implícitas estejam disponíveis para utilização. Além disso, a partir de definições ontológicas simples - como propriedades inversas, herança, etc. -, é possível obter propriedades tão complexas quanto se desejar. Em conjunto, tais propriedades garantem ao motor de inferência (e, consequentemente, ao projeto) um altíssimo grau de flexibilidade.

2.4 Modelagem Semântica

Como propõe Hepp (2017) [14], instituições modernas estão enfrentando problemas cujas soluções (atualmente, precárias) seriam melhor solucionadas com a utilização de tecnologias semânticas como ontologias e motores semânticos por meio de soluções modernas envolvendo UBP.

A partir disso, surge o conceito de modelagem semântica [15, 16], que consiste em anotar, semanticamente, modelos BPMN, de forma que contenham referências a elementos ontológicos, objetivos, relacionamentos e restrições semânticas. Com tais informações, o motor semântico seria capaz de atuar corretamente e completamente, produzindo valiosas informações contextuais.

Por fim, para que o motor semântico atue em cima das atividades e processos descritos na fase de modelagem, é necessário que existam referências entre atividades e entidades semânticas, de forma que seja possível identificar todas as informações contextuais explícitas a respeito de determinada etapa do processo, seja ele estruturado ou não.

Capítulo 3

Metodologia

A fim de contextualizar o projeto, vê-se a necessidade de discutir os componentes a serem desenvolvidos pelo presente trabalho, bem como a arquitetura geral da solução. Nesta seção, portanto, serão apresentadas quais ferramentas foram utilizadas e como os componentes se encaixam e se comunicam entre si, de forma a atingir os objetivos gerais e específicos descritos na seção Introdução.

Ainda, este trabalho utiliza o método de pesquisa Design Science Research (DSR)[17]. Também conhecida como *constructive research*, a DSR é uma abordagem metodológica que consiste em construir artefatos que preencham a lacuna entre pesquisa e a aplicação prática. Uma ilustração da metodologia pode ser observada na Figura 3.1.

A partir da figura, podemos observar as seguintes etapas:

1. Identificação e conscientização, do problema; no presente trabalho, encontrados na seção Introdução. É responsável por definir, de forma clara, as causas e consequências do problema abordado;

2. Identificação dos artefatos e configurações das classes de problemas. Econtrado na seção de metodologia, é onde é feito uma descrição mais detalhada dos artefatos a serem produzidos pelo trabalho, bem como uma classe geral de problemas capazes de serem solucionados pelo sistema proposto;

3. Proposição de artefatos para resolver o problema específico, com detalhamento a respeito da solução desenvolvida, bem como de escolhas de arquitetura. Encontrado na seção de modelagem;

4. Desenvolvimento e avaliação dos artefatos, encontrados na seção Resultados, mostram os artefatos produzidos e seu funcionamento em conjunto, cumprindo os objetivos propostos;

5. Explicitação dos aprendizados e generalização para uma classe de problemas; encontrados na seção de resultados.

Portanto, o trabalho utiliza por completo a metodologia Design Science Research (DSR).

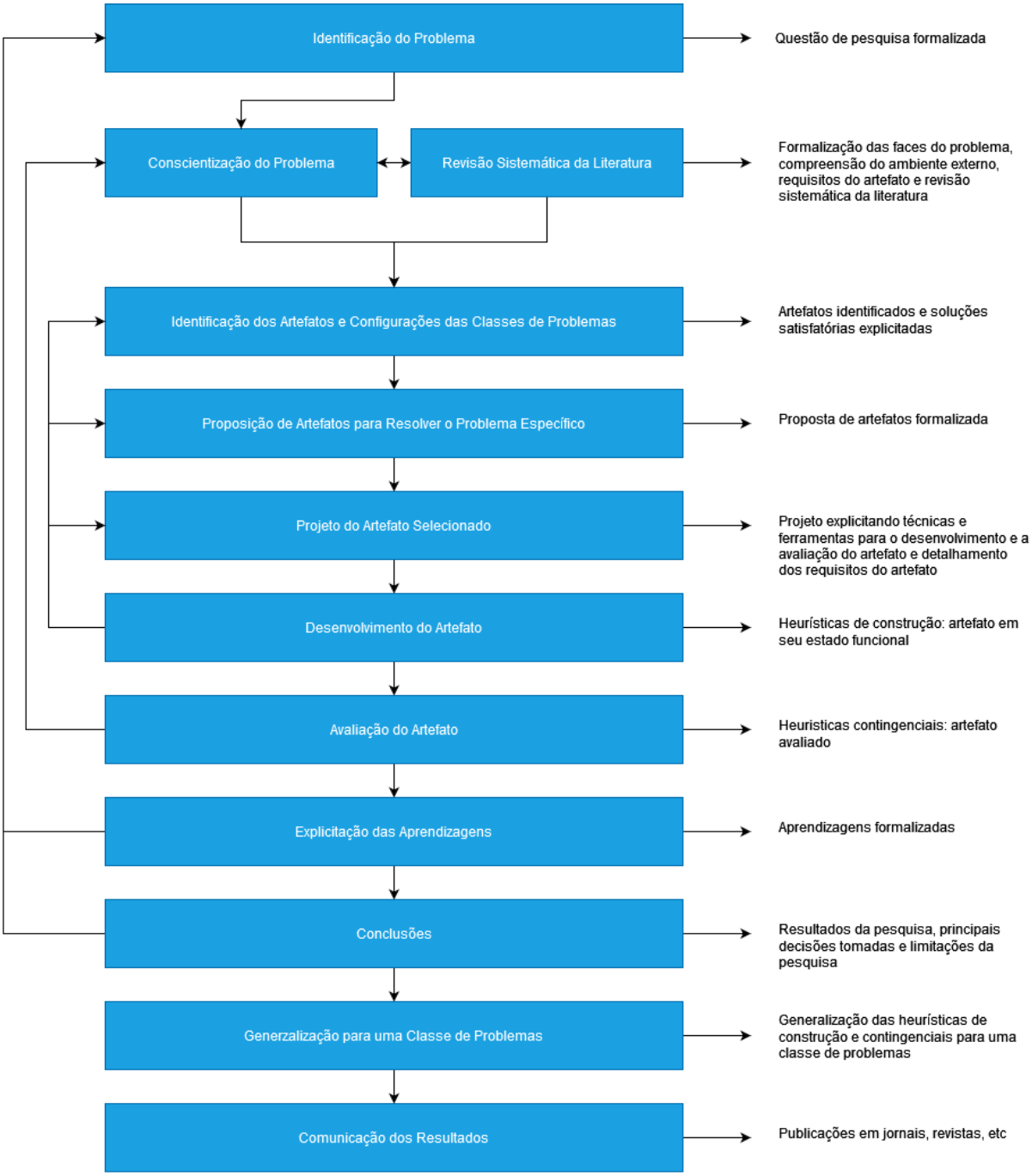

Figura 3.1: Descritivo da metodologia DSR.

3.1 Descrição do Estudo de Caso

Como estudo de caso, propõe-se um processo intensivo em conhecimento, de forma que dificilmente poderia ser implementado com os limites técnicos de processos estruturados. Na situação-problema, já trabalhada por Guilherme (2021) [18], a comissão de formatura do Departamento de Ciência da Computação deseja realizar a festa de formatura de seus alunos.

Para a realização do evento, os organizadores planejaram tarefas inicias e as dividiram em 5 *bags of activities*, de forma que cada bag seja responsável por apenas 1 grande objetivo em comum. Sendo assim, as bags escolhidas foram:

1. Definição de equipes, e atribuições de responsabilidades para cada equipe e seus respectivos integrantes;
2. Definição de atributos gerais da festa de formatura, como data, local, tema, número de convites por aluno, orçamento máximo disponível, etc.;
3. Análise de opções, buscando encontrar o melhor custo por benefício;
4. Planejamento de contingências;
5. Execução do evento;

Cada *bag* possui atividades relacionadas aos seus respectivos objetivos, com grau variável de granularidade e de descrição. No presente trabalho, a bag de número 2 será expandida, de forma a analisar os efeitos do motor semântico e validar ou não sua utilidade prática.

3.2 Artefatos Produzidos

Para isso, de acordo com o descrito na seção anterior, será necessário desenvolver, mesmo que brevemente, uma ontologia do domínio "festa de formatura". Nela, deverão estar contidas informações e descrições relativas suas entidades e relacionamentos. Nesse caso, os relacionamentos estarão concentrados no relacionamento entre atividades, e em seu requisitos semânticos.

Assim, a partir das atividades descritas na bag em questão, a associação do motor semântico é capaz de inferir novas atividades que deveriam estar contidas, e a integração com o banco de dados permite que seja realizada uma checagem automática da presença ou não de tais atividades, de forma que o processo pode ocorrer naturalmente, sem eventuais duplicatas.

Por fim, o último requisito do estudo de caso é que o processo de inferência ocorra sem intervenção manual, e apenas a quantidade de vezes mínima necessária para que todas as atividades sejam inferidas e adicionadas com sucesso.

Capítulo 4

Trabalhos Relacionados

Nessa etapa foram realizadas pesquisas de trabalhos relacionados e ferramentas que propõem abordagens relacionadas ao projeto em questão. Para isso, os critérios de análise envolvem os seguintes pontos:

1. Disponibilidade de bibliotecas (ou capacidade nativa) para suporte de gerenciamento de processos de negócio em BPMN;
2. Capacidade de gestão de processos de negócio não estruturados ou não estruturados com segmentos pré-definidos;
3. Ferramenta de modelagem flexível para modificações e extensões;
4. Capacidade de suportar ferramentas semânticas;
5. Comunidade ativa;
6. Documentação compreensiva e descritiva;
7. Limitações gerais;

Para isso, duas opções foram encontradas: YAWL[19], Camunda[20], aplicações de Business Process Management (BPM) com capacidade de estruturação completa de processos usando a notação BPMN. Apesar de possuírem módulos de comunicação externa com HTTP (e, portanto, estarem abertas à certo grau de extensibilidade), ambas não possuem suporte à processos não estruturados. Ainda, o nível de atividade da comunidade e a completude da documentação também foram pontos não ideais, com poucos participantes ativos nos fóruns oficiais e descrições de funcionalidades ocasionalmente não existentes. Por fim, YAWL não apresenta suporte para alteração e adição de atributos em atividades, de forma a inviabilizar a anotação semântica, que se dá por meio de uma referência à IRI daquela classe de atividades na ontologia.

Tabela 4.1: Critérios por Ferramenta.

Critério	**Yawl**	**Camunda**	**Django**
Suporte à BPMN	Sim	Sim	Sim
Processos estruturados	Sim	Sim	Sim
Processos não estruturados	Não	Não	Sim
Processos não estruturados c/ partes definidas	Não	Não	Sim
Extensibilidade	Mínimo	Bom	Ótimo
Suporte à ferramentas semânticas	Não	Não	Sim
Atividade da comunidade	Mínimo	Bom	Ótimo
Qualidade & completude da documentação	Bom	Ótimo	Ótimo

Como uma terceira opção, foi analisado o framework Django, que difere das outras opções por não ser uma ferramenta por si só, e sim um framework de desenvolvimento. Com ele, é possível modelar entidades, classes, relacionamentos, e trabalhá-las em um banco de dados relacional. Sendo assim, o *tradeoff* da escolha se encontra no fato de o *motor de workflow* não ser implementado nativamente, devendo também ser desenvolvido. Por isso, optar pela utilização do Django implica em uma maior quantidade de trabalho total, já que o motor de workflow também deverá ser implementado (versus a utilização dos motores disponíveis com YAWL e Camunda). Contudo, a opção é suficientemente flexível, contemplando capacidade para comunicação externa, comunicação com o motor semântico e suporte à diversas bibliotecas semânticas.

Um comparativo entre as opções pode ser observado na Tabela 4.1.

A partir da Tabela 4.1, é possível observar que todos os critérios são amplamente satisfeitos pelo framework Django, ao passo que as demais ferramentas possuem claros pontos negativos. Ainda, é válido ressaltar que o critério contemplando o suporte à ferramentas semânticas é absolutamente necessário para o projeto, inviabilizando a escolha das ferramentas YAWL e Camunda.

Tendo em vista a inexistência de ferramentas que contemplem todos os critérios requeridos pelo trabalho, é evidente que existe uma lacuna na área de gestão de processos não estruturados. Sendo assim, com auxílio de um motor semântico, o artefato produzido será capaz de suportar nativamente todo o espectro de processos, e prover inferências semânticas quando necessário.

Capítulo 5

Modelagem

Para a modelagem do artefato deste trabalho, foram consideradas as ferramentas descritas na seção anterior, bem como suas eventuais integrações e futuros trabalhos possivelmente provenientes da presente modelagem e implementação.

5.1 Descrição da Solução

A partir dos objetivos descritos na sessão de introdução, assim como os requisitos apresentados anteriormente, será apresentada, nessa seção, a solução encontrada. Para tanto, serão detalhadas as ferramentas utilizadas, a arquitetura proposta, e, finalmente, o modelo entidade-relacionamentos do banco de dados associado à solução.

5.1.1 Ferramentas utilizadas

Primeiramente, para a implementação de uma ontologia do domínio "festa de formatura", foi utilizado o software Protégé[21], por Braga (2021) [18]. Contudo, devido ao fato de a ontologia não estar completa o suficiente para contemplar processos não estruturados com as informações necessárias para o motor de workflow semântico, definições extras foram adicionadas utilizando *owlready2*[22], biblioteca em Python. Com ele, é possível definir classes, instâncias, relacionamentos e realizar inferências usando o motor semântico Pellet [12]. Ainda, o código produzido é simultaneamente interpretável por leitores OWL[23] e utilizável em tempo de execução por interpretadores Python.

Sendo assim, integrações com o banco de dados são facilitadas, possibilitando edição/adição/deleção de forma simples e elegante. Por fim, a biblioteca permite que o trabalho esteja contido em um único ecossistema Python, facilitando manutenções e desenvolvimentos futuros.

Como software de gestão de processos, foi utilizada o framework aberto Django[24], ainda em Python, capaz de criar modelos, tabelas, e relacionamentos, além de integrações diretas com o banco de dados, de forma a simplificar o trabalho. Ainda, o robusto sistema de sinais pré-adição e pré-edição permite que atividades adicionadas manualmente também ativem o sistema de inferência automaticamente.

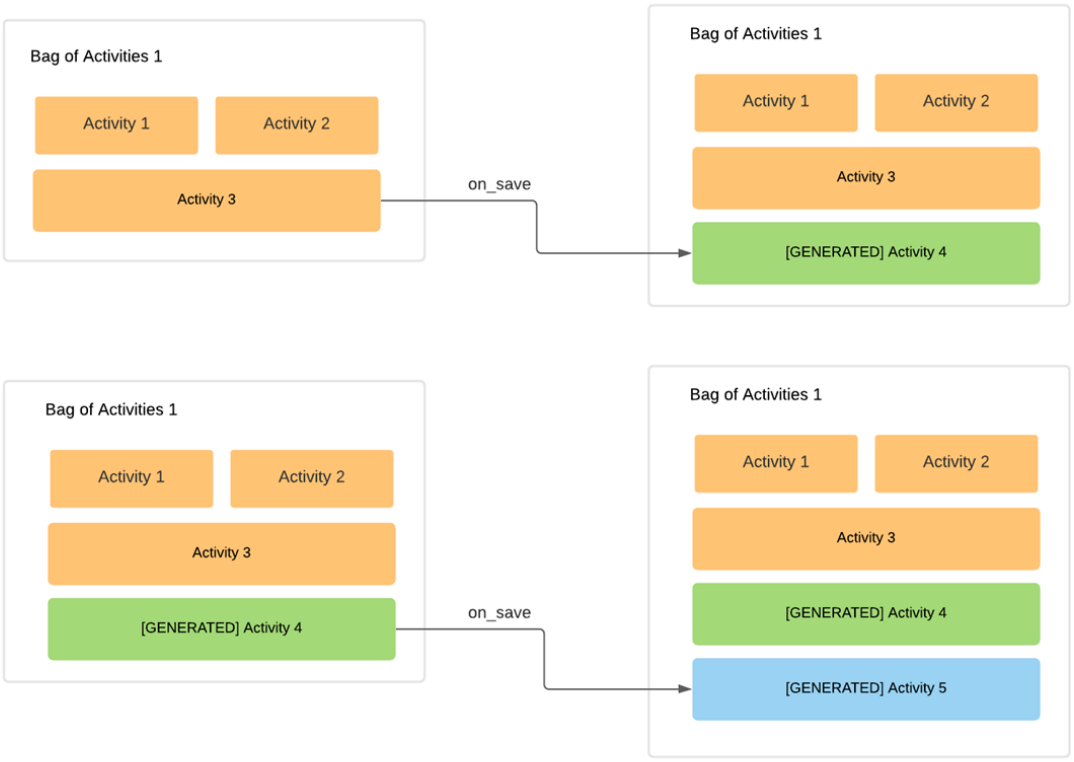

Figura 5.1: Estrutura de sinais integrada com motor semântico.

A Figura 5.1 explicita o processo descrito anteriormente. Nela, é simulada a adição de uma nova atividade, descrita por "Activity 3". Ao adicioná-la, um sinal *on_save* é enviado, que chamará o motor de inferência, que, a partir das regras semânticas descritas com o *owlready2*, criará uma nova atividade. Essa nova atividade, por sua vez, também irá gerar um sinal *on_save*, de forma que todo o processo se dá de forma recursiva até que não haja nenhuma nova inferência.

5.1.2 Motor de Workflow

O motor de workflow foi desenvolvido, como descrito na seção de metodologia, a partir da biblioteca Django, seguindo a nomenclatura proposta pelo padrão BPMN[6]. Ainda,

os conceitos de classes definidos na notação foram abstraídos de forma a se beneficiar de heranças múltiplas, facilitando as diversas referências entre objetos, a manutenibilidade e a extensibilidade do artefato. Assim, foram definidos os seguintes principais modelos:

1. FlowElement, unidade atômica do processo BPMN; é responsável por representar todos os elementos associados ao fluxo de processos estruturados;
2. FlowNode, que herda de FlowElement; que representa um nó do fluxo;
3. SequenceFlow, que herda de FlowElement; que possui informações a respeito da ordem de execução de processos, contendo referências ao nó atual e possíveis futuros nós;
4. Activity, que herda de FlowNode; que representa uma atividade num processo qualquer;
5. FlowElementsContainer, que representa um container, ou um grupo de atividades associadas;
6. Process, que herda de FlowElementsContainer e representa um processo;
7. Bag, que representa a *bag of activities* para processos não estruturados.

Outras classes não necessitam de maiores distinções e explicações, e existem apenas como exercício de implementação para processos estruturados. Relacionamentos entre modelos possuem tabelas próprias, e seus *constraints* de chave estrangeira são lidados automaticamente pelo Django.

5.1.3 MER

Por fim, o modelo entidade-relacionamentos proposto para que todos os objetivos e necessidades fossem supridos é apresentado a seguir, na Figura 5.2, disponível abaixo. Nela, é possível identificar 5 seções distintas, delimitadas por 3 cores diferentes. Em verde, modelos e relacionamentos associados à processos de negócio não estruturados (que possuam ou não partes pré-definidas). Em azul, classes necessárias para que o motor de workflow seja capaz de garantir uma ordem lógica relacional entre as atividades dos processos. Em cinza, tabelas frequentemente utilizada por todo o espectro de processos (como Eventos).

No modelo, é possível notar alguns pontos:

1. Atividades, Eventos, e Subprocessos são todas subclasses de FlowElement, seguindo a implementação proposta por [15];
2. Containers (de forma geral, piscinas) possuem Lanes (ou raias), que possuem grupos associados responsáveis por representar os diversos papéis de cada equipe;

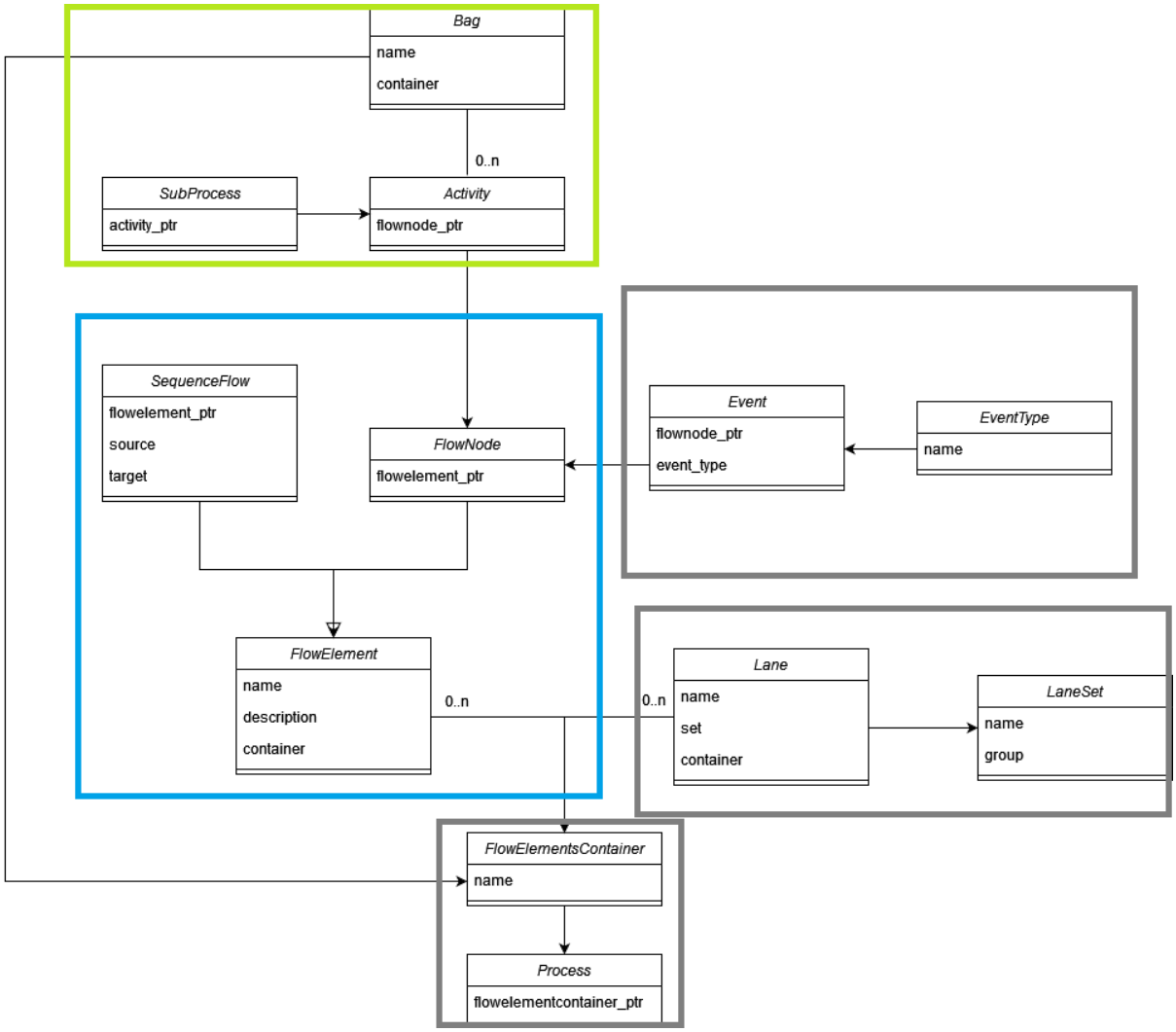

Figura 5.2: Modelo entidade relacionamento da solução proposta.

3. Processos e bags possuem heranças diretas com FlowElementContainer, de forma que, quando associados com subprocessos, podem formar associações recursivas. Dessa forma, é possível descrever processos de quaisquer complexidade, e complementar processos não estruturados com subprocessos estruturados.

Capítulo 6

Resultados

Resultados e análises aqui apresentadas são fundamentadas na observação empírica da solução proposta pelo trabalho. Para isso, serão evidenciados maiores detalhes a respeito da implementação e de suas escolhas, além da validação e descrição do funcionamento da aplicação final.

6.1 Desenvolvimento

Aqui, serão discutidos aspectos relativos à implementação da solução para o problema proposto. Códigos produzidos estão disponíveis no apêndice do trabalho, ou em repositórios de endereço público. No Apêndice A, encontra-se o endereço online do repositório, além de instruções para instalação e execução do projeto. No Apêndice B, o código produzido para estender a ontologia original (que se encontra no Anexo I) com as definições semânticas necessárias para o estudo de caso. No Apêndice C, encontram-se as definições para a modelagem do motor de workflow, feitas em Django.

6.1.1 Ontologia

Originalmente, a ontologia da festa de formatura possui as seguintes classes: *Course* e *GraduationCeremony*, que herdam de *Evento*, que herda de *Thing*; *Administrators* e *Professor*, que herdam de *Employee*, que herda de *Person*; *Graduates*, que herda de *Student*, que herda de *Person*. É possível visualizar tais relações entre classes na Figura 6.1, disponível abaixo.

Por sua vez, a extensão da ontologia desenvolvida para ampliar a descrição do domínio possui a seguinte base do conhecimento:

1. Antes que seja possível definir uma data final para um evento qualquer, primeiro devemos saber quais datas são possíveis;

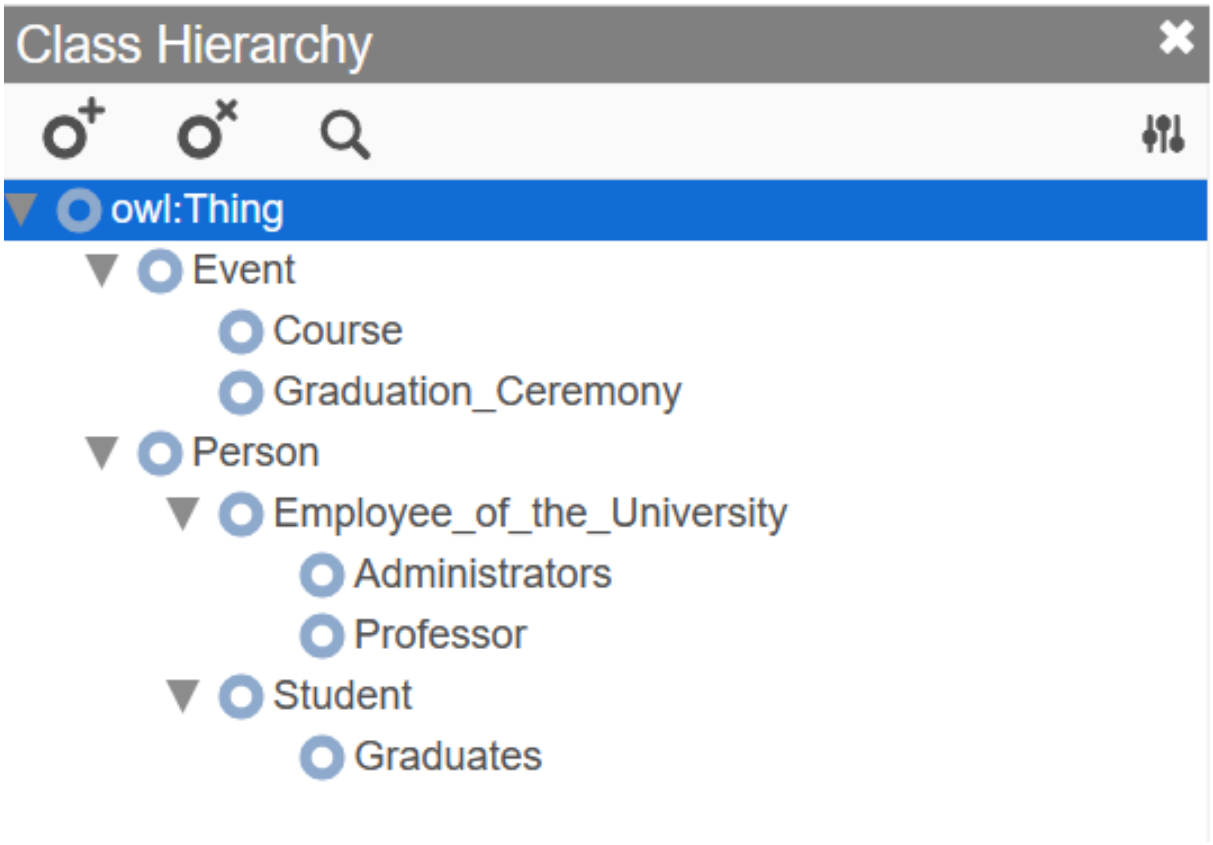

Figura 6.1: Regras semânticas descritas na ontologia original. .

2. Uma data é tida como possível se está disponível;
3. Uma data é disponível se não possui nenhuma restrição associada;
4. Uma data não possui restrições associadas se todas as pessoas de interesse podem comparecer.

É possível visualizar tais descrições semânticas na Figura 6.2. Nela, é possível identificar 5 novas classes, e algumas definições semânticas, destacadas em vermelho. Na primeira marcação, descreve-se que uma *AvailableDate* é uma *Date* que não possui restrições (neste caso, de comparecimento). Na segunda marcação em vemelho, identifica-se que a *DateCollectionActivity* requer que exista também uma atividade *IdentifyPOIActivity*. O mesmo tipo de relacionamento pode ser observado na *DateDefinitionActivity*, que requer a existência de uma atividade *DateCollectionActivity*.

De acordo com o descrito em seções anteriores, conter definições semânticas de classes, como, por exemplo, datas disponíveis para realização de um evento, possibilitam que todas as atividades associadas à datas disponíveis sejam alteradas com uma única alteração semântica. Tal procedimento permite que o sistema proposto seja mais facilmente

```
class Date(Thing):
    defined_class = True
    date = None
    has_restriction = []


class AvailableDate(Thing):
    defined_class = True
    equivalent_to = [
        Date & (not Date.has_restriction)
    ]


class IdentifyPOIActivity(Thing):
    person = None
    date = None
    description = "Atividade de identificação de pessoas de interesse para o evento"

class DateCollectionActivity(Thing):
    requires = IdentifyPOIActivity
    person = None
    date = None
    description = "Atividade de coleta de possíveis datas para o evento"

class DateDefinitionActivity(Thing):
    requires = DateCollectionActivity
```

Figura 6.2: Regras semânticas descritas na extensão da ontologia. Em vermelho, regras associadas às definições de disponibilidade da datas, e à atividades requeridas.

manutenível e expansível, pois a alteração em um único lugar (por exemplo, na ontologia) acarreta mudanças sincronizadas no sistema informacional associado, bem como em todos os processos que utilizem a definição modificada. Ainda, devido à natureza semântica da informação, o motor de workflow é capaz de validar as relações e garantir a corretude das modificações.

6.2 Funcionamento

Para evidenciar o funcionamento da aplicação, a *bag* de definição de atributos gerais para o evento (identificada na seção Descrição do Estudo de Caso como *bag* 2) contém as seguintes atividades, sem definição lógica de ordenação:

1. Montar equipe de organização do evento;
2. Reunião de planejamento;

3. Definição de métodos para angariar recursos;
4. Definição de data;
5. Definição de local.

A partir dessas atividades e da ontologia de domínio descrita anteriormente, é possível perceber que a atividade de definição de data deveria ocorrer apenas após definir quais datas são possíveis, o que por si só poderia ocorrer depois de definir quais pessoas são tidas como "pessoa de interesse", isto é, aquelas que são absolutamente essenciais no evento de formatura.

A Figura 6.3, disponível abaixo, mostra-se o estado da *bag* antes da atuação do motor de inferência.

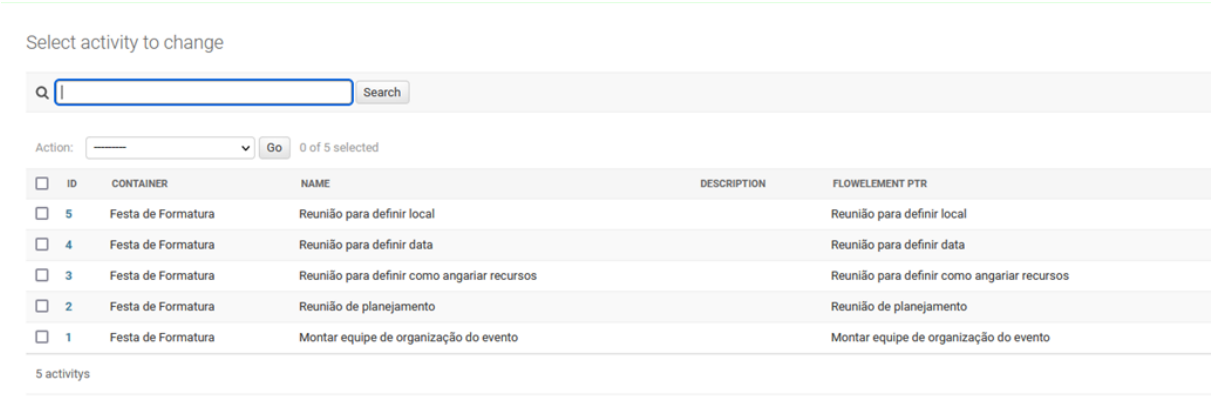

Figura 6.3: Estado inicial da *bag*.

Para que o motor de inferência possa atuar, é necessário referenciar as atividades existentes com as descritas na ontologia. Para isso, basta preencher o atributo IRI das atividades contidas na bag. Com isso feito, basta aplicar as mudanças no banco, que deverá acarretar na chamada do sinal on_save, processo mostrado na Figura 6.4. Na figura, em vermelho, destacam-se o campo IRI, que irá modificar o atributo equivalente na ativade, e o botão para salvar as modificações no banco.

Além de edições, a adição de novas instâncias de atividades também ativam o sinal,

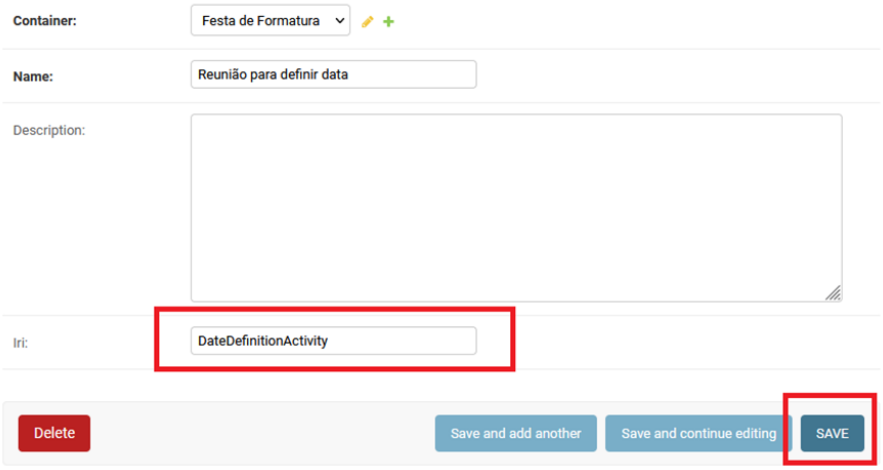

Figura 6.4: Edição de atividades na bag, pelo banco de dados.

Na Figura 6.5, é possível perceber que a chamada do método save faz com que todas as bags associadas a determinada atividade (na eventualidade de que uma atividade participe de mais de uma bag) sincronizem seus motores semânticos, que lerá os atributos de todas as atividades associadas. Como descrito na ontologia, a atividade DateDefinition requer que outras atividades estejam presentes na bag, e, portanto o motor semântico as adiciona.

```python
class Activity(FlowNode):
    iri = models.CharField(max_length=255, blank=True, null=True)

    def save(self, *args, **kwargs):
        super(Activity, self).save(*args, **kwargs)

        if self.iri:
            for bag in self.bags.all():
                bag.sync_semantic_reasoner()
```

Figura 6.5: Método a ser chamado no sinal on_save de classes Activity.

Ao fim do processo, a bag que antes continha apenas 5 atividades, agora contém todas as atividades requeridas pelas regras semânticas, destacadas em vermelho na imagem 6.6.

ID	CONTAINER	NAME	DESCRIPTION	FLOWELEMENT PTR
68	Festa de Formatura	semantic auto-generated object of class IdentifyPOIActivity	Atividade de identificação de pessoas de interesse para o evento	semantic auto-generated object of class IdentifyPOIActivity
67	Festa de Formatura	semantic auto-generated object of class DateCollectionActivity	Atividade de coleta de possíveis datas para o evento	semantic auto-generated object of class DateCollectionActivity
5	Festa de Formatura	Reunião para definir local		Reunião para definir local
4	Festa de Formatura	Reunião para definir data		Reunião para definir data
3	Festa de Formatura	Reunião para definir como angariar recursos		Reunião para definir como angariar recursos
2	Festa de Formatura	Reunião de planejamento		Reunião de planejamento
1	Festa de Formatura	Montar equipe de organização do evento		Montar equipe de organização do evento

Figura 6.6: Estado final da bag, após inferência do motor semântico.

Capítulo 7

Conclusão

Com base no apresentado na validação teórica e prática apresentada na seção anterior, mostrou-se que o motor de workflow semântico é capaz de, a partir de modificações na ontologia, realizar sincronizações em múltiplas atividades e garantir a corretude delas. Assim, o motor conclui com êxito os objetivos gerais. Além disso, devido ao procedimento descrito com base em eventos e sinais do Django, o requerimento de que o fluxo de inferência ocorra apenas quando necessário também foi exitoso e demonstrado.

Por fim, os objetivos específicos também foram atingidos. A melhoria de manutenibilidade pode ser observada ao considerarmos atualizações de regras de negócio. A melhoria de flexibilidade, por sua vez, ao considerarmos que os processos são não estruturados (e, portanto, mais flexíveis por definição), bem como a facilidade em adicionar novas atividades à processos já existentes. Por fim, a verificação da ontologia garante a corretude semântica de processos, desde que estejam suficientemente descritos.

Dessa forma, as vantagens de motores de workflow semânticos com processos não estruturados são significativas.

7.1 Trabalhos Futuros

Naturalmente, embora os objetivos propostos tenham sido concluídos exitosamente, existe espaço para melhoria. Nesse sentido, trabalhos futuros devem se concentrar nos seguintes aspectos:

1. Suporte a agentes e a metodologias Belief-Desire-Intention (BDI), que deverão garantir ainda mais flexibilidade e maior potencial para inferir situações de contorno;
2. Suporte a Processos Intensivos em Conhecimento, abrangindo a KIPO [18] e suas subontologias;

3. Suporte a APIs RESTful, abrindo o motor de inferência e o processo de gestão de processos não estruturados para ambientes externos, de forma que o sistema se torne altamente integrável com estruturas internas já existentes em organizações que objetivem utilizar o artefato produzido no trabalho;
4. Ampliar o escopo de inferências suportadas;
5. Desenvolver novos, ainda mais abrangentes e mais flexíveis estudos de caso, com múltiplas inferências simultâneas;

Em específico para a sugestão de suporte à agentes e à metodologias BDI, é proposto que seja utilizada a biblioteca SPADE[25]. Entre seus pontos positivos, encontram-se os fatos de ela também ser desenvolvida em Python, de forma que o eco sistema do projeto se mantenha com apenas uma linguagem, garantindo maior mantenabilidade futura. Além disso, o SPADE conta com suporte nativo para comunicação inter-agente, além de comunicação via internet, com protocolo XMP, facilmente extensível para suportar HTTP e HTTPS, possibilitando abertura a APIs RESTful.

Referências

[1] *Why you should use BPM software*, 2021. https://monday.com/blog/project-management/bpm-software/, acesso em 2021-11-09, Section: Project management. 1

[2] Nadarajah, Devika e Sharifah Latifah Syed Abdul Kadir: *A review of the importance of business process management in achieving sustainable competitive advantage*. The TQM Journal, 26(5), 2014, ISSN 1754-2731. https://doi.org/10.1108/TQM-01-2013-0008, acesso em 2021-11-02, Publisher: Emerald Group Publishing Limited. 1, 5

[3] Aguilar-Savén, Ruth Sara: *Business process modelling: Review and framework*. International Journal of Production Economics, 90(2), 2004, ISSN 0925-5273. https://www.sciencedirect.com/science/article/pii/S0925527303001026, acesso em 2021-11-02. 5

[4] Hung, Richard Yu Yuan: *Business process management as competitive advantage: a review and empirical study*. Total Quality Management & Business Excellence, 17(1), 2006, ISSN 1478-3363. https://doi.org/10.1080/14783360500249836, acesso em 2021-11-02, Publisher: Routledge _eprint: https://doi.org/10.1080/14783360500249836. 5

[5] *Business process model and notation (BPMN), version 2.0*. 6

[6] *Business process model and notation specification version 2.0*. https://www.omg.org/spec/BPMN/2.0/, acesso em 2021-11-06. 6, 20

[7] Di Ciccio, Claudio, Andrea Marrella e Alessandro Russo: *Knowledge-intensive processes: Characteristics, requirements and analysis of contemporary approaches*. Journal on Data Semantics, 4, 2015. 7, 9

[8] *Case management and smart process applications*. https://www.aiim.org/, acesso em 2021-11-06. 7

[9] Jaliniauskas, Alvydas: *The analysis of unstructured processes in business administration*. Informatica, 11(2), 2000, ISSN 0868-4952. https://content.iospress.com/articles/informatica/inf11-2-02, acesso em 2021-11-02, Publisher: IOS Press. 8

[10] Xiao Hang Wang, Da Qing Zhang, Tao Gu e Hung Keng Pung: *Ontology based context modeling and reasoning using OWL*. Em *IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second.* IEEE, 2004, ISBN 978-0-7695-2106-0. http://ieeexplore.ieee.org/document/1276898/, acesso em 2021-11-02. 11

[11] Galhardo, Raphaela: *Modelagem semântica de processos industriais com aplicações.* https://1library.org/document/oz154mey-modelagem-semantica-de-processos-industriais-com-aplicacoes.html, acesso em 2021-11-02. 11

[12] Sirin, Evren, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur e Yarden Katz: *Pellet: A practical OWL-DL reasoner*. Journal of Web Semantics, 5(2), 2007, ISSN 1570-8268. https://www.sciencedirect.com/science/article/pii/S1570826807000169, acesso em 2021-11-02. 12, 19

[13] Pinto, Guilherme Braga: *Software para desenho de processos de negócios semanticamente descritos: uma aplicação em uma redação jornalística.* 2015. 12

[14] Hepp, Martin e Dumitru Roman: *An ontology framework for semantic business process management.* Wirtschasinformatik Proceedings, 2007. 12

[15] Dimitrov, Marin, Alex Simov, Sebastian Stein e Mihail Konstantinov: *A BPMO based semantic business process modelling environment.* Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management SBPM, 2007. 12, 21

[16] Corea, Carl, Michael Fellmann e Patrick Delfmann: *Ontology-based process modelling – will we live to see it?* arXiv:2107.06146 [cs], 2021. http://arxiv.org/abs/2107.06146, acesso em 2021-11-08. 12

[17] Lacerda, Daniel, Aline Dresch, Adriano Proença e José Antonio Valle Antunes Júnior: *Design science research: A research method to production engineering.* Gestão & Produção, 20, 2012. 13

[18] Guilherme Braga Pinto, Flávia Santoro e Edison Ishikawa: *Implementação da knowledge intensive process ontology em owl.* Congresso de Iniciação Científica da UnB e 18º Congresso de Iniciação Científica do DF, 2021. 15, 19, 29

[19] *Yawl bpm.* https://yawlfoundation.github.io/, acesso em 2021-11-06. 17

[20] *Workflow platform | camunda.* https://camunda.com/, acesso em 2021-11-06. 17

[21] *Protégé software | stanford edu.* https://protege.stanford.edu/, acesso em 2021-11-08. 19

[22] *Owlready2 0.35 documentation.* https://owlready2.readthedocs.io/en/v0.35/, acesso em 2021-11-06. 19

[23] *OWL web ontology language overview.* https://www.w3.org/TR/owl-features/, acesso em 2021-11-09. 19

[24] *Django documentation | django documentation | django.* https://docs.djangoproject.com/en/3.2/, acesso em 2021-11-06. 20

[25] *SPADE — SPADE 3.2.0 documentation.* https://spade-mas.readthedocs.io/en/latest/readme.html, acesso em 2021-11-06. 30

Apêndice A

Repositório e Dependências

O repositório git para o projeto se encontra no seguinte link: <https://github.com/glins97/sfdjango>
Para poder rodar o sistema como um todo, seguindo as boas práticas de desenvolvimento de software em python, será necessário primeiramente criar uma ambiente virtual. Segue, abaixo, a relação de bibliotecas necessárias.

```
asgiref==3.4.1
Django==3.2.8
django-extensions==3.1.3
djangorestframework==3.12.4
drf-generators==0.5.0
Owlready2==0.34
pytz==2021.3
sqlparse==0.4.2
typing-extensions==3.10.0.2
```

A mesma listagem está presente no arquivo *requirements.txt*, contido no repositório. Como outros pacotes em Python, é possível instalar rapidamente todas as dependências com o comando abaixo, que lerá o conteúdo do arquivo de dependências e instalará recursivamente as bibliotecas necessárias.

```
python -m pip install -r requirements.txt
```

Apêndice B

Extensão da Ontologia do Domínio

```
from owlready2 import *
onto_path.append("uploads/ontologies/formatura_ontologia.owl")
onto = get_ontology("https://raw.githubusercontent.com/gui1080/Knowledge_Intensive_Pro
onto.load()

with onto:
    class DateRestrictionType(Thing):
        defined_class = True
        name = None
        description = None


    class DateRestriction(Thing):
        defined_class = True
        person = None
        type = None


    class Date(Thing):
        defined_class = True
        date = None
        has_restriction = []


    class AvailableDate(Thing):
```

```
        defined_class = True
        equivalent_to = [
            Date & (not Date.has_restriction)
        ]


    class IdentifyPOIActivity(Thing):
        person = None
        date = None
        description = "Atividade de identificação de pessoas de interesse para o event


    class DateCollectionActivity(Thing):
        requires = IdentifyPOIActivity
        person = None
        date = None
        description = "Atividade de coleta de possíveis datas para o evento"


    class DateDefinitionActivity(Thing):
        requires = DateCollectionActivity


onto.save('generated_ontology.owl')
```

Apêndice C

Modelagem do Banco de Dados

```
from django.db import models
from django.contrib.auth.models import User, Group

from semantic.manager import get_inferred_activities

class FlowElement(models.Model):
    container = models.ForeignKey('FlowElementsContainer',
        on_delete=models.CASCADE)
    name = models.CharField(max_length=255)
    description = models.TextField(blank=True)

    def __str__(self):
        return self.name


class FlowNode(FlowElement):
    pass


class SequenceFlow(FlowElement):
    source = models.ForeignKey(FlowNode, related_name='outgoing',
        on_delete=models.CASCADE)
    target = models.ForeignKey(FlowNode, related_name='incoming',
        on_delete=models.CASCADE)
```

```
    def __str__(self):
        return self.name


class Activity(FlowNode):
    iri = models.CharField(max_length=255, blank=True, null=True)

    def save(self, *args, **kwargs):
        super(Activity, self).save(*args, **kwargs)

        if self.iri:
            for bag in self.bags.all():
                bag.sync_semantic_reasoner()



class FlowElementsContainer(models.Model):
    from kipco.models import ProcessGoal

    name = models.CharField(max_length=255)
    goal = models.ForeignKey(ProcessGoal,
        on_delete=models.CASCADE)
    ontology = models.FileField(upload_to='uploads/ontologies',
        blank=True, null=True)

    def __str__(self):
        return self.name


class Process(FlowElementsContainer):
    pass


class Task(Activity):
    pass
```

```
class SubProcess(Activity):
    pass


class EventType(models.Model):
    name = models.CharField(max_length=255)
    description = models.TextField(blank=True)

    def __str__(self):
        return self.name


class Event(FlowNode):
    event_type = models.ForeignKey(EventType,
        on_delete=models.CASCADE)

    def __str__(self):
        return self.name


class LaneSet(models.Model):
    name = models.CharField(max_length=255)
    container = models.ForeignKey(FlowElementsContainer,
        on_delete=models.CASCADE)

    def __str__(self):
        return self.name

class Lane(models.Model):
    name = models.CharField(max_length=255)
    set = models.ForeignKey(LaneSet, on_delete=models.CASCADE)
    group = models.ForeignKey(Group, on_delete=models.CASCADE)

    def __str__(self):
        return self.name

class Bag(models.Model):
```

```
name = models.CharField(max_length=255)
container = models.ForeignKey(FlowElementsContainer,
    on_delete=models.CASCADE)
activities = models.ManyToManyField(Activity, related_name='bags')
group = models.ForeignKey(Group
    on_delete=models.CASCADE, blank=True, null=True)

def sync_semantic_reasoner(self):
    if not self.container.ontology:
        return

    current_activities = [obj.name for obj in self.activities.all()]
    for obj in self.activities.all():
        if obj.iri:
            inferred_activity = get_inferred_activities(obj,
                self.container.ontology)
            if inferred_activity \
                and inferred_activity['name'] not in current_activities:
                el = FlowElement(
                    container=obj.container,
                    name=inferred_activity['name'],
                    description=inferred_activity['description']
                )
                el.save()

                fn = FlowNode(
                    flowelement_ptr=el,
                    container=el.container,
                    name=el.name
                )
                fn.save()

                act = Activity(flownode_ptr=fn,
                    container=el.container,
                    name=el.name,
                    description=el.description,
                    iri=inferred_activity['class'],
```

```
                )
                act.save()

                self.activities.add(act)
                self.save()

                act.save()


        self.save()
```

Anexo I

Ontologia de Domínio Original

A ontologia descrita abaixo pode ser encontrada em:

https://github.com/gui1080/Knowledge_Intensive_Process_Ontology_OWL/blob
/master/Ontologia%20da%20Fomatura/formatura_ontologia.owl

Ontologia de domínio original:

```
    <?xml version="1.0"?>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.semanticweb.org/guilherme_braga/ontologies/2021/7/untitled-o
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     ontologyIRI="http://www.semanticweb.org/guilherme_braga/ontologies/2021/7/untitle
    <Prefix name="" IRI="http://www.semanticweb.org/guilherme_braga/ontologies/2021/7/
    <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
    <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
    <Prefix name="xml" IRI="http://www.w3.org/XML/1998/namespace"/>
    <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
    <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
    <Declaration>
        <Class IRI="#Administrators"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Course"/>
    </Declaration>
    <Declaration>
```

```
    <Class IRI="#Employee_of_the_University"/>
</Declaration>
<Declaration>
    <Class IRI="#Event"/>
</Declaration>
<Declaration>
    <Class IRI="#Graduates"/>
</Declaration>
<Declaration>
    <Class IRI="#Graduation_Ceremony"/>
</Declaration>
<Declaration>
    <Class IRI="#Person"/>
</Declaration>
<Declaration>
    <Class IRI="#Professor"/>
</Declaration>
<Declaration>
    <Class IRI="#Student"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#attends"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#is_invited"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#prepares"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#Graduation_course"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#Name"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#Salary"/>
```

```
</Declaration>
<Declaration>
    <NamedIndividual IRI="#Ceremony_Class_of_2022"/>
</Declaration>
<Declaration>
    <NamedIndividual IRI="#Introduction_to_Algorithms"/>
</Declaration>
<Declaration>
    <NamedIndividual IRI="#Professor_P0812"/>
</Declaration>
<Declaration>
    <NamedIndividual IRI="#Student_N0194"/>
</Declaration>
<Declaration>
    <NamedIndividual IRI="#Student_N0195"/>
</Declaration>
<SubClassOf>
    <Class IRI="#Administrators"/>
    <Class IRI="#Employee_of_the_University"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Course"/>
    <Class IRI="#Event"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Employee_of_the_University"/>
    <Class IRI="#Person"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Graduates"/>
    <Class IRI="#Student"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Graduates"/>
    <ObjectMaxCardinality cardinality="1">
        <ObjectProperty IRI="#attends"/>
        <Class IRI="#Graduation_Ceremony"/>
```

```
    </ObjectMaxCardinality>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Graduation_Ceremony"/>
    <Class IRI="#Event"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Professor"/>
    <Class IRI="#Employee_of_the_University"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Professor"/>
    <ObjectMinCardinality cardinality="1">
        <ObjectProperty IRI="#attends"/>
        <Class IRI="#Course"/>
    </ObjectMinCardinality>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Student"/>
    <Class IRI="#Person"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Student"/>
    <ObjectSomeValuesFrom>
        <ObjectProperty IRI="#attends"/>
        <Class IRI="#Course"/>
    </ObjectSomeValuesFrom>
</SubClassOf>
<DisjointClasses>
    <Class IRI="#Course"/>
    <Class IRI="#Graduation_Ceremony"/>
</DisjointClasses>
<ClassAssertion>
    <Class IRI="#Graduation_Ceremony"/>
    <NamedIndividual IRI="#Ceremony_Class_of_2022"/>
</ClassAssertion>
<ClassAssertion>
```

```
    <Class IRI="#Course"/>
    <NamedIndividual IRI="#Introduction_to_Algorithms"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="#Professor"/>
    <NamedIndividual IRI="#Professor_P0812"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="#Student"/>
    <NamedIndividual IRI="#Student_N0194"/>
</ClassAssertion>
<ClassAssertion>
    <Class IRI="#Graduates"/>
    <NamedIndividual IRI="#Student_N0195"/>
</ClassAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="#prepares"/>
    <NamedIndividual IRI="#Professor_P0812"/>
    <NamedIndividual IRI="#Introduction_to_Algorithms"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="#attends"/>
    <NamedIndividual IRI="#Student_N0194"/>
    <NamedIndividual IRI="#Introduction_to_Algorithms"/>
</ObjectPropertyAssertion>
<ObjectPropertyAssertion>
    <ObjectProperty IRI="#attends"/>
    <NamedIndividual IRI="#Student_N0195"/>
    <NamedIndividual IRI="#Ceremony_Class_of_2022"/>
</ObjectPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#Name"/>
    <NamedIndividual IRI="#Professor_P0812"/>
    <Literal>Bob</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#Salary"/>
```

```
    <NamedIndividual IRI="#Professor_P0812"/>
    <Literal datatypeIRI="http://www.w3.org/2001/XMLSchema#double">5000.0</Literal
</DataPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#Graduation_course"/>
    <NamedIndividual IRI="#Student_N0194"/>
    <Literal>Computer Science</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#Name"/>
    <NamedIndividual IRI="#Student_N0194"/>
    <Literal>Alice</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#Graduation_course"/>
    <NamedIndividual IRI="#Student_N0195"/>
    <Literal>Visual Arts</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#Name"/>
    <NamedIndividual IRI="#Student_N0195"/>
    <Literal>Claude</Literal>
</DataPropertyAssertion>
<AsymmetricObjectProperty>
    <ObjectProperty IRI="#attends"/>
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
    <ObjectProperty IRI="#is_invited"/>
</AsymmetricObjectProperty>
<AsymmetricObjectProperty>
    <ObjectProperty IRI="#prepares"/>
</AsymmetricObjectProperty>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#attends"/>
    <Class IRI="#Student"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
```

```
    <ObjectProperty IRI="#is_invited"/>
    <Class IRI="#Employee_of_the_University"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#prepares"/>
    <Class IRI="#Professor"/>
</ObjectPropertyDomain>
<ObjectPropertyRange>
    <ObjectProperty IRI="#is_invited"/>
    <Class IRI="#Graduation_Ceremony"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#prepares"/>
    <Class IRI="#Course"/>
</ObjectPropertyRange>
<DataPropertyDomain>
    <DataProperty IRI="#Graduation_course"/>
    <Class IRI="#Student"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#Name"/>
    <Class IRI="#Person"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#Salary"/>
    <Class IRI="#Employee_of_the_University"/>
</DataPropertyDomain>
<DataPropertyRange>
    <DataProperty IRI="#Graduation_course"/>
    <Datatype abbreviatedIRI="xsd:string"/>
</DataPropertyRange>
<DataPropertyRange>
    <DataProperty IRI="#Name"/>
    <Datatype abbreviatedIRI="xsd:string"/>
</DataPropertyRange>
<DataPropertyRange>
    <DataProperty IRI="#Salary"/>
```

```
        <Datatype abbreviatedIRI="xsd:double"/>
    </DataPropertyRange>
</Ontology>


<!-- Generated by the OWL API (version 4.5.9.2019-02-01T07:24:44Z)

https://github.com/owlcs/owlapi -->
```