Relato de Experiência: Utilização de Ontologias Para Reestruturação de Serviços de Suporte e Desenvolvimento de Software

Junilson Pereira Souza (junilson@gmail.com)

Programa de Pós-Graduação em Gestão e Organização do Conhecimento (PPGGOC)
Escola de Ciência da Informação (ECI)
Universidade Federal de Minas Gerais (UFMG)

Resumo

Objetos de estudo. Processo de manutenção de software e respectivo sistema de gestão de demandas de uma empresa de desenvolvimento de software. **Problemas**. Dificuldade em organizar o fluxo de trabalho, falhas na comunicação, falta de padronização e dificuldade em gerar informações gerenciais. **Objetivo**. Criar uma ontologia que sirva de base para revisar a estrutura de uma ferramenta computacional de suporte ao processo de manutenção de software. **Justificativa**. A padronização dos tipos de demandas, relacionamentos e atributos permitirá uma melhor comunicação e gestão de demandas e melhoria nos serviços, permitindo trazer maior satisfação para as partes interessadas e maior sustentabilidade empresarial.

1. Introdução

A manutenção de produtos de software representa uma parcela significativa dos esforços e custos para empresas que provêm produtos de software como serviços. Nesse sentido, a utilização de ferramentas computacionais para gerenciamento de todo fluxo de trabalho torna-se praticamente uma obrigatoriedade, trazendo à reboque um desafio considerável que é a necessidade de adequar tais ferramentas ao contexto organizacional. Em alguns cenários, é comum a utilização de ferramentas chamadas *issue trackers* para o propósito pretendido. Uma característica comum dessas ferramentas é uma grande capacidade de personalização por meio do uso de recursos tais como campos e fluxos personalizados.

A personalização da ferramenta feita de maneira não estruturada pode acarretar vários problemas dentre os quais podem ser citados: a dificuldade de

comunicação em função de uso de terminologia inconsistente, a dificuldade em gerenciar e executar o trabalho de várias partes interessadas em função de um fluxo de trabalho que não reflita o entendimento comum. Tais problemas podem levar a um aumento do esforço, custo e prazo na entrega das demandas, além de impactar na qualidade dos produtos e serviços prestados, levando a insatisfação das partes interessadas.

O objetivo deste trabalho é criar um projeto de ontologia para mapear um processo de manutenção de software que possa ser implementado em uma ferramenta computacional, endereçando os problemas relatados pelas diversas partes interessadas.

Nas seções seguintes são apresentados um breve resumo dos conceitos associados ao processo de manutenção de software, além do conceito de ontologia e projetos dessa natureza. Em seguida, são detalhados os procedimentos metodológicos utilizados, seguidos dos resultados e respectivas discussões. Ao final, são feitas considerações sobre os resultados obtidos e sinalizações para a continuidade e implantação da ontologia.

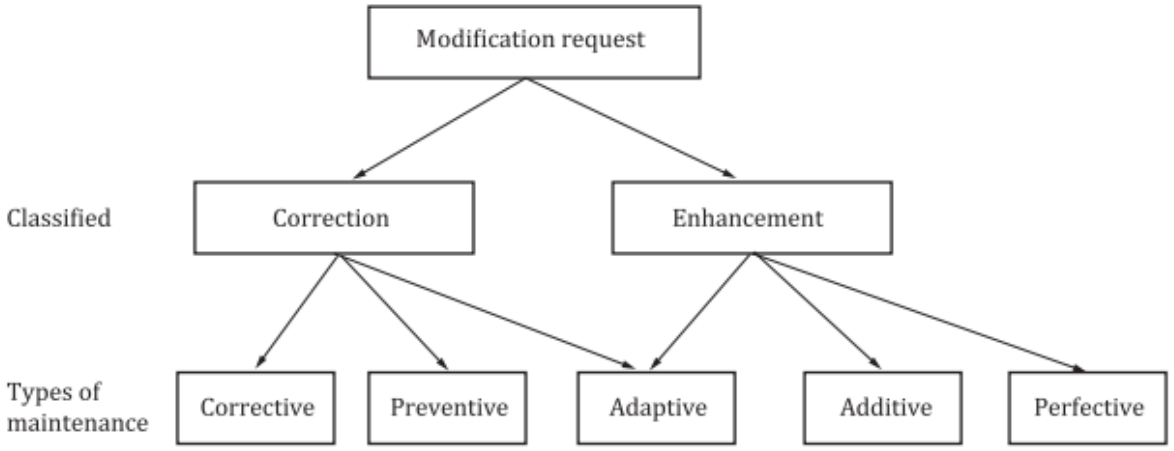
2. Referencial Bibliográfico

Esta seção apresenta os dois principais pilares conceituais explorados no trabalho, que são os processos de manutenção de software e as ontologias.

2.1. Manutenção de software

A manutenção de produtos de software representa uma parcela significativa dos esforços e custos para empresas que provêm produtos de software como serviços. Dentre as várias referências para a realização do processo de manutenção, destaca-se a norma ISO/IEC/IEEE 14764:2022, *Software engineering - Software life cycle processes - Maintenance* (ISO, 2022). A norma propõe uma estruturação das requisições de modificação conforme mostrado na figura 1, com os respectivos tipos de manutenção detalhados na sequência.

Figura 1 - Tipos de requisições de mudança



Fonte: (ISO, 2022).

Manutenção corretiva: modificação de um produto de software realizada após a entrega para corrigir problemas detectados. A modificação visa reparar o produto de software para atender aos requisitos de sistema definidos.

Manutenção preventiva: modificação de um produto de software após a entrega para corrigir falhas latentes no software antes que elas ocorram no sistema em produção.

Manutenção adaptativa: modificação de um produto de software, realizada após a entrega, para mantê-lo utilizável em um ambiente alterado ou em constante mudança. A manutenção adaptativa fornece melhorias necessárias para acomodar as mudanças no ambiente em que o software opera. Essas mudanças ajudam a acompanhar a evolução do ambiente; por exemplo, uma atualização do sistema operacional resulta em alterações nos aplicativos.

Manutenção aditiva: modificação de um produto de software realizada após a entrega para adicionar funcionalidades ou recursos que aprimorem o uso do produto. O tipo de manutenção aditiva se distingue do tipo de manutenção "perfectiva" e é reconhecido como um tipo de manutenção diferente por ser capaz de: — fornecer novas funções ou recursos adicionais para melhorar a usabilidade, o

desempenho, a manutenibilidade ou outros atributos do software no futuro; — adicionar funcionalidades ou recursos com adições ou alterações relativamente grandes no software para melhorar os atributos do software após a entrega, com oportunidades identificadas para negociar quaisquer adições ou alterações na estratégia, métodos, recursos, acordos ou níveis de serviço de manutenção entre fornecedores e adquirentes.

Manutenção aperfeiçoativa (*perfective*): modificação de um produto de software para proporcionar melhorias para os usuários, aprimoramento das informações para os usuários e registro de dados para melhorar o desempenho, a capacidade de manutenção ou outros atributos do software.

Os processos técnicos são usados para definir os requisitos de um sistema de software, transformar esses requisitos em um produto eficaz, permitir a reprodução consistente do produto quando necessário, usar o produto para fornecer os serviços requeridos, manter a prestação desses serviços e descartar o produto quando ele for desativado. Os processos técnicos definem as atividades que permitem que as funções da organização e do projeto otimizem os benefícios e reduzam os riscos decorrentes de decisões e ações técnicas. Essas atividades permitem que os sistemas e serviços de software possuam a pontualidade e disponibilidade, custo-benefício, funcionalidade, confiabilidade, manutenibilidade, capacidade de produção, usabilidade e outras qualidades exigidas pelas organizações compradoras e fornecedoras. Elas também permitem que os produtos e serviços estejam em conformidade com as expectativas ou requisitos legais da sociedade, incluindo fatores de saúde, segurança, proteção e meio ambiente. Os processos técnicos consistem no seguinte (ISO, 2022):

a) Processo de Análise de Negócios ou Missão: Este processo define o propósito, escopo e ambiente do sistema para garantir que as soluções propostas satisfaçam os objetivos de negócio ou missão da organização.

b) Processo de Definição de Necessidades e Requisitos das Partes Interessadas: Define as necessidades de todas as partes interessadas (incluindo usuários e organizações de manutenção) e as transforma em requisitos que afetam a manutenibilidade do sistema.

c) Processo de Definição de Requisitos de Sistema/Software: Define exaustiva e inequivocamente os requisitos do sistema e/ou software, incluindo

funções, interfaces, desempenho e manutenibilidade, com base nas necessidades das partes interessadas.

d) Processo de Definição de Arquitetura: Transforma os requisitos em uma arquitetura de alto nível, definindo componentes de software e sua estrutura, com foco em considerações que simplificam a manutenção, como a separação de preocupações.

e) Processo de Definição de Projeto: Cria um projeto detalhado do software que enfatiza a manutenibilidade, rastreabilidade e compartimentalização de funções, especialmente ao lidar com erros e segurança.

f) Processo de Análise de Sistemas: Utilizado para realizar avaliações e decisões de troca (trade-offs) para garantir que a solução seja acessível, operável e sustentável ao longo do ciclo de vida.

g) Processo de Implementação: Desenvolve, documenta e testa itens de software e bancos de dados, aplicando a escolha da linguagem, estrutura e padrões de codificação para maximizar a manutenibilidade.

h) Processo de Integração: Combina os elementos de software e hardware desenvolvidos para formar o sistema completo, garantindo que as interfaces funcionem conforme o projetado.

i) Processo de Verificação: Ajuda a confirmar que o software modificado ou novo atende aos seus requisitos e que não introduziu novos defeitos, usando testes, como o de regressão.

j) Processo de Transição: Sequência controlada de ações para mover o software de um ambiente para outro ou do desenvolvedor para a manutenção, envolvendo transferência de código, dados e treinamento.

k) Processo de Validação: Confirma objetivamente que o sistema ou software pode atingir seu uso pretendido, metas e objetivos, assegurando que o produto final satisfaz as necessidades reais do cliente.

l) Processo de Operação: Utiliza o produto para fornecer os serviços necessários, monitorando o sistema e gerando relatórios de incidentes ou problemas (MRs/PRs) que podem acionar a manutenção.

m) Processo de Manutenção: Tem como propósito sustentar a capacidade do sistema de fornecer um serviço, tomando ações corretivas, adaptativas, perfectivas e preventivas no software implantado.

n) Processo de Descarte: Estabelecido para o fim da vida útil do software, envolve a análise da descontinuação do suporte ativo, o arquivamento de registros e a destruição segura dos elementos do sistema

2.2. Ontologias

O avanço das tecnologias computacionais em geral e dos mecanismos de gerenciamento de dados em particular, têm sido insuficientes para garantir um uso pleno e efetivo por todas as partes interessadas, especialmente quando são consideradas as dimensões de acessibilidade, interoperabilidade, compartilhamento e reuso. Em determinadas situações, a diversidade de tecnologias adotadas pelos especialistas pode ser um fator complicador. Além disso, apesar de ter sido experimentada uma grande expansão na capacidade de memória e processamento, os computadores ainda demonstram não serem inteligentes; tanto por não entenderem a si mesmos, nem a forma como são programados ou a interpretação pretendida (Arp; Smith; Spear, 2015).

Há ainda outros obstáculos para acesso, seleção e uso dos dados, tais como: (a) idiossincrasia humana: soluções pensadas apenas para legitimar uma percepção e não com base em uma compreensão objetiva e isenta da realidade; (b) torre de Babel: multiplicidade de significados, vocabulários e interpretações; (c) entrada e saída em sistemas: a baixa qualidade dos dados de entrada em um sistema geram resultados geralmente insatisfatórios, impactando inclusive na capacidade computacional de análise; (d) solipsismo em sistemas: quando a análise é baseada apenas em intuição (Almeida, 2021).

Nesse contexto, um elemento que pode servir como parte da solução é a ontologia, que pode ser entendida em dois contextos: um filosófico e outro tecnológico. Historicamente, o conceito remonta a Grécia Antiga, tendo em Aristóteles seu precursor. O termo em si foi adotado em obras da teoria escolástica (Almeida, 2021). No contexto filosófico, conforme (Arp; Smith; Spear, 2015), “a ontologia tem sido tradicionalmente definida como a teoria do que existe (ou do ‘ser enquanto ser’): o estudo dos tipos de entidades na realidade e das relações que essas entidades mantêm entre si”.

Nos últimos anos, o termo “ontologia” tem se tornado recorrente nas ciências da computação e informação. Nessa perspectiva, uma conceituação importante foi

feita por Tom Gruber em 1993, que descreve “ontologia como uma descrição, como uma especificação formal de um programa, dos conceitos e relações que podem existir para um agente ou uma comunidade de agentes” (Gruber, 1992). Uma associação importante de um ontologia com sua realização prática é dada por Almeida (2021), destacando que ontologia é um “artefato formal, para fins de representação da informação e do conhecimento, que têm como vantagens a formulação rigorosa - livre de ambiguidades - de definições, assim como a possibilidade de implementação computacional”.

2.3. Projetos de Ontologias

Para que a ontologia possa adquirir um caráter prático e aplicado, é importante que sejam estruturados projetos que deem suporte a sua implementação. O principal objetivo de um projeto é orquestrar boas práticas tanto conceituais quanto terminológicas, de forma a conseguir empreendimentos de sucesso. Soma-se aos princípios, o uso de metodologias adequadas para desenvolvimento, tais como a OntoForInfoScience que, por sua vez, reutiliza etapas oriundas de outras já existentes como a Methontology, o 101 Method e a NeOn (Almeida, 2021). Neste trabalho, foi utilizada a metodologia Pronto (Almeida, 2024).

2.3.1. Princípios e boas práticas de projeto

Projetos de ontologia devem ser construídos a partir de referências conceituais sólidas, inerentes à origem e propósito da área. Nesse sentido, torna-se necessário fazer uso de uma série de princípios que possam nortear os trabalhos desenvolvidos. A tabela 1 apresenta princípios teórico-complementares que devem ser seguidos em projetos de ontologia.

Tabela 1 - Princípios teórico-complementares para projetos de ontologia

Princípio	**Descrição**
Realismo	O objetivo de uma ontologia é descrever a realidade. O realismo é a posição filosófica segundo a qual a realidade e seus constituintes existem independentemente de nossas visões (linguísticas, conceituais, teóricas ou culturais).
Perspectivismo	Este princípio reconhece que a realidade é complexa e variada. O Perspectivismo afirma que existem várias descrições igualmente precisas da realidade.
Falibilismo	O Falibilismo sustenta que as ontologias, assim como as teorias científicas que representam, são falíveis.
Adequatismo	Defende que as entidades em um domínio específico devem ser levadas a sério nos seus próprios termos e não devem ser vistas como redutíveis a outros tipos de entidades mais simples.
Reutilização	Ontologias já existentes devem ser tratadas como referências (benchmarks) e reutilizadas sempre que possível ao construir ontologias para novos domínios.
Utilidade	Não é razoável sacrificar princípios realistas por considerações de utilidade no curto prazo.
Atualização	As ontologias científicas estão sempre sujeitas à atualização a partir de avanços no conhecimento.
Facilidade	Ao projetar uma ontologia de domínio, o desenvolvedor deve começar identificando as características do assunto que são mais fáceis e claras de entender e definir.

Fonte: (Almeida, 2021).

Além dos princípios teórico-complementares faz-se necessário atentar para questões que impactarão cada etapa do projeto de ontologias, incluindo aspectos como seleção, definição e formatação de termos, além da criação de taxonomias. De forma a garantir a consistência e qualidade do projeto de ontologia, faz-se necessário seguir um conjunto de princípios, como os apresentados na tabela 2.

Tabela 2 - Princípios conceituais e terminológicos

Categoria	**Princípio**
Seleção de termos	1. Incluir termos usados por grupos de cientistas ou profissionais influentes para representar universais.
Seleção de termos	2. Buscar consenso de cientistas ou profissionais para o uso de termos, o que envolve a discussão entre grupos de especialistas.
Seleção de termos	3. Identificar regiões de sobreposição interdisciplinar e verificar onde o uso dos termos não é consistente.
Seleção de termos	4. Manter tanto quanto possível os termos usados por especialistas, fazendo uso de recursos existentes.
Formatação de termos	5. Usar substantivos singulares.
Formatação de termos	6. Usar minúsculas para substantivos comuns.
Formatação de termos	7. Evitar acrônimos.
Formatação de termos	8. Associar cada termo a um identificador alfanumérico único.
Formatação de termos	9. Garantir a univocidade dos termos.
Formatação de termos	10. Garantir a univocidade das expressões de relacionamentos.
Formatação de termos	11. Evitar substantivos incontáveis.
Definições de termos	12. Fornecer todos os termos com definições, exceto o raiz.
Definições de termos	13. Usar definições no formato aristotélico (S é um G que D).
Definições de termos	14. Usar características essenciais para definir termos.
Definições de termos	15. Iniciar pelos termos mais gerais do domínio.
Definições de termos	16. Evitar circularidade ao definir termos.

Definições de termos	17. Ao definir, usar termos mais simples que o termo sob definição.
Definições de termos	18. Evitar criar termos para universais por combinação lógica.
Criação de taxonomias	19. Fundamentação ontológica e classificação por características.
Criação de taxonomias	20. Estrutura: níveis, é-um consistente e herança única.
Criação de taxonomias	21. Disjunção: entidades no mesmo nível são disjuntas.
Criação de taxonomias	22. Exaustividade: classificar tanto quanto possível do domínio.
Criação de taxonomias	23. Rigor conceitual: não adotar termos ambíguos.
Criação de taxonomias	24. Uniformidade: critérios de classificação consistentes.
Criação de taxonomias	25. Clareza e precisão: não adotar termos ou categorias vagas.
Criação de taxonomias	26. Meta-categorias: não adotar as que dependam de si mesmas.

Fonte: (Almeida, 2021).

2.3.2. Método para desenvolvimento

Além dos princípios elencados, um projeto de ontologia efetivo e de qualidade, deve seguir uma estrutura metodológica em que as etapas e respectivos passos sirvam de guia para os trabalhos rumo ao sucesso do empreendimento. A tabela 3 apresenta a metodologia PRONTO (Almeida, 2024).

Tabela 3 - Etapas e passos da metodologia PRONTO

Etapa	**Passos**
1. Especificação	1.1. Estabelecer questões de competência. 1.2. Estabelecer um conjunto de metadados. 1.3. Produzir o documento de especificação.
2. Aquisição de conhecimento	2.1. Extrair termos de outros artefatos. 2.2. Extrair termos de outros documentos. 2.3. Extrair termos elicitados de especialistas. 2.4. Registrar termos obtidos. 2.5. Criar modelo preliminar.
3. Organização de termos representativos de entidades	3.1. Importar termos genéricos. 3.2. Inserir termos de domínio. 3.3. Organizar taxonomia preliminar.
4. Organização de termos representativos de relacionamentos	4.1. Selecionar e inserir relacionamentos padrão. 4.2. Criar e inserir relacionamentos adicionais.
5. Definição de entidades	5.1. Aplicar método de definições. 5.2. Registrar definições em linguagem natural. 5.3. Registrar conjunto de metadados.
6. Formalização	6.1. Produzir axiomas. 6.2. Classificar a ontologia.
7. Avaliação	7.1. Avaliar para ajustes. 7.2. Avaliar para resultados.
8. Documentação	8.1. Reunir material usado. 8.2. Criar um documento resumo da ontologia.

Fonte: (Almeida, 2024).

A seção a seguir apresenta os procedimentos metodológicos adotados com base no uso da metodologia selecionada.

3. Procedimentos metodológicos

O projeto de criação da ontologia ocorreu no contexto de uma empresa de médio porte e com histórico de atuação de mais de 30 anos. A organização está ligada ao ramo de desenvolvimento de soluções de software cujo principal serviço é a oferta de um produto para um domínio específico. A sede da empresa fica na cidade de Belo Horizonte, Minas Gerais. A partir deste ponto, a empresa será referenciada como Empresa XPTO.

O projeto de ontologia nasceu dentro de outra iniciativa que é o projeto de melhoria dos processos de software e serviços de tecnologia da informação da Empresa XPTO, baseados nos modelos de referência de software (MR-SW) e serviços (MR-SV), constituintes do programa de Melhoria do Processo de Software Brasileiro (MPS-BR) e promovidos pela Softex (https://softex.br).

Os trabalhos específicos ligados à ontologia tiveram uma duração de 30 dias, considerando o prazo decorrido entre a identificação do problema e a execução de todos os passos metodológicos propostos para a obtenção da ontologia. Para coleta de informações, utilizou-se como base o conhecimento previamente adquirido durante o projeto de melhoria, além de acesso a documentação técnica de processos e procedimentos, além da ferramenta Redmine (https://redmine.org), em uma versão específica instalada nos domínios da Empresa XPTO. Participaram diretamente da iniciativa, o autor deste trabalho, no papel de consultor em melhoria de processos e a responsável pelo grupo de processos da Empresa XPTO, além de outros profissionais entrevistados sob demanda.

Para o desenvolvimento das etapas da metodologia foram utilizados o Google Workspace (https://workspace.google.com) para a criação de arquivos texto e planilhas com informações gerais, o Protégé (https://Protege.stanford.edu) para a modelagem da ontologia e o GitHub (https://github.com) como repositório para controle de versões dos artefatos criados.

As seções a seguir apresentam os principais produtos resultantes da execução das etapas da metodologia proposta.

3.1. Especificação

As principais informações geradas nesta etapa estão associadas ao propósito do projeto, o domínio da ontologia e os respectivos requisitos, conforme apresentado na figura 2.

Figura 2 - Documento de Especificação de Ontologia

Propósito

- Criar e manter a ontologia dos serviços e processos de manutenção de software.

Escopo

- Serviços de suporte e desenvolvimento.
- Trata-se de uma ontologia com carater informal.

Linguagem de implementação

OWL 2.

Usuários e usos pretendidos

- Gestores de processo: manter a ontologia e garantir implementação no Redmine.
- Analistas executores do processo de manutenção de software: validar e aplicar a ontologia.

Requisitos de ontologia

Requisitos não funcionais

- Permitir a implementação na ferramenta de gestão de demandas (Redmine).

Requisitos funcionais (questões de competência)

- Qual é o tipo de atividade de manutenção (Replace, Repair, Inspect, etc.) de uma solicitação específica?
- Qual é o estado atual de uma tarefa de manutenção e suas transições permitidas?
- Quais são as dependências de precedência entre tarefas de manutenção?
- Que itens (equipamentos/componentes) participam de uma atividade de manutenção?
- Uma solicitação de manutenção é preventiva ou corretiva?
- Quais são as saídas esperadas de uma atividade de manutenção específica?
- Que procedimento de manutenção descreve como executar uma determinada atividade?

Fonte: autor - com base no exemplo de (Almeida, 2024).

Além do documento de especificação, também é esperado nesta etapa que sejam definidos os metadados. Para o projeto, foram priorizados os seguintes metadados: definicao, fonteDefinicao e responsavelDefinicao.

3.2. Aquisição de conhecimento

Os artefatos de representação foram obtidos por meio das definições de solicitações de mudança e atividades técnicas providas pela norma ISO/IEC/IEEE 14764:2022 (ISO, 2022).

Os termos originalmente utilizados foram extraídos das classificações de tipos disponíveis no Redmine da empresa. Foi feita a criação de um projeto específico no Protégé para fins de visualização da estrutura original.

Para registro de termos de artefatos de representação foi feita uma busca inicial superficial sobre ontologias específicas relacionadas ao tema, tendo sido identificada a *Representation Formalism for Software Engineering Ontologies* (REFSENO), que serviu de referência para o entendimento de "subontologias" (RUIZ et al., 2011).

Para aquisição do conhecimento de especialistas, foram feitas entrevistas com os profissionais responsáveis pelo processo de software da empresa para validar a implementação original, disponível no Redmine.

Os termos compilados foram registrados em uma planilha, disponível no repositório do projeto. Para fins de simplificação, foi criado apenas um modelo da versão atual.

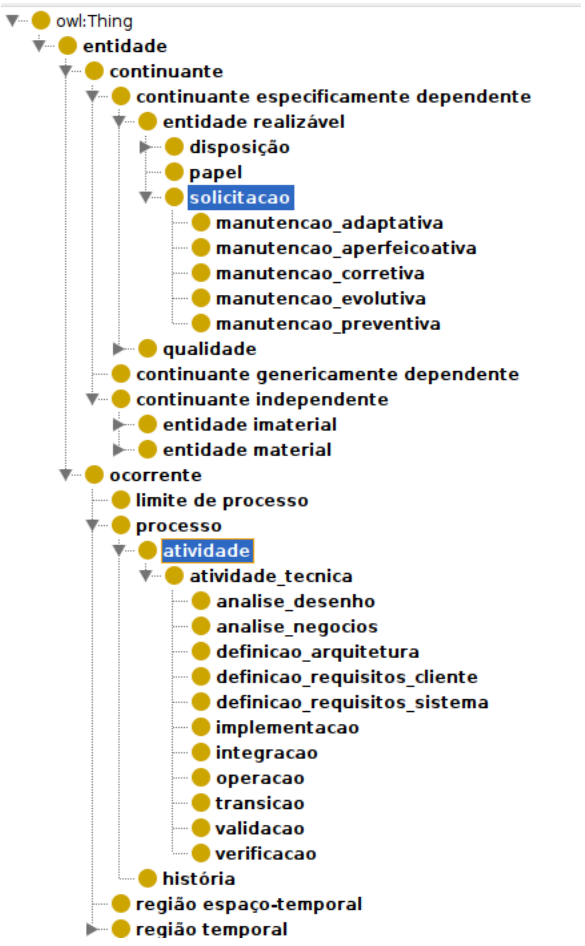
3.3. Organização de termos representativos de entidades

Os termos genéricos foram incorporados ao projeto por meio da importação da Basic Formal Ontology (BFO) para o Protégé, disponível em: https://github.com/NCOR-BR/Framework-BFO/blob/main/BFO/bfo-2020%20PT.owl.

3.4. Organização de termos representativos de relacionamentos

Após a importação tanto da BFO quanto da planilha com as classes identificadas na etapa anterior, foi feita uma estruturação no Protégé, conforme mostrado na figura 5. Foram utilizados os relacionamentos padrão providos pela BFO. Para fins de simplificação, não foram incluídos relacionamentos adicionais.

Figura 5 - Organização das classes no Protégé



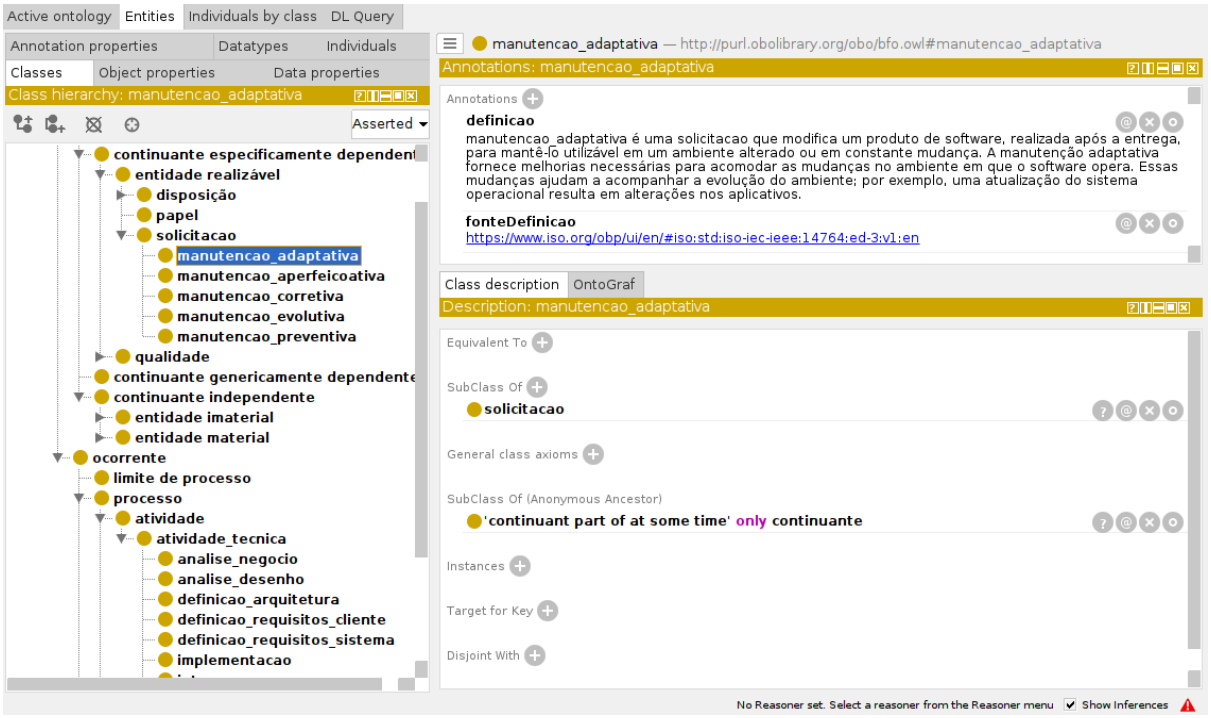
Fonte: autor.

3.5. Definição de entidades

Foram criadas definições para todas entidades, conforme o método aristotélico, que segue a seguinte estrutura: "S é um G que D", onde S - representa a espécie, G - representa o gênero e D - representa o diferencial da espécie.

Em seguida, as definições em linguagem natural foram registradas na view Annotations, metadados definicao e fonteDefinicao do Protégé, conforme exemplo ilustrativo apresentado na figura 6. Para fins de simplificação, nesta primeira versão do projeto, não foram adicionados outros metadados.

Figura 6 - Exemplo de definição de entidade no Protégé



Fonte: autor

3.6. Formalização

Foram produzidos axiomas para explicitar as dependências entre atividades com a descrição seguindo a sintaxe manchester:

```
analise_negocio precedes some definicao_requisitos_cliente

definicao_requisitos_cliente precedes some definicao_requisitos_sistema

definicao_requisitos_sistema precedes some definicao_arquitetura

definicao_arquitetura precedes some analise_desenho

analise_desenho precedes some implementacao

implementacao precedes some integracao

integracao precedes some verificacao

verificacao precedes some transicao

transicao precedes some validacao

validacao precedes some operacao
```

Na execução do reasoner, foram identificadas inconsistências. Entretanto, considerando o escopo deste projeto, decidiu-se por não tratar.

3.7. Avaliação

Dadas as limitações de prazo e conhecimento no uso do Protégé, não foram realizadas avaliações com relação aos resultados apresentados pela ferramenta, da mesma forma que não foram feitas avaliações dos resultados nem realizadas consultas que respondessem às questões de competência.

3.8. Documentação

Os artefatos produzidos neste projeto estão no repositório do GitHub (https://github.com/valuedriven/software-main-ontology). Para fins de simplificação, não foi criado nenhum mecanismo para publicação e acesso ao conteúdo da ontologia, além dos artefatos disponíveis no repositório e que podem ser acessados por meio do Protégé.

4. Resultados

Um importante resultado obtido no desenvolvimento do projeto foi a possibilidade de se organizar de maneira sistemática as entidades que, inicialmente, estavam alocadas de maneira ad-hoc, dificultando o entendimento dos relacionamentos e hierarquias de classes.

Outro resultado importante foi a definição rigorosa de cada entidade, o que forçou a busca de literatura especializada, permitindo o alinhamento e consenso sobre a diversidade de assuntos existentes em iniciativas dessa natureza.

Foi possível ainda, por meio da execução do projeto, corrigir inconsistências significativas na implementação original no Redmine, como a existência de entidade “Outros”, a eliminação de algumas ocorrências de herança múltipla entre entidades, além da eliminação de entidades consideradas “fantasmas”, que não traziam nenhum ganho para a implementação original.

5. Discussão

O desenvolvimento do projeto de ontologia reforça um aspecto central prometido em iniciativas dessa natureza e já assinalados nos resultados que foi a

padronização da terminologia empregada. Os ganhos obtidos com esse resultado vão além de apenas “usar termos comuns aos envolvidos” mas possibilita um melhor entendimento do processo de manutenção de software.

Um fator que potencializou o desenvolvimento do projeto foi a experiência e conhecimento do autor em projetos de melhoria de processos de software, além do conhecimento do contexto empresarial.

Dentre os aspectos que dificultaram o desenvolvimento do projeto, encontram-se a inexperiência do autor em iniciativas dessa natureza. Dentre as etapas desenvolvidas, a que mais foi afetada por essa limitação foi a avaliação. A despeito dessa limitação, os resultados ora obtidos são potencialmente aplicáveis na implementação das mudanças na ferramenta.

Outra limitação deveu-se ao escopo inicial proposto. Como foi feito um foco apenas no processo de manutenção de software, algumas atividades relevantes para a sustentação do produto de software ficaram de fora, tais como a gestão de serviços de tecnologia da informação, o que pode ser complementado nas ações futuras por meio da inclusão de referências como as normas ISO relativas ao assunto e o framework ITIL.

6. Considerações finais

Com base nos resultados e discussões apresentados, pode ser considerado que este trabalho atingiu o objetivo inicialmente traçado que foi criar um projeto de ontologia para mapear um processo de manutenção de software que pudesse ser implementado em uma ferramenta computacional, endereçando os problemas relatados pelas diversas partes interessadas.

Como proposta de ações que podem ser desenvolvidas a partir deste trabalho, destacam-se: a) realizar os passos não realizados propostos pela

metodologia PRONTO, de forma a permitir a implementação plena da ontologia na ferramenta Redmine; b) realizar uma avaliação mais robusta, especialmente a partir da revisão e implementação de consultas relativas às questões de competência; c) usar o projeto de ontologia como base para realizar análise de impacto da migração de versão do Redmine em função da elevada quantidade de campos personalizados; d) incluir agentes, artefatos e o produto de software como parte da modelagem da ontologia; e) criar um documento resumo da ontologia provendo uma forma de consulta rápida a ontologia, utilizando, por exemplo, a ferramenta Widoco; f) efetivar a implementação da ontologia na ferramenta Redmine.

Referências

ALMEIDA, Mauricio Barcellos. **Ontologia na prática – projeto, metodologia e construção**. Curitiba, PR: Editora CRV, 2024. v. 01

ALMEIDA, Maurício Barcelos. **Ontologia em ciência da informação: teoria e método : coleção representação do conhecimento em ciência da informação**. 1ª ed. *[S.l.]*: CRV, 2021. v. 1

ARP, Robert; SMITH, Barry; SPEAR, Andrew D. **Building Ontologies with Basic Formal Ontology**. Cambridge, Massachusetts: The MIT Press, 2015.

GRUBER, Tom. **What is an Ontology?** Disponível em: <http://www-ksl.stanford.edu/kst/what-is-an-ontology.html>. Acesso em: 24 nov. 2025.

ISO. **ISO/IEC/IEEE 14764:2022(en), Software engineering — Software life cycle processes — Maintenance**. Disponível em: <https://www.iso.org/obp/ui/en/#iso:std:iso-iec-ieee:14764:ed-3:v1:en>. Acesso em: 27 nov. 2025.

RUIZ, FRANCISCO *et al.* AN ONTOLOGY FOR THE MANAGEMENT OF SOFTWARE MAINTENANCE PROJECTS. **International Journal of Software Engineering and Knowledge Engineering**, 21 nov. 2011.