Uma Visão Ontológica Sobre os Testes de Software Utilizando Ontologia de Fundamentação-A (UFO-A)

Lidvaldo José dos Santos[1], Sildenir Alves Ribeiro[1,2], Eber Assis Schmitz[1], Mônica Ferreira da Silva[1]

[1]Instituto Tércio Pacitti
Universidade Federal do Rio de Janeiro (UFRJ) – Rio de Janeiro, RJ – Brazil

[2]Centro Federal de Educação Tecnológica do Rio de Janeiro – Rio de Janeiro, RJ – Brazil

lidiosantos@tic.ufrj.br,
sildenir.ribeiro@ppgi.ufrj.br{eber,monica}@nce.ufrj.br

Abstract. *This paper presents a fragment of the ontological model of the domain of software testing domain, represented by the main objects, relations, phases and artifacts produced As a result, two complementary conceptual approaches of the elements involved in Software Testing are presented, based on the Foundational Ontology (UFO-A), as a basis, in order to provide a clearer understanding of the conceptual modeling that exists in the testing process.*

Resumo. *Este artigo apresenta um fragmento de modelo ontológico do domínio de testes de software, representado pelos principais objetos, papéis, relações, fases e artefatos produzidos. Posteriormente, são apresentadas duas abordagens conceituais complementares dos elementos envolvidos no teste de software, utilizando como base a Ontologia de Fundamentação-A (UFO-A), com o objetivo de fornecer uma compreensão mais clara da modelagem conceitual existente no processo de testes.*

1. Introdução

O teste de *software* é uma subárea da engenharia de *software* e consiste em uma operação dinâmica de casos de teste, com o objetivo de avaliar o comportamento de um *software* previamente definido (França and Marins, 2018).

Embora alguns processos de desenvolvimento de *software* concentrem-se em sua análise final, hoje em dia, é amplamente compreendido que a execução do teste é uma parte do processo de verificação e validação necessária para avaliar e manter a qualidade de um produto de *software* (Young and Pezze, 2008).

As ontologias de fundamentação surgiram no intuito de fornecer um vocabulário para a representação do conhecimento, de modo a evitar interpretações ambíguas ou lacunas semânticas. Quando aplicadas ao processo de testes de *software*, estas ontologias podem auxiliar o especialista de testes na compreensão do processo, de modo a obter um melhor proveito das interpretações conceituais da ontologia elaborada (Souza, 2011).

Este artigo tem por objetivo representar uma abordagem complementar de um modelo ontológico na fase de teste de *software*, fornecendo uma compreensão mais ampla das terminologias utilizadas. Na seção 2, é feito um breve resumo dos trabalhos relacionados às ontologias de teste de *software*. Na seção 3, são apresentados os conceitos de ontologia e um exemplo de mapeamento de domínio de teste de *software*. Na seção 4 é apresentada uma abordagem complementar dos principais elementos e papéis existentes na fase de teste de

software. Por último, a seção 5 contém as considerações finais com orientações para trabalhos futuros.

2 Trabalhos Relacionados

Diversos estudos foram realizados com o propósito de modelar o domínio de teste de *software*. Falbo (2017) apresenta uma ontologia de referência em teste de *software* denominada ROoST: *Reference Ontology on Software Testing*. Com foco nos processos de teste, propõe quatro sub-ontologias: processo e atividades de teste, artefatos de teste, técnicas de teste e equipes e ambiente de teste.

A partir de uma revisão sistemática de literatura, Falbo and Souza (2013) buscam investigar ontologias no domínio de teste de *software* que possuam uma boa abrangência de domínio, seguindo um método de avaliação e reutilizando ontologias fundamentais.

Tebes et al. (2020) desenvolveu uma arquitetura ontológica para representação dos testes de *software* e a partir de uma revisão sistemática de literatura, foi possível discutir aspectos relacionados à ambiguidade e incompletude de testes, bem como atividades e métodos de trabalho nas ontologias selecionadas.

Com o intuito de fornecer uma compreensão mais ampla sobre a web semântica, Dadkhah et al. (2020) identifica 52 estudos primários a partir de uma revisão sistemática de literatura, fornecendo suporte as atividades de teste de *software*.

Guizzardi and Falbo (2008) abordam versões de dois fragmentos da UFO, definidas como UFO-B e UFO-C. Demonstram a utilização da ontologia de fundamentação na avaliação e fornecimento de semântica do mundo real para uma ontologia no domínio de engenharia de *software* e discutem sua relevância por meio de um estudo de caso no domínio de processos de *software*.

Com base em um estudo de caso, Sun et al. (2020) elabora a ontologia de um processo completo de teste de *software*, abrangendo objetos, métodos e casos de teste. Com base nos resultados apresentados, esta ontologia tem por objetivo contribuir no compartilhamento e disseminação do conhecimento de domínio, além da melhoria do processo e qualidade do teste de *software*.

Todas as ontologias citadas nesta seção contribuem para a organização do conhecimento. No domínio de teste de *software,* a ontologia ROoST é mais completa do que as outras ontologias propostas. O modelo proposto por esse artigo contribui com a representação do domínio do teste de *software*, possibilitando uma melhor interpretação das tarefas de teste apresentada em ROoST.

3. Conceitos Gerais

As Ontologias de Fundamentação constituem sistemas de categorias bem fundamentadas e independentes de domínio, que tem sido utilizada com sucesso na melhoria da qualidade dos modelos conceituais e linguagens de modelagem (Guizzardi and Falbo, 2008).

3.1 Ontologia

A Ontologia tem a sua origem na Filosofia, e, é o estudo dos tipos de coisas que existem, trata, também, da natureza do ser e das questões metafísicas em geral. Segundo Guizzardi (2005), na Ciência da Computação, ontologia é um instrumento conceitual bastante útil, principalmente nas áreas de modelagem de dados e inteligência artificial; para Souza (2018), ontologia é uma especificação explícita e formal de uma conceituação compartilhada. Com base neste conteúdo, as ontologias podem ser classificadas através de: (a) Ontologias Genéricas, a partir da descrição de conceitos gerais como espaço, tempo e ação; (b) Ontologias de domínio, que apresenta um

vocabulário relacionado a um domínio genérico, desenvolvendo conceitos de domínios particulares; (c) Ontologia de Tarefas, que descreve conceitos relativos à atividades genéricas ou tarefas, independente do domínio que ocorram; (d) Ontologias de aplicação, que aborda conceitos que dependem tanto de um domínio específico como de uma tarefa específica.

3.1.1 *Unified Foundational Ontology*

Para Guizzardi e Wagner (2004), a ontologia de fundamentação UFO está relacionada a uma combinação de outras ontologias como *Generalized Formalized Ontology* (GFO), *Descriptive Ontology for Linguistic and Cognitive Engineering* (DOLCE) e de princípios da metodologia OntoClean. A UFO pertence a uma família de ontologias e que subdivide em três especialidades: A UFO-A, UFO-B e UFO-C. Cada especialidade retrata um contexto: a UFO-A é conhecida como *endurants* ou de objetos, a UFO-B como perdurants ou processos e a UFO-C representada pelas entidades sociais, sendo uma junção das duas anteriores.

3.1.2 UFO-A

A UFO-A pode ser caracterizada por uma ontologia que pretende fornecer um maior nível semântico para a modelagem conceitual de um dado domínio de conhecimento. Esta aborda questões como as noções de tipos e suas instâncias; objetos e suas propriedades intrínsecas e a relação entre identidade e classificação; distinções entre tipos e suas relações; relações parte-todo, além de classes, propriedades, relacionamentos e regras (Guizzardi, 2005).

3.2. Mapeamento do Domínio

3.2.1 Exemplo de Modelo Ontológico em Testes de Software

A Figura 1 extraída do artigo de Souza et al. (2017), ilustra um modelo de ontologia de tarefas, representando testes e técnicas desta fase.

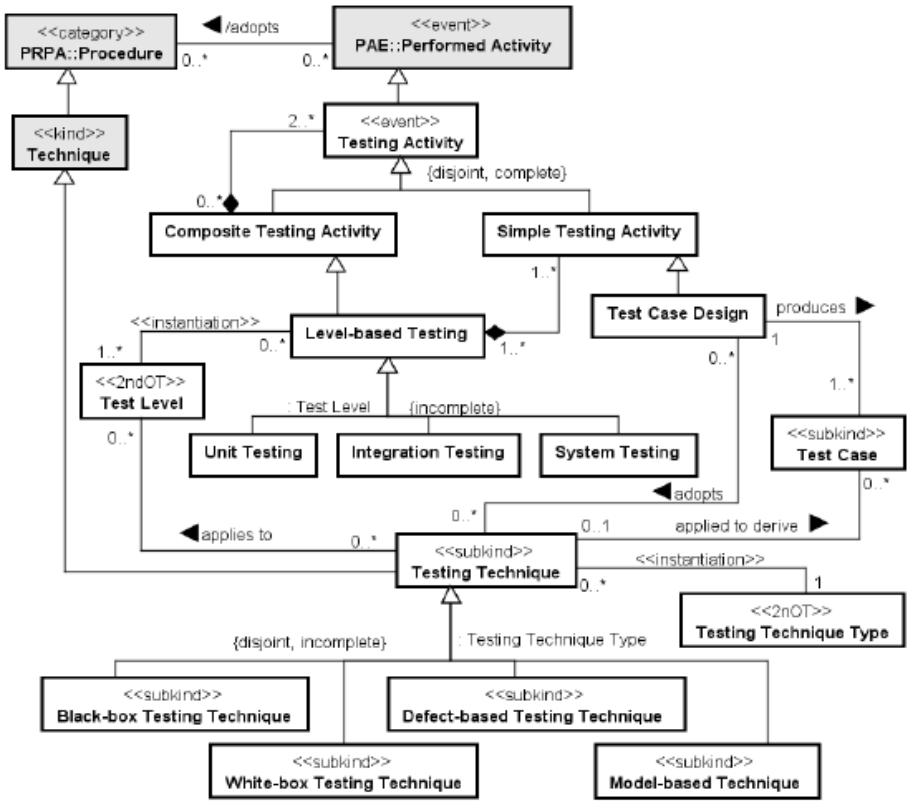

Figura 1. Sub-Ontologia - Técnicas e Testes - *ROoST´s.* Souza et al. (2017)

A sub-ontologia ROoST´s apresenta as etapas do processo de testes, com seus respectivos elementos e relacionamentos.

Na próxima seção, serão descritos os principais elementos do domínio da fase de teste, de forma a auxiliar na compreensão do modelo e do contexto semântico.

4. Representação dos Elementos da Ontologia do Domínio de Teste

4.1 Principais elementos e papéis

Com base na ontologia de domínio proposta por Souza et al. (2018), a Figura 2 apresenta uma abordagem específica dos principais elementos e papéis existentes na fase de teste de *software*:

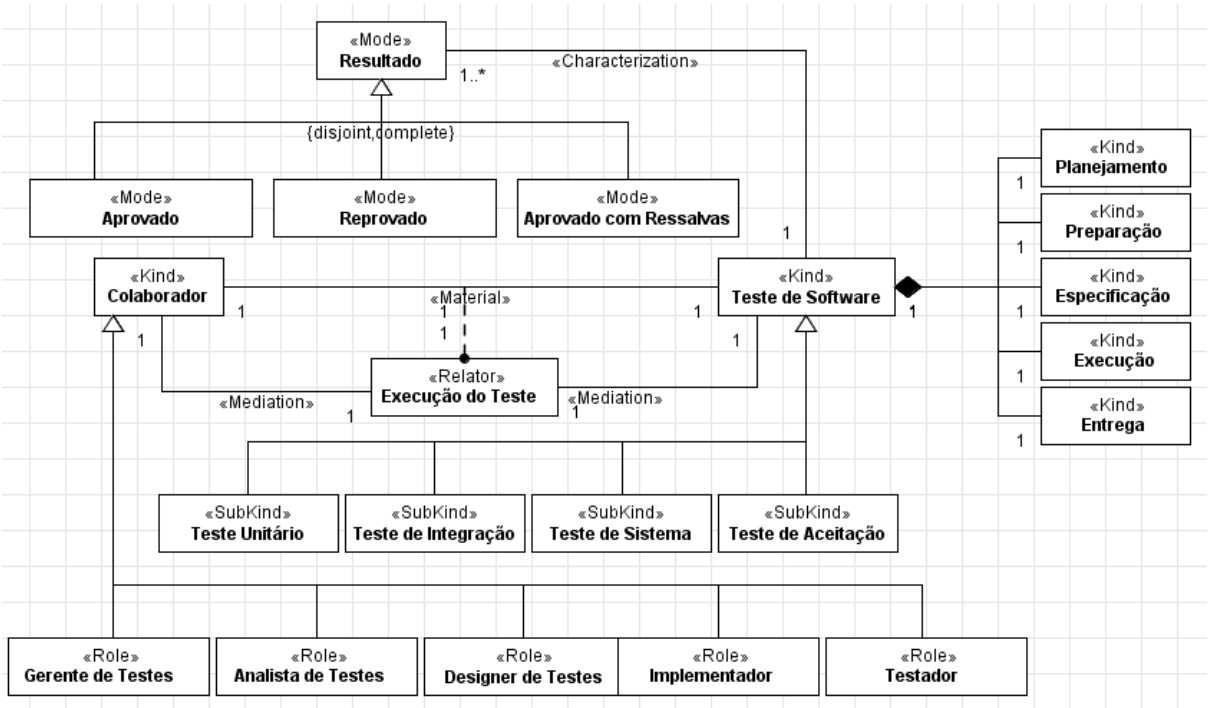

Figura 2. Modelagem - Representação Específica da Ontologia do Domínio de Teste

Na Figura 2 é apresentada uma ontologia proposta na UFO-A, criada com o objetivo de detalhar os elementos identificados no processo de testes de *software*.

Conforme mencionado em seções anteriores, a modelagem ontológica através da UFO permite definir e representar conceitos e relações de qualquer domínio de forma objetiva, com riqueza em detalhes semânticos e sem ambiguidades. A modelagem visa tirar proveito dessa vantagem, estendendo alguns elementos detectados.

A partir da representação acima, a classe *Colaborador* foi modelada como um sortal (*Kind*). Esse estereótipo representa classes cujas instâncias possuem o mesmo princípio de identidade (Ex. nome, cpf). As instâncias são rígidas, pois pertencem ao mesmo tipo, e existencialmente dependentes.

Em seguida, temos a apresentação do estereótipo *Role*, que corresponde a um tipo não-rígido e que define o papel de um *Kind*. Na Figura 2, temos o Gerente de Testes, responsável pelo gerenciamento funcional e operacional da equipe de testes, representado pelo tipo (*Role*). Pode-se dizer que, uma instância de Colaborador pode, em um determinado momento, se tornar

uma instância de Gerente de Testes e depois deixar de ser. Um outro aspecto deste estereótipo é que este representa classes existencialmente dependentes.

Assim como o Gerente de Testes, temos outros elementos representados no modelo que definem o papel do Colaborador. São eles: O Analista de Testes, responsável pela operacionalização do processo de teste; o Designer de Testes, que tem a função de identificar e descrever as técnicas de teste apropriadas; o Implementador, responsável por desenvolver os componentes de acordo com os padrões adotados no teste de *software*; e o Testador, responsável pela execução do teste. Estes papéis, caracterizados pelo estereótipo (*Role*), surgiram a partir de uma propriedade intrínseca presente nestas classes e que foram criadas para distinguir seus diferentes tipos.

Na relação entre as classes Colaborador e Teste de *Software*, existe um estereótipo chamado (*Relator*) associado à classe Execução de Teste. Este é representado no modelo a partir de uma mediação (*Mediation*), entre essas classes. Também neste relacionamento, existe uma relação implícita, sendo um tipo de relação material. Relações materiais precisam de um objeto (*Relator*) para existir e por isso a relação é chamada de material. O objeto que intervém nesta relação é a Execução. A cardinalidade representada na relação entre Colaborador e Teste de *Software* possui uma semântica em que um colaborador é responsável pela execução de um Teste de *Software*.

Pode-se destacar neste modelo que, existe uma relação de composição parte-todo entre a classe Teste de *Software* e suas respectivas etapas, representadas pelo estereótipo (*Kind*). São elas: O planejamento, onde são estruturados o plano e ideias dos testes; A preparação, que tem por objetivo compor o ambiente de teste (pessoal, equipamentos, ferramentas de automação) para que os testes sejam executados conforme planejados; A especificação, responsável pelas atividades de elaboração, revisão dos casos de teste e revisão do roteiro de testes; A execução, onde os testes são executados e os resultados obtidos são registrados; A entrega, sendo a última fase do ciclo de vida de testes, onde o projeto é finalizado e toda documentação é finalizada e arquivada.

Conforme destacado na composição anterior, estas partes apresentam um papel funcional em relação ao todo. Além disso, possuem propriedades como transitividade e anti-simetria. Pode-se dizer também que, possuem propriedades que podem ser compartilhadas e são essenciais para representação do todo, que é o teste de *software*.

Nos testes baseados em níveis, temos subclasses associadas ao plano de testes, e que podem ser representadas pelo estereótipo *(SubKind*). São elas: O Teste de Sistema, que tem por objetivo verificar o sistema como um todo, geralmente verificando os requisitos não-funcionais, tais como velocidade, precisão, segurança e confiabilidade; O teste unitário, por sua vez, representa a etapa inicial dos testes, onde cada unidade do sistema é isolada das demais e testadas isoladamente; O teste de integração, tem por objetivo verificar a interação entre os componentes de *software*; O teste de aceitação, tem como função avaliar se o comportamento do sistema está de acordo com os requisitos do cliente. Cada uma dessas especializações é um subtipo de teste de *software,* que possui dependência e características intrínsecas (próprias). Além disso, essas especializações carregam uma identidade, que é rígida. Com isso, estas especializações foram criadas para distinguir os diferentes tipos de teste de *software*.

Como resultado final dos testes de *software*, existem três classes, derivadas do tipo (*mode*) Resultado, que define se o *software* foi testado ou não de forma correta. Um teste pode estar na fase de Aprovado, caso este seja executado com sucesso e sem erros. Caso seja reprovado, ele terá de ser corrigido para que o mesmo seja realizado novamente pela equipe de testes. No caso de o teste ser aprovado com ressalva, significa que um defeito foi detectado,

porém o mesmo não afeta a funcionalidade do sistema. Observando de forma mais técnica, diz-se que um modo é existencialmente dependente de outros indivíduos. Neste caso, há uma dependência existencial (modo intrínseco), que determina os três resultados possíveis observados no modelo. Existe uma partição na classe Resultado (disjunta e completa); esta depende de um valor, de uma propriedade intrínseca (disjunta). Nesta partição, existirá apenas um modo a ser representado, não admitindo outro comportamento (completo).

4.2 Adaptação Complementar do Fragmento de Ontologia de Testes

Tendo em vista a importância da relação dos principais artefatos e normas existentes na fase de testes, é apresentado na Figura 3, uma adaptação complementar ao fragmento de ontologia proposto por Guizzardi et al. (2008), para representação do processo de testes de *software*.

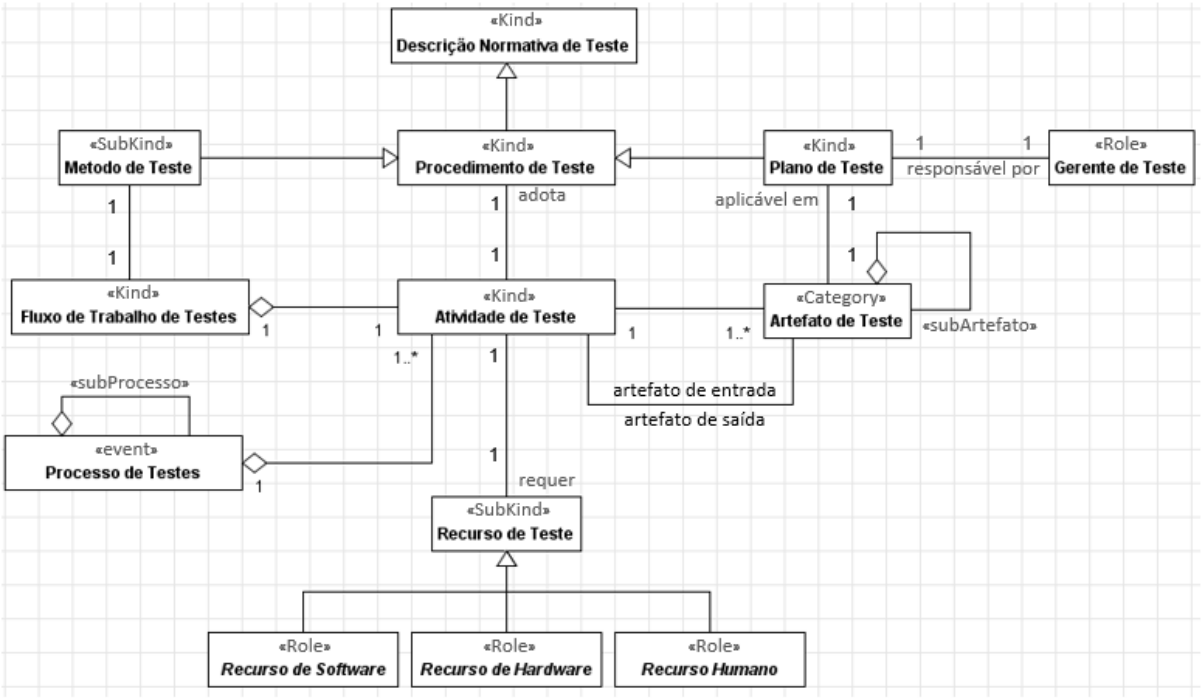

Figura 3. Adaptação complementar de um fragmento de Ontologia proposto por Guizzardi et al. (2008)

Conforme demonstrado na Figura 3, temos como representação o *Processo de Testes*, que é um evento complexo e pode ser decomposto em *Atividade de Teste* ou outros processos, chamados de subprocessos. Um subprocesso é apresentado como uma decomposição de um processo, que satisfaz os critérios para ser um processo, ou seja, possui um propósito, tem coesão e é colocado sob responsabilidade de uma organização, no ciclo de vida de *software*.

Uma *Atividade de Teste*, representada pelo estereótipo (*kind*), é um conjunto de tarefas que podem gerar *Artefatos*. Para ser realizada, esta atividade requer recursos e utiliza artefatos produzidos por outras atividades. Possui três etapas: A preparação do teste, a execução e o registro do teste.

Os *Artefatos de Teste*, representados pelo estereótipo (*Category*), fornecem a entrada e saída para as *Atividades de Teste* e o mecanismo pelo qual as informações são transmitidas entre as atividades. Estes podem ser decompostos em super-artefatos e sub-artefatos. Pode-se destacar, por exemplo, que na Atividade de Teste, existe um artefato de entrada seria a especificação da arquitetura do sistema e como artefato de saída, temos a geração do plano e relatório de integração.

Em complemento, temos um elemento *substancial* que participa de uma ação, representado pelo estereótipo (*subkind*) *Recurso de Teste*. Este pode ser utilizado para apoiar a execução das *Atividades de Teste*. É subdividido em três papéis: (a) *Recursos Humanos*, que descrevem os papéis que agentes humanos devem desempenhar em uma atividade, tais como o Gerente de Testes, o Testador, dentre outros; (b) *Recursos de Hardware* incluem quaisquer equipamentos necessários para a realização de uma atividade, tais como computadores e impressoras; (c) *Recursos de Software*, que se referem a qualquer produto utilizado na realização de uma atividade, tal como um sistema de gerenciamento de banco de dados ou ferramentas que auxiliem na criação do *software*.

Um Procedimento de Teste, conforme apresentado no modelo pelo estereótipo (*kind*), é um tipo de Descrição Normativa de Teste e são empregados na realização de atividades. Por sua vez, uma Descrição Normativa, segundo Bottazzi (2008), é um tipo de substancial inanimado social que define uma ou mais regras/normas reconhecidas por pelo menos um agente social, e que pode definir entidades sociais como Universais. A Descrição Normativa de Testes utilizada neste modelo é representada pela Norma IEEE 829 ou Padrão 829 para documentação de Testes de *Software*. Esta especifica um conjunto de documentos pertinentes às atividades de um projeto de teste e também para a elaboração de um artefato. Os documentos definidos por esta norma cobrem as tarefas de planejamento, especificação e descrição de testes.

No modelo representado acima, destaca-se o *Plano de Teste*, representado pelo estereótipo (*PlanType*), que é o documento responsável por apresentar o planejamento para execução dos testes do *software* em desenvolvimento, incluindo a abrangência, abordagem, recursos e cronograma das atividades de teste. Associado a este Plano, temos a participação do Gerente de Testes, responsável por acionar uma ação, que é o *Plano de Teste*, atuando na defesa da qualidade e dos testes, no planejamento e gerenciamento de recursos e na resolução de problemas.

Um Método de Teste, por sua vez, representado pelo estereótipo (*subkind*), é um procedimento sistemático que especifica um fluxo de atividades para a realização de uma ou mais atividades. Este método pode ser adotado na realização de mais de uma atividade, pois apresenta um fluxo de trabalho para cada uma destas atividades.

Esta seção propôs fornecer uma visão mais detalhada dos principais artefatos e normas existentes no processo de teste de *software*.

5. Considerações Finais

Este artigo apresentou os principais elementos envolvidos no modelo de ontologia de testes, com base na ontologia de fundamentação UFO-A.

A partir do detalhamento de um modelo conceitual, nota-se que a utilização desta representação provê benefícios durante o processo de construção de ontologias, uma vez que elucida os elementos representados no modelo pertencente a um domínio em particular, neste caso representado pela ontologia de testes de software.

Trabalhos futuros tem por objetivo uma representação mais ampla da ontologia de domínio de testes, a partir de experimentos e estudos de caso, de forma a obter um melhor gerenciamento dos testes de *software* e auxílio às equipes responsáveis destas atividades.

Espera-se que os desdobramentos realizados neste artigo sejam utilizados como fundamentação teórica para novos processos de modelagem e sua adequada categorização.

Referências:

Andrade, A. P. and Viana, P. (2012). Criação e Geração de Planos de Teste de Software. Obtido em, 9.

Bertolino, A. (2007, May). Software testing research: Achievements, challenges, dreams. In Future of Software Engineering (FOSE'07) (pp. 85-103). IEEE.

Birrell, N. D. and Ould, M. A. (1988). A practical handbook for software development. Cambridge University Press.

Bottazzi, E. and Ferrario, E. (2008). Towards a DOLCE Ontology of Organizations. Journal of Business Process Integration and Management (IJBPIM), Inderscience Publisher.

Dadkhah, M., Araban, S., & Paydar, S. (2020). A systematic literature review on semantic web enabled software testing. Journal of Systems and Software, 162, 110485.

Falbo, R. D. A., Natali, A. C. C., Mian, P. G., Bertollo, G. and Ruy, F. B. (2003). ODE: Ontology-based software development environment. In IX Congreso Argentino de Ciencias de la Computación.

França, B. B., & Marins, W. F. (2018). Processo de Teste em uma Fábrica de Software Acadêmica: Um Relato de Experiência. Revista Ada Lovelace, 2, 27-33.

Gorayeb, I. L. and Oliveira, S. R. B. (2017). Um relato de melhoria do processo de teste de software aplicado a uma fábrica de software. Revista Traços, 12(26).

Guizzardi, G. (2005). Ontological foundations for structural conceptual models.

Guizzardi, G., Falbo, R. A. and Guizzardi, R. S. (2008). The role of Foundational Ontologies for Domain Ontology Engineering: a case study in the Software Process Domain. IEEE Latin America Transactions, 6(3), 244-251.

Guizzardi, G. and Wagner, G. (2004, June). A Unified Foundational Ontology and some Applications of it in Business Modeling. In CAiSE Workshops (3) (pp. 129-143).

Souza, Érica Ferreira. (2011). Gestão de conhecimento aplicado a teste de software: Uma abordagem baseada em ontologia. INPE/Ministério da Ciência e Tecnologia. http://mtc-m16d.sid.inpe.br/col/sid.inpe.br/mtc-m19/2012/03.05.16.55/doc/P_Erica_Ferreira_Souza_2011.pdf .

Souza, É. F., Falbo, R. A. and Vijaykumar, N. L. (2013). Ontologies in software testing: a systematic. In VI Seminar on Ontology Research in Brazil (p. 71).

Souza, É. F. D., Falbo, R. D. A. and Vijaykumar, N. L. (2017). ROoST: reference ontology on software testing. Applied Ontology, 12(1), 59-90.

Souza, A. O. (2018). Mapeamento ontológico para classificar mensagens significativas em atividades colaborativas (Bachelor's thesis, Universidade Tecnológica Federal do Paraná).

Sun, Z., Hu, C., Li, C. and Wu, L. (2020). Domain Ontology Construction and Evaluation for the Entire Process of Software Testing. IEEE Access, 8, 205374-205385.

Tebes, G., Olsina, L., Peppino, D. and Becker, P. (2020). TestTDO: A Top-Domain Software Testing Ontology. XXIII CIbSE, 20, 1-14.

Young, M. and Pezze, M. (2008). Teste e analise de software. Porto Alegre: Artmed.