Issue procedure ontology (ipo): uma ontologia para sintomas, problemas e soluções

Matheus Dimas de Morais

Professor DIII coordenação de Informática do Instituto Federal Fluminense (IFF). Possui mestrado em Pesquisa Operacional e Inteligência Computacional pela Universidade Candido Mendes em Campos dos Goytacazes - RJ (2015). Especialização em MBA Profissional em Engenharia de Sistemas pela Escola Superior Aberta do Brasil em Vitória - ES (2015). Graduação em Ciências da Computação pela Universidade Candido Mendes em Campos dos Goytacazes – RJ (2011).

Mark Douglas de Azevedo Jacyntho

Professor adjunto da coordenação de Ciência da Computação, da Universidade Candido Mendes em Campos dos Goytacazes – RJ. Professor DIV da coordenação de Informática do Instituto Federal Fluminense (IFF). Possui doutorado em Informática (2012), mestrado em Informática (2002) e graduação em Engenharia de Computação (1998), os três títulos obtidos no Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio).

http://dx.doi.org/10.1590/1981-5344/2446

No cenário atual de intensa competitividade entre as empresas, cada vez mais o uso do tempo se torna algo a ser maximizado, visando maior produtividade. Assim, para as organizações se manterem em plena operação, problemas devem ser solucionados o mais breve possível. A informática pode ajudar nesse processo, oferecendo sistemas computacionais que auxiliem na resolução de tais problemas. Com o advento da Web Semântica, surge

a possibilidade de representar as informações de tal forma que o computador consiga compreendê-las, possibilitando a criação de sistemas inteligentes com o uso de ontologias. A principal contribuição deste trabalho é uma ontologia extensível (core ontology) para o domínio de sintomas, problemas e soluções, denominada Issue Procedure Ontology (IPO). Esta ontologia pretende prover às máquinas a semântica necessária para que estas forneçam não apenas informações, mas, sobretudo, identifiquem problemas a partir de um conjunto de sintomas e, em seguida, sugiram possíveis soluções, de forma autônoma, para os problemas em questão. Ao final, é realizada uma avaliação da ontologia a fim de corroborar seu poder de expressividade.

Palavras-chave: *Ontologia extensível. Sintomas, problemas e soluções. Web semântica.*

Issue procedure ontology (ipo): an ontology for symptoms, problems and solutions

In the current scenario of intense competitiveness among companies, use of time increasingly becomes something to be maximized, aiming at greater productivity. Thus, for organizations remain in full operation, problems should be solved as soon as possible. Information technology can help this process by providing computer systems to assist in troubleshooting. With the advent of the Semantic Web, there is the possibility to represent information in such a way that the computer can understand them, enabling the creation of intelligent systems with the use of ontologies. The main contribution of this work is an extensible ontology (core ontology) for the domain of symptoms, problems and solutions, named Issue Procedure Ontology (IPO). This ontology aims to provide the necessary semantics for machines so that they provide not only information, but, above all, identify problems from a set of symptoms and then autonomously suggest possible solutions for the problems in question. In the end, is performed an evaluation of ontology in order to corroborate its expressiveness.

Keywords: *Core ontology. Symptoms, problems and solutions. Semantic Web.*

Recebido em 26.05.2015 Aceito em 28.07.2016

1 Introdução

No cenário atual de intensa competitividade entre as empresas, cada vez mais o uso do tempo se torna algo a ser maximizado, visando maior produtividade. Assim, as organizações devem alocar seus recursos e gerenciá-los de forma adequada, para que a força de trabalho tenha plena condição de realizar suas tarefas com maior eficiência (MAGALHÃES; PINHEIRO, 2007; REZENDE, 2002).

A área de Tecnologia da Informação (TI) tem sido de suma importância para as organizações, uma vez que os sistemas computacionais podem agilizar o trabalho feito por seus colaboradores, servindo como meio para as organizações alcançarem seus objetivos. Hoje, para muitas empresas, a TI se tornou parceira estratégica, fazendo parte do negócio da empresa (PEREIRA; DE SOUZA; DA COSTA, 2012).

Assim, os serviços de TI podem ajudar diversos setores das empresas a solucionar seus problemas. Um problema geralmente possui sintomas que indicam sua existência e uma ou mais possíveis soluções, meios que se consiga sanar (ou amenizar) o problema, eliminando (ou reduzindo), assim, os sintomas.

O processo de (i) analisar os sintomas, (ii) identificar o problema e (iii) localizar a solução, depende da ação humana do profissional, que pode errar em seu diagnóstico, devido sua falta de experiência ou atenção. Desta forma, sistemas computacionais podem ser utilizados para facilitar e auxiliar o profissional nessa importante tarefa.

Com o advento da *Web* Semântica ou *Web* de Dados Ligados (*Web of Linked Data*), surge a possibilidade de representar as informações de maneira que o computador consiga compreendê-las (BERNERS-LEE; HENDLER; LASSILA, 2001).

Para que os dados sejam compreendidos pelos computadores, se faz necessário que os mesmos sejam estruturados de forma padronizada e que seja dado sentido semântico explícito a eles. Para associar semântica aos dados, o consórcio W3C[1] propõe a utilização de ontologias ou vocabulários. Ontologia é uma representação formal de um determinado domínio de conhecimento, onde conceitos (classes) deste domínio são explicitamente definidos, bem como suas propriedades e relacionamentos (JACYNTHO, 2012). São as ontologias que dão poder de inferência às máquinas, possibilitando, com uso de raciocinadores (*softwares* que fazem uso dos axiomas ontológicos para inferir novos dados), a descoberta de novos conhecimentos (LICHTNOW; DE OLIVEIRA, 2009; PICKLER, 2007).

Visando prover uma representação semântica para o domínio de sintomas, problemas e soluções, foi projetada uma ontologia extensível (*core ontology*) denominada *Issue Procedure Ontology* (IPO) que pretende prover às máquinas a semântica necessária para que estas forneçam não

[1] Disponível em: <http://www.w3.org/>. Acesso em: 15 maio 2015.

apenas informações, mas, sobretudo, identifiquem problemas a partir de um conjunto de sintomas e, em seguida, sugiram possíveis soluções, de forma autônoma, para os problemas em questão. A intenção é prover uma ontologia genérica, reutilizável e extensível para o domínio de sintomas-problemas-soluções. Uma ontologia que pode ser usada diretamente, mas que também pode ser estendida ou especializada para tipos de problemas mais específicos.

Este artigo tem como objetivo apresentar a ontologia IPO. Para tal, foi organizado da seguinte forma: seção 2 expõe, brevemente, o referencial teórico sobre ontologia e o domínio de problemas; na seção 3 é apresentada a ontologia em si, abordando os passos metodológicos e suas classes e propriedades; seção 4 apresenta um exemplo de instanciação da ontologia; para complementar, na seção 5, é demonstrado um procedimento, com o detalhamento de todos os seus passos (*workflow*); já a seção 6 é dedicada a uma avaliação da ontologia, com base nos exemplos anteriores, mostrando que a mesma permite, de fato, responder as questões de competências que motivaram a sua criação; finalmente, na seção 7, este trabalho é concluído com as observações finais e trabalhos futuros.

2 Referencial teórico

2.1 Web semântica e ontologias

A *Web* Semântica pretende evoluir a *Web* atual, conhecida como "*Web* de documentos", que tem como principal objetivo disponibilizar informações para as pessoas, para a "*Web* de dados" que visa disponibilizar os dados para que as máquinas possam manipulá-los de modo mais eficiente, transformando a *Web* em um grande banco de dados. Assim, a *Web* Semântica não é uma *Web* separada, mas uma extensão da atual, na qual a informação é publicada com um significado explícito e estruturado, permitindo melhor interação entre máquinas e pessoas (BERNERS-LEE; HENDLER; LASSILA, 2001).

Pode-se entender a *Web* Semântica como um conjunto de tecnologias e padrões que visam possibilitar que as máquinas entendam o significado, ou a semântica, das informações publicadas na *Web* e ontologia é uma delas (YU, 2011).

Uma ontologia representa formalmente um determinado domínio de conhecimento, definindo seus principais conceitos ou classes e relacionamentos que representam hierarquias (superclasses e subclasses). Outros relacionamentos também são encontrados nas ontologias, definidos por meio de propriedades que descrevem características ou atributos das classes, e também propriedades que relacionam termos com outros termos. Assim, uma ontologia codifica o conhecimento de um determinado domínio em um formato estruturado que possibilita a máquina entender esse conhecimento, viabilizando a *Web* Semântica (JACYNTHO, 2012).

Segundo Noy e McGuinness (2001), uma ontologia representa um determinado domínio de conhecimento e consiste em: classes, que representam os principais conceitos do domínio; propriedades que representam relacionamentos e atributos das classes; axiomas, que são regras de restrição sobre o domínio e instâncias das classes.

Com os dados publicados na *Web* de forma estruturada e anotados semanticamente com ontologias, a máquina passa a compreender estes dados e, com o ferramental semântico descrito na ontologia, deduzir novos dados de maneira autônoma, gerando conhecimento.

2.2 Sintomas, problemas e soluções

A ontologia desenvolvida nesse trabalho tem como objetivo representar o domínio de sintomas, problemas e soluções. Por ser uma *core ontology*, ou seja, uma ontologia genérica para quaisquer tipos de problemas, há que se compreender como são percebidos esses elementos (sintomas, problemas e soluções) em diversos contextos.

Segundo o dicionário Michaelis (2009), um problema é uma questão levantada para inquirição, consideração, discussão, decisão ou solução, e acrescenta que um problema é um tema cuja solução ou decisão requer considerável meditação ou habilidade.

A definição de problema está intimamente relacionada com o contexto em que se aplica. Assim, no contexto da administração, um problema é indicado por alguma frustração, irritação, percepção de diferença entre a situação ideal e a real e perspectivas de prejuízos. Entende-se que um problema gera sempre uma decisão, ou seja, a partir da definição do problema, temos que buscar alternativas, ou soluções, para resolvê-lo (MAXIMIANO, 2004).

Ao se analisar o contexto de TI, o modelo ITIL v3 (CARTLIDGE et al., 2007) define um problema como uma causa não conhecida de um ou mais incidentes, ou seja, um incidente sinaliza um ou mais problemas. Um incidente pode ser considerado, dentro deste contexto, como uma interrupção não planejada ou redução na qualidade de um serviço de TI. Assim, com base nos incidentes (sinais da ocorrência de um problema) é identificado o problema que os causa. Após a identificação do problema, é aplicada uma solução, geralmente em forma de procedimentos (workflow).

Já na medicina, essa distinção fica bem clara, onde os problemas são identificados como doenças. Assim, um paciente reporta alguns sintomas, ou sinais, que indicam a existência de uma doença. O médico, a partir dos sintomas, identifica a doença que está causando os sintomas. Uma vez diagnosticada a doença, o paciente é submetido ao tratamento adequando. Um tratamento pode ser representado por uma sequência de passos como um *workflow* (MAYO CLINIC STAFF, 2013).

Com base nas abordagens de problemas nos contextos citados acima, conclui-se que um problema é algo ou alguém que não está no seu estado normal ou ideal. Quando algo ou alguém não se apresenta em seu

estado ideal, sinais ou sintomas podem ocorrer evidenciando a existência do problema. Além disso, após a descoberta da causa dos sintomas, ou seja, a descoberta do problema em si, inicia-se o procedimento para solucionar o mesmo, onde, caso já exista uma solução previamente conhecida e catalogada, basta aplicá-la. Contudo, caso o problema seja desconhecido, é necessário estudar o problema visando o entendimento do mesmo para, posteriormente, elaborar e testar uma ou mais soluções. Um processo muitas vezes demorado e oneroso, cuja solução encontrada, obviamente, deve ser cuidadosamente catalogada para outras eventuais ocorrências futuras.

3 IPO: ontologia para sintomas, problemas e soluções

Para desenvolver a ontologia descrita nesse trabalho, foi utilizada a metodologia *Ontology Development 101*, descrita por Noy e McGuinness (2001). Essa metodologia foi escolhida pela simplicidade e iteratividade de suas etapas, a saber:

1) determine o domínio e o escopo da ontologia;
2) considere o reuso de ontologias existentes;
3) enumere os termos importantes da ontologia;
4) defina as classes e a hierarquia de classes;
5) defina as propriedades das classes;
6) defina as restrições;
7) crie instâncias.

Conforme mencionado anteriormente, a ontologia *Issue Procedure Ontology* (IPO) pretende representar o domínio de problemas, relacionando sintomas e soluções. Visando especificar o escopo e o domínio da ontologia, foram considerados inicialmente problemas comuns em alguns cenários motivacionais, dentre os quais, setor de suporte de TI de uma empresa (*help desk*), diagnóstico de doenças por um médico, identificação de defeitos na linha de produção de uma indústria e problemas no *design* de um software. Após analisá-los, foram elencadas algumas questões de competência para as quais a ontologia deveria prover as respostas. Estas questões servem para definir o escopo da ontologia, bem como avaliar sua expressividade. São elas:

QC1. Qual é a categoria de um problema/sintoma/solução?
QC2. Quais são sintomas de um problema?
QC3. Quais são as soluções de um problema?
QC4. Qual o causador do problema?
QC5. Qual o hospedeiro do problema?
QC6. Quem criou/registrou esse problema/sintoma/solução?
QC7. Quais ações (*workflow*) a serem tomadas para solucionar o problema?
QC8. Um problema causa outro problema?

QC9. Um problema depende de outro problema?
QC10. Quais os possíveis problemas, a partir de um conjunto de sintomas?
QC11. Em quais possíveis classes (de problema) uma ocorrência no espaço/tempo pode ser classificada, a partir dos sintomas identificados?

A fim de se aplicar a diretriz de reuso de ontologias existentes, foi feito um levantamento das ontologias *Linked Data* relacionadas no indexador *Linked Open Vocabularies (LOV)*[2], onde verificou-se que nenhuma ontologia se propunha a descrever o domínio proposto com a expressividade necessária para responder as questões levantadas acima. Porém a IPO estende e/ou reusa termos de outras ontologias, a saber: *Friend of a Friend (FOAF)* (BRICKLEY; MILLER, 2014) - para descrição de pessoas e organizações; *Dublin Core* (DCMI USAGE BOARD, 2012) - para descrição de recursos gerais; *Simple Knowledge Organization System (SKOS)* (MILES; BECHHOFER, 2009) - para categorização de problemas por meio de um tesauro.

A ontologia foi construída usando-se a linguagem de ontologias *Web Ontology Language* (OWL 2.0), recomendada pelo W3C. Segundo Hitzler *et al.* (2012), OWL é uma linguagem projetada para representar o conhecimento sobre as "coisas", grupos de "coisas", e as relações entre as "coisas", com grande poder de expressividade.

Para a ontologia criada, foi adotado o prefixo *ipo* para representar o *namespace* http://purl.org/ipo/core#. As especificações técnicas em RDF[3] de todos os termos da ontologia IPO foram publicadas na *Web* e podem ser encontradas no endereço http://purl.org/ipo/core.

O modelo conceitual da ontologia IPO pode ser visto na Figura 1~~Figura 1~~, onde são apresentados seus conceitos (classes) e suas relações (propriedades), por meio de um diagrama de classes UML[4].

Formatado: Cor da fonte: Preto

[2] Disponível em: <http://lov.okfn.org/>. Acesso em: 14 maio 2015.
[3] *Resource Description Framework* (RDF) é o modelo de dados em grafo padrão da Web Semântica. Maiores detalhes podem ser obtidos em Cyganiak, Wood e Lanthaler (2014, p. 1).
[4] Disponível em: <http://www.uml.org/>. Acesso em: 15 maio 2015.

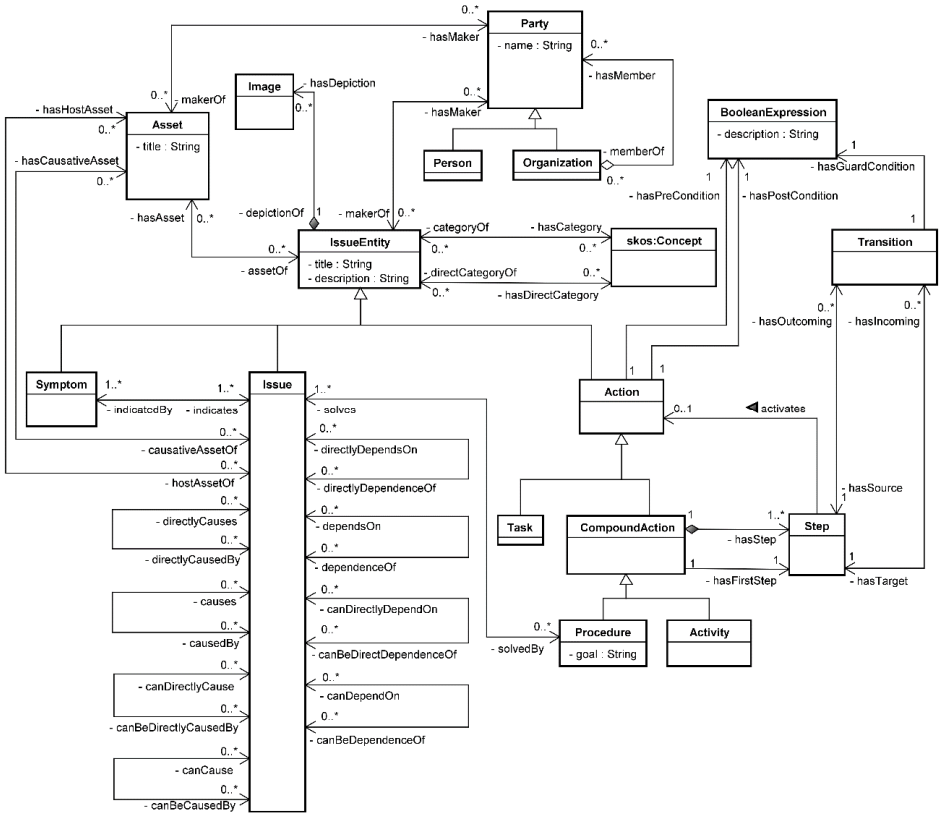

Figura 1 - Modelo conceitual da ontologia IPO

Fonte: Os autores (2015).

A seguir serão descritas as classes e as propriedades da ontologia IPO. Para tal, seguindo uma abordagem *top-down*, a ontologia foi dividida em quatro módulos, a saber:

IssueEntity - Super Tipo Raiz
Classes: *IssueEntity, Asset, Image, skos:Concept*.
Propriedades: *categoryOf, hasCategory, directCategoryOf, hasDirectCategory, hasMaker, makerOf, hasAsset, assetOf, hasDepiction, depictionOf, title, description*.

Pessoas e Organizações
Classes: *Party, Person e Organization.*
Propriedades: *hasMember, memberOf, name*.

Problemas e Sintomas
Classes: *Symptom, Issue*.
Propriedades: *directlyCauses, directlyCausedBy, causes, causedBy, dependenceOf, dependsOn, directDependenceOf, directlyDependsOn, canDirectlyCause, canBeDirectlyCausedBy,*

canCause, canBeCausedBy, canBeDependenceOf, canDependOn, canBeDirectDependenceOf, canDirectlyDependOn, indicatedBy, indicates, hasCausativeAsset, causativeAssetOf, hasHostAsset, hostAssetOf.

Problemas e Soluções
Classes: *Action, Task, CompoundAction, Procedure, Activity, Step, Transition, BooleanExpression*.
Propriedades: solves, solvedBy, hasFirstStep, hasStep, activates, hasSource, hasTarget, hasIncoming, hasOutcoming, hasGuardCondition, hasPostCondition, hasPreCondition, goal.

3.1 IssueEntity - super tipo raiz

As classes e propriedades deste módulo visam descrever diversos relacionamentos que são comuns aos três principais termos da ontologia: Sintoma, Problema e Ação (Solução).

Classe: *IssueEntity*
Superconceito (supertipo) que reúne as características comuns aos três principais conceitos dentro do domínio da ontologia: Sintoma (*Symptom*), Problema (*Issue*) e Ação (*Action*).

Classe: *Asset*
Qualquer "coisa" de valor relacionada a uma *IssueEntity*[5]. Por exemplo, um problema (doença) diagnosticado em um paciente, pode ter um vírus e o paciente como *Asset*, pois o vírus é o agente causador do problema e o paciente é o hospedeiro no qual o problema se manifesta.
Um *Asset* pode ser qualquer "coisa": uma pessoa, um objeto, um relatório, um documento, etc.

Classe: *Image*
Um artefato que ilustra ou registra uma percepção visual.
Ela pode ser usada para ilustrar uma *IssueEntity* visando uma melhor compreensão dela.

Classe: *skos:Concept*
A ontologia IPO reusa a ontologia SKOS para definir esquemas de classificação, ou seja, um conjunto de categorias (ou conceitos) relacionadas (hierarquicamente e de outras formas) formando um tesauro, sob as quais instâncias da classe *IssueEntity* podem ser agrupadas.
A classe *Concept* da ontologia SKOS possui propriedades que permitem a criação de hierarquias de categorias (*Concepts*), permitindo

[5] Como a classe IssueEntity já foi descrita anteriormente, será utilizada o próprio nome da classe, em inglês, para citá-la no texto. Para os elementos ainda não descritos, serão citados em português e entre parênteses o nome da classe em inglês.

ainda expressar transitividade entre as categorias. Essa abordagem deve ser utilizada como uma alternativa para classificação por subclasse de *IssueEntity*, quando se tratar de classificações não intrínsecas, apenas de agrupamento. Por exemplo, no domínio da medicina, podemos agrupar as doenças como doenças virais, doenças bacterianas, etc.

É importante ressaltar que existe outra forma de classificação, por meio da criação de subclasses de *IssueEntity*. O uso de subclasses leva a uma capacidade de inferência mais refinada e deve ser utilizado quando se tratar de classificação intrínseca de tipo/subtipo, onde novas classes, com novas restrições, precisam ser criadas para descrição de tipos de problema mais específicos de um contexto particular.

Propriedades: *hasCategory* e *categoryOf*
Uma *IssueEntity* pode ser agrupada em várias categorias, utilizando um esquema de classificação por meio de um tesauro de categorias (ou conceitos). A propriedade *hasCategory* relaciona uma *IssueEntity* com sua(s) categoria(s) (*skos:Concept*). *categoryOf* é propriedade inversa de *hasCategory*.

Propriedades: *hasDirectCategory* e *directCategoryOf*
hasDirectCategory é subpropriedade de *hasCategory* e indica uma categoria à qual a *IssueEntity* está diretamente relacionada. *directCategoryOf* é subpropriedade de *categoryOf* e propriedade inversa de *hasDirectCategory*.

Propriedades: *hasAsset* e *assetOf*
hasAsset relaciona uma *IssueEntity* com um *Asset*. A propriedade *assetOf* é propriedade inversa de *hasAsset*. *assetOf* pode ser utilizada para facilitar a recuperação de registros de problemas relacionados ao *Asset*.

Propriedades: *hasDepiction* e *depictionOf*
hasDepiction relaciona uma *IssueEntity* com uma *Image*, visando uma melhor descrição. *depictionOf* é inversa de *hasDepiction* e é funcional, ou seja, uma *Image* está relacionada por essa propriedade com no máximo uma única *IssueEntity*.

Propriedades: *hasMaker* e *makerOf*
hasMaker associa uma *IssueEntity* com uma Pessoa ou Organização (*Party*) que a criou ou registrou. Essa propriedade também é utilizada pela classe *Asset* para relacionar seu fabricante, desenvolvedor, inventor, etc. *makerOf* é propriedade inversa de *hasMaker*.

Propriedade: *description*
Uma descrição textual que descreve algo com mais detalhes.

Propriedade: *title*
Título (palavra ou frase) que resumidamente descreve algo.

3.2 Pessoas e organizações

Esse módulo provê termos para descrever pessoas e organizações, permitindo a representação de funcionários e colaboradores de empresas, definição de departamentos, entre outros.

Classe: *Party*
Superconceito (supertipo) comum aos conceitos Pessoa (*Person*) e Organização (*Organization*), e que assume um papel de agente (*Party*) dentro do domínio abordado. Um agente é uma pessoa ou organização atuante, ou seja, capaz de realizar algo.

Classe: *Person*
Esta classe representa pessoas. Um exemplo seria uma pessoa que trabalha em uma organização e produz algum *Asset* ou registra uma *IssueEntity*.

Classe: *Organization*
Representa um grupo de pessoas organizadas visando um objetivo em comum: social, comercial ou político. Por exemplo, uma empresa ou um departamento de uma empresa.

Propriedades: *hasMember* e *memberOf*
A propriedade *hasMember* relaciona uma *Organization* com seus membros, que são instâncias de *Party*, podendo ser *Persons* ou outras *Organizations*. Por exemplo, esta propriedade poderia ser usada para registrar que um funcionário é membro de um departamento. Uma outra abordagem é utilizar essa propriedade para relacionar duas organizações, representado, por exemplo, um departamento que é membro de sua empresa. *memberOf* é propriedade inversa de *hasMember*.

Propriedade: *name*
Nome para identificação de algo.

3.3 Problemas e sintomas

Esse é o módulo principal, pois contém termos que representam sintomas e problemas (*Symptom, Issue*) e os relacionam, permitindo que a máquina deduza, a partir dos sintomas, os possíveis problemas.

Classe: *Issue*
Um problema ou questão a ser resolvida. Por exemplo, algo que não está operando normalmente ou um impedimento para realização de alguma tarefa.
Um *Issue* pode ser causa e/ou causado, direta ou indiretamente, por outro *Issue*, da mesma forma que o *Issue* A pode depender do *Issue* B, precisando que o *Issue* B seja solucionado antes de solucionar A.
O *Issue* pode ter um conjunto de ações (*Procedure*) que irá solucioná-lo e ainda, pode ser indicado por vários sintomas (*Symptoms*), onde um conjunto de *Symptoms* pode identificar um *Issue*.

Classe: *Symptom*
Representa um sintoma (sinal ou indicação) de um ou vários problemas (*Issues)*. Algo que é percebido quando um problema ocorre.

Propriedades: *indicates* e *indicatedBy*
indicates relaciona um *Symptom com um Issue que* ele indica. *indicatedBy* é propriedade inversa de *indicates*.
Um *Symptom* pode indicar vários *Issues*, como por exemplo um sintoma de febre pode indicar diversas doenças.

Propriedades: *causes* e *causedBy*
causes registra que um *Issue* causa outro *Issue* de forma direta ou indireta. Esta propriedade expressa uma relação de causalidade entre *Issues* e possui a característica de transitividade, ou seja, se o *Issue* A causa o *Issue* B e o *Issue* B causa o *Issue* C, então o *Issue* A causa o *Issue* C. *causedBy* é propriedade inversa de *causes*.

Propriedades: *directlyCauses* e *directlyCausedBy*
directlyCauses é subpropriedade de *causes* e indica que um *Issue* causa outro *Issue* de forma direta. *direclyCausedBy* é subpropriedade de *causedBy* e propriedade inversa de *directlyCauses*.
Por exemplo, considere um paciente que está com Pneumonia devido a uma Gripe mal curada, neste caso a Gripe foi *causa direta* da Pneumonia. Pelo axioma de subpropriedade, infere-se também que a Gripe foi a *causa* da Pneumonia.

Propriedades: *dependsOn* e *dependenceOf*
dependsOn relaciona um *Issue* com outro *Issue* do qual o primeiro depende direta ou indiretamente. Esta propriedade expressa uma relação de dependência entre *Issues* e possui a característica de transitividade, ou seja, se o *Issue* A depende do *Issue* B e o *Issue* B

depende do *Issue* C, então o *Issue* A depende do *Issue* C. *dependenceOf* é propriedade inversa de *dependsOn*.

Propriedades: *directlyDependsOn* e *directDependenceOf*
direcltyDependsOn é subpropriedade de *dependsOn* e indica que um *Issue* depende diretamente de outro *Issue*. *directDependenceOf* é subpropriedade de *dependenceOf* e propriedade inversa de *directlyDependsOn*.
Com base no exemplo exposto ao descrever a propriedade *directlyCauses*, a Pneumonia *depende diretamente* que a Gripe seja curada para, enfim, ser tratada. Pelo axioma de subpropriedade, deduz-se também que a Pneumonia *depende* da Gripe.

Propriedade: *canCause* e *canBeCausedBy*
canCause indica que um *Issue* pode causar outro *Issue* de forma direta ou indireta. Esta propriedade visa expressar uma relação de possível causalidade e possui a característica de transitividade, ou seja, se o *Issue* A pode causar o *Issue* B e o *Issue* B pode causar o *Issue* C, então o *Issue* A pode causar o *Issue* C. *canBeCausedBy* é propriedade inversa de *canCause*.

Propriedades: *canDirectlyCause* e *canBeDirectlyCausedBy*
canDirectlyCause é subpropriedade de *canCause* e indica que um *Issue* pode causar outro *Issue* de forma direta. Esta propriedade visa expressar uma relação de possível causalidade, onde um *Issue* pode ser a causa direta de outro *Issue*. *canBeDirectlyCausedBy* é subpropriedade de *canBeCausedBy* e propriedade inversa de *canDirectlyCause*.
Por exemplo, a doença Gripe pode ser *causa direta* da Pneumonia, ou seja, em alguns casos a Gripe causa a Pneumonia e em outros casos não. Pelo axioma de subpropriedade, infere-se também que Gripe *pode causar* Pneumonia.

Propriedades: *canDependOn* e *canBeDependenceOf*
canDependOn indica que um *Issue* pode depender de outro *Issue* direta ou indiretamente. Esta propriedade expressa uma relação de possível dependência entre *Issues* e possui a característica de transitividade, ou seja, se o *Issue* A pode depender do *Issue* B e o *Issue* B pode depender do *Issue* C, então o *Issue* A pode depender do *Issue* C. *canBeDependenceOf* é propriedade inversa de *canDependOn*.

Propriedades:
canDirectlyDependOn* e *canBeDirectDependenceOf
canDirectlyDependOn é subpropriedade de *canDependOn* e indica que um *Issue* pode depender diretamente de outro *Issue*.

canBeDirectDependenceOf é subpropriedade de *canBeDependenceOf* e propriedade inversa de *canDirectlyDependOn*.
Com base no exemplo exposto ao descrever a propriedade *canDirectlyCause*, a Pneumonia *pode depender*, *de forma direta*, que a Gripe seja curada para, enfim, ser tratada. Pelo axioma de subpropriedade, Pneumonia *pode depender* de Gripe.

Propriedades: *hasCausativeAsset* e *causativeAssetOf*
hasCausative relaciona um *Issue* com algum agente (*Asset*) causador deste *Issue*. *causativeAssetOf* é propriedade inversa de *hasCausativeAsset*.
Por exemplo, uma doença relacionada com seu o agente causador (um vírus, bactéria, etc.).

Propriedades: *hasHostAsset* e *hostAssetOf*
hasHostAsset relaciona um *Issue* com hospedeiro (*Asset*) no qual este *Issue* ocorreu. *hostAssetOf* é propriedade inversa de *hasHostAsset*.
Por exemplo, uma doença relacionada com a pessoa doente, ou seja, a pessoa é o hospedeiro da doença.

3.4 Problemas e soluções

As classes e propriedades desse módulo relacionam a classe *Issue* com a classe *Procedure* que representa uma possível solução para o problema. Ainda provê meios para delinear uma solução por meio de um *workflow*.

Classe: *Action*
Representa uma ação a ser realizada para resolver o problema. Uma *Action* pode ser apenas uma ação primitiva ou tarefa (*Task*) ou um conjunto de ações (*CompoundAction*).

Classe: *Task*
Uma ação elementar e atômica, que executa algo simples.

Classe: *CompoundAction*
Uma ação composta por várias outras ações. Uma *CompoundAction* pode ter o objetivo de solucionar um ou mais *Issues*, representando um procedimento (*Procedure*) ou não ter um objetivo explícito, apenas ser um grupo de *Actions* a ser reusado, se comportando como uma atividade (*Activity)*. Uma *CompoundAction* possui um ou mais passos de execução (*Steps*) que ativam uma *Action* (*Task* ou outra *CompoundAction*), permitindo assim, que uma *CompoundAction* reuse outras.
Uma *CompoundAction* pode ser utilizada para criar uma estrutura de *workflow*, visando uma melhor estruturação das ações que a compõem.

Classe: *Procedure*

Uma sequência de passos *(Steps)* com objetivo explícito que após executados solucionam um ou mais *Issues.*

Classe: *Activity*

Um conjunto de passos (*Steps*) que realizam uma atividade, porém não tem um objetivo explícito e, portanto, não visa a solução de um determinado *Issue*. Na verdade, trata-se de um agrupamento de ações visando reuso.

Classe: *Step*

Um passo ou etapa a ser realizada dentro de uma *CompoundAction*. Todo *Step* possui uma *Action* a ser executada e uma transição para outro *Step* a ser efetuada após finalizada a execução da *Action*.

Com os *Steps* é possível estabelecer uma ordem para a execução das *Actions*, pois cada *Step* possui uma transição (*Transition*) que estabelece o *Step* de origem e de destino. Uma vez que uma *CompoundAction* possui um *Step* inicial (*hasFirstStep*), a partir desse *Step*, é possível executar todos os outros *Steps* que compõe a *CompoundAction,* seguindo a transição entre eles.

Classe: *Transition*

Transition representa a transição entre dois *Steps*. Cada *Transition* tem um *Step* de origem (*source*) e um *Step* de destino (*target*). Uma *Transition* possui uma condição de guarda que especifica uma expressão booleana (*BooleanExpression*) que tem que ser verdadeira para que a transição ocorra.

Através da condição de guarda, pode-se implementar, de maneira simplificada, um *workflow*, estabelecendo estruturas de decisão, repetição, escolha, etc.

Classe: *BooleanExpression*

Uma expressão lógica cujo valor é verdadeiro ou falso, visando validar uma transição entre dois *Steps,* ou servir com uma pré-condição para a execução de uma *Action* ou ainda, servir como uma pós-condição que valide a execução de uma *Action*.

Propriedade: *hasPreCondition*

Indica uma pré-condição para que a *Action* seja executada. Uma pré-condição representa um pré-requisito definido por uma expressão booleana (*BooleanExpression*) para execução da *Action*, sem o qual não é garantida a execução correta da mesma.

Propriedade: *hasPostCondition*

Indica uma pós-condição (efeito) definida por uma expressão booleana (*BooleanExpression*) que será atingida após a execução da *Action*, desde que a pré-condição tenha sido respeitada.

Propriedade: *hasStep*

Indica um passo de execução (*Step*) que compõe a *CompoundAction*.

Propriedade: *hasFirstStep*

Uma *CompoundAction* possui um ou mais passos (*Steps*), assim essa propriedade indica o primeiro passo pelo qual a execução da *CompoundAction* é iniciada.

Propriedade: *goal*

Descrição do objetivo que se deseja alcançar após a execução do *Procedure*.

Propriedades: *solves* e *solvedBy*

solves relaciona a *Procedure* com o *Issue* que a ela soluciona. *solvedBy* é propriedade inversa de *solves*.

Propriedade: *activates*

Indica a ação a ser ativada por um *Step*.

Propriedades: *hasIncoming* e *hasTarget*

hasIncoming indica as transições (*Transitions*) de entrada do *Step*, ou seja, as que iniciam a execução deste *Step*. *hasTarget* é propriedade inversa de *hasIncoming*, indicando o *Step* destino da *Transition*. *hasTarget* é também uma propriedade funcional, uma vez que uma Transition tem um único *Step* destino.

Propriedades: *hasOutcoming* e *hasSource*

hasOutcoming indica as transições (*Transitions*) de saída do *Step,* ou seja, as que ocorrem após a execução deste *Step* e que acionam os próximos *Steps* a serem executados. *hasSource* é propriedade inversa de *hasOutcoming*, indicando o *Step* de origem da *Transition*. *hasSource* é também um propriedade funcional, uma vez que uma *Transition* tem um único *Step* de origem.

Propriedade: *hasGuardCondition*

Indica uma condição booleana (*BooleanExpression*) para que uma *Transition* ocorra.

Por exemplo, uma *Transition* pode ter como *guardCondition* que o *Step* de origem seja executado 10 vezes. Assim, enquanto este *Step* não for executado 10 vezes, não será iniciando o *Step* destino. Este exemplo ilustra uma estrutura de repetição dentro de um *workflow*.

4 Exemplo de instanciação da ontologia

Para demonstrar o uso da ontologia IPO, a Figura 2~~Figura 2~~ ilustra o mapeamento de instâncias (RDF) para classes/propriedades da ontologia (OWL), para o domínio de problemas médicos (doenças), onde foi abordada a *Gripe* como exemplo de instância da classe *Issue*. É demonstrado o relacionamento entre o problema *Gripe* com o sintoma *Febre*, instância da classe *Symptom*. O problema Gripe também é relacionado com um possível tratamento, instância da classe *Procedure*. Outro relacionamento representado é o vírus *Influenza,* instância da classe *Asset,* que possui uma relação de causador da Gripe. Além disso, é demonstrado um relacionamento entre o problema Gripe com seu hospedeiro, no caso, o Ser Humano, também instância da classe *Asset*. Como forma de organização, a Gripe foi agrupada na categoria *Doenças Virais*, instância da classe *skos:Concept*. Para demonstrar a dependência entre problemas, foi adicionado o problema Pneumonia, onde a Gripe pode causar Pneumonia e Pneumonia pode depender que a Gripe seja solucionada para, então, ser tratada. Os relacionamentos representados na cor verde são inferidos automaticamente pela máquina com base no axioma de subpropriedade. Para todos os outros relacionamentos, com base no axioma de propriedade inversa, os respectivos relacionamentos inversos também seriam inferidos, mas não foram ilustrados para não sobrecarregar a Figura 2~~Figura 2~~.

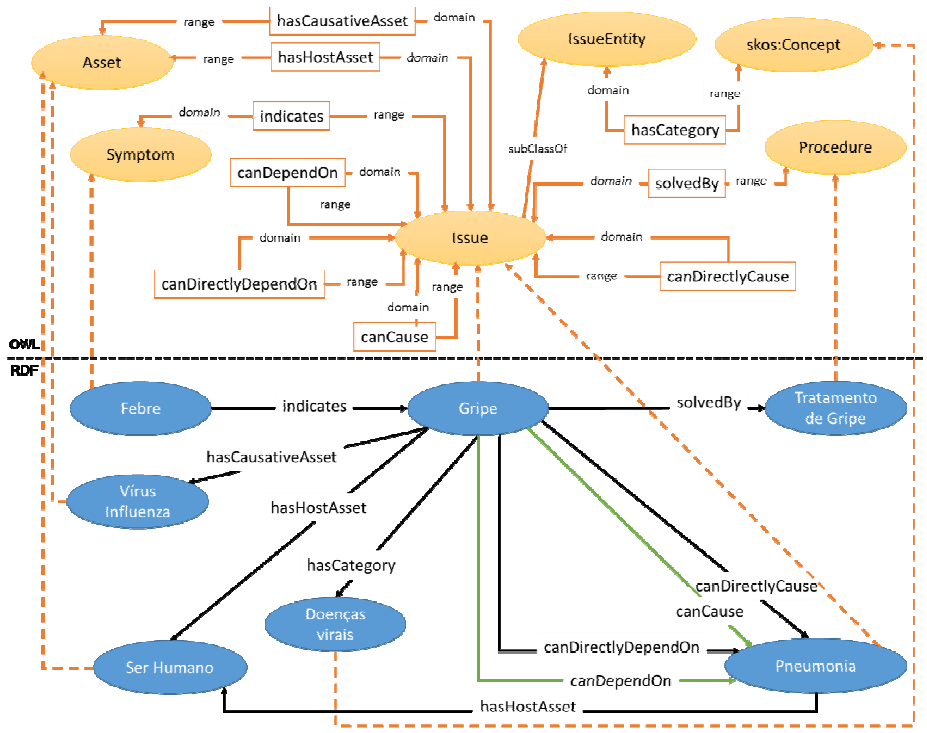

Figura 2 - Mapeamento entre dados RDF e a ontologia IPO

Fonte: Os autores (2015).

5 Exemplo de instanciação de procedimento (*Procedure*)

Como já foi dito anteriormente, um procedimento é representado por um *workflow*, contendo os passos a serem executados. A ontologia IPO fornece meios para representar, semanticamente, fluxogramas instanciando a classe *Procedure*. A Figura 3~~Figura 3~~ fornece um exemplo de um procedimento para tratamento do problema Pneumonia representado por um fluxograma.

Formatado: Cor da fonte: Preto

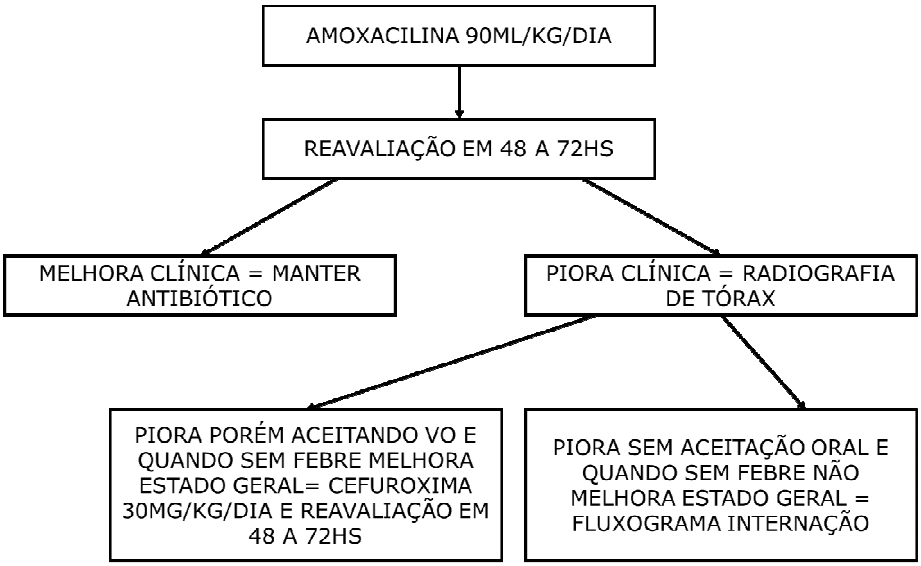

Figura 3 - Algoritmo de tratamento da Pneumonia

Fonte: MAYO CLINIC STAFF (2013).

A ontologia IPO determina que um *Procedure* possui um objetivo (*goal*) a ser alcançado no final de sua execução. Além disso, um *Procedure* possui (*hasStep*) um ou mais passos (*Step*), a serem executados (*Step*). Cada *Step* ativa (*activates*) uma ação (*Action*) que deve ser executada.

Uma *Action* é superclasse de *Task*, que representa uma tarefa simples e atômica, e de *CompoundAction*, que representa uma ação composta por diversas outras ações, simples ou compostas.

Após executar a *Action* de um determinado *Step*, deve-se passar para o próximo *Step*, ocorrendo assim, uma transição entre etapas (*Transition*). Cada *Transition*, porém, possui uma condição de guarda (*hasGuardCondition*) que consiste em uma expressão lógica (*BooleanExpression*) que deve ser atendida para que a *Transition* ocorra. Essas condições de guarda possibilitam que estruturas mais complexas de fluxograma, como condicionais e repetições, sejam construídas.

Para exemplificar uma instanciação da classe *Procedure*, a Figura 4~~Figura 4~~ traz o procedimento de tratamento da Pneumonia, que pode ser visto na Figura 3~~Figura 3~~, instanciado com a ontologia IPO. É importante notar que o *ipo-m:Step5* ativa um novo procedimento, ou seja, um *Procedure* pode fazer uso de outro *Procedure*, o que é bastante útil para o reuso de procedimentos ou tratamentos.

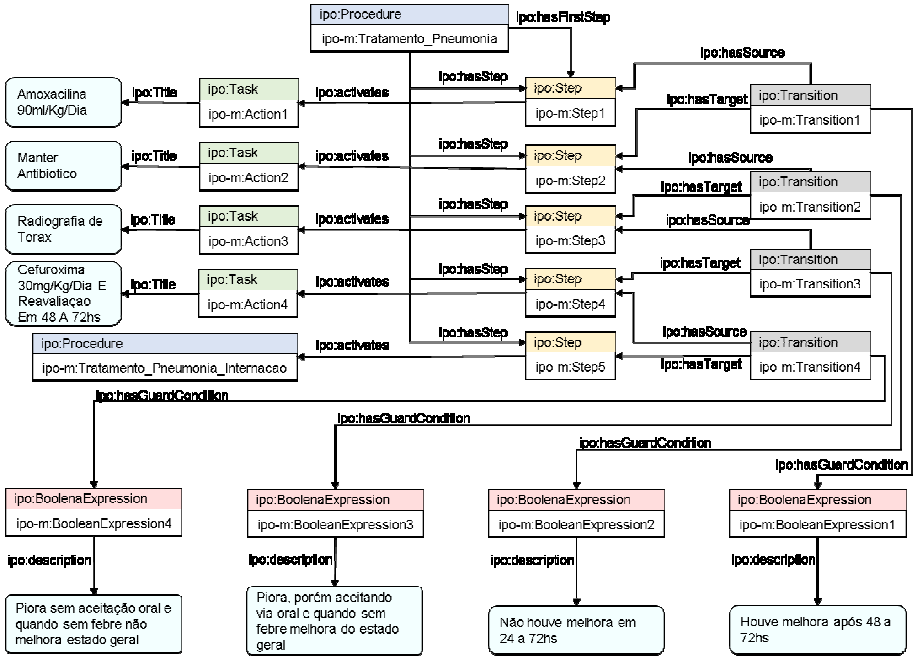

Figura 4 - Instanciação do algoritmo de tratamento da Pneumonia

Fonte: Os autores (2015).

6 Avaliação da ontologia proposta

A avaliação é uma etapa importante no processo de criação de uma ontologia. Para realizar essa validação, se faz necessário verificar se a ontologia é expressiva o suficiente para prover respostas para as questões de competência levantadas anteriormente e verificar se ela consegue representar todos os dados pertinentes ao domínio.

Por se tratar de uma ontologia extensível (*core ontology)*, a utilização deve ser feita a partir de especializações para um domínio mais específico, como por exemplo, o domínio de problemas e soluções médicas.

Como dito anteriormente, existem duas formas de classificação de uma *IssueEntity:* (1) uma classificação não intrínseca, ou seja, uma classificação visando um simples agrupamento de *IssueEntities,* onde a ontologia IPO provê a utilização da propriedade *hasCategory* relacionando com *skos:Concept*, possibilitando a criação de esquemas de categorias e subcategorias; (2) uma classificação intrínseca a um *IssueEntity*, de modo que essa classificação implica na criação de subclasses de *IssueEntity*, com restrições específicas. Por exemplo, a classe *Gripe*, sendo definida por uma restrição específica que é estar relacionado a pelo menos um dos sintomas *Febre* ou *Coriza* (instância da classe *Symptom*). Desta forma, ocorrências podem ser classificadas, com base na descrição de seus sintomas, como sendo uma instância da nova classe *Gripe*.

A seguir, para cada questão de competência, será demonstrado os meios oferecidos pela ontologia IPO que permitem responder a cada questão. Foi adotada a instanciação exposta na seção 4 para exemplificação das questões.

QC1. Qual é a categoria de um problema/sintoma/solução?

A propriedade *hasCategory* relaciona uma *IssueEntity* (classe genérica para *Symptom*, *Issue* e *Action*) com a classe *skos:Concept*. Assim, através dessa propriedade é possível obter a categoria (ou categorias) à qual a *IssueEntity* está relacionada. Como exemplo, se obteria a categoria *Doenças Virais* para o problema *Gripe* através da propriedade *hasCategory*. É importante notar que a IPO ainda provê a propriedade inversa *isCategoryOf* que relaciona a classe *skos:Concept* com uma *IssueEntity*, o que permite se obter todas as *IssueEntities* de uma determinada categoria.

QC2. Quais são sintomas de um problema?

A propriedade *indicatedBy* relaciona um *Issue* com a classe *Symptom*. Dessa forma, essa propriedade permite obter o(s) sintoma(s) que sinaliza(m) o *Issue*.

A ontologia IPO provê ainda a propriedade inversa *indicates* que relaciona um *Symptom* com um *Issue*, onde, através desta, se pode obter todos os *Issues* indicados por um dado sintoma. Por exemplo, o sintoma *Febre* indica *Gripe*.

QC3. Quais são as soluções de um problema?

Para se obter essa resposta, pode ser utilizada a propriedade *solvedBy*, que relaciona um *Issue* com a classe *Procedure*. Portanto, essa propriedade permite obter o(s) procedimento(s) que soluciona(m) o *Issue*.

Para se obter todos os problemas solucionados por um *Procedure*, pode ser fazer uso da propriedade inversa *solves*. Como pode ser visto no caso de a *Gripe* ser solucionada pelo procedimento *Tratamento de Gripe.*

QC4. Qual o causador do problema?

Uma informação importante sobre um problema é o seu causador. Para tal, a IPO fornece a propriedade *hasCausativeAsset* que relaciona um *Issue* com um *Asset.* Assim, através dessa propriedade é possível obter o causador de um dado problema.

Também é possível saber todos os problemas causados por um *Asset*, por meio da propriedade inversa *causativeAssetOf*. Por exemplo, a *Gripe* tem como agente causador o vírus *Influenza*.

QC5. Qual o hospedeiro do problema?

A propriedade *hasHostAsset* pode ser utilizada para relacionar um *Issue* com o seu hospedeiro (*Asset*), ou seja, relacionar um problema com a "coisa" na qual ele ocorre.

Também é possível saber todos os problemas que ocorreu em um determinado *Asset*, por meio da propriedade inversa *hostAssetOf*. Por exemplo, *Gripe* tem como hospedeiro o Ser Humano.

QC6. Quem criou/registrou esse problema/sintoma/solução?

A propriedade *hasMaker* relaciona um *IssueEntity* com a classe *Party* (*Person* ou *Organization*). Assim, através dessa propriedade se pode obter a pessoa ou organização que criou ou registrou uma determinada *IssueEntity*. É importante notar que a IPO ainda oferece a propriedade inversa *makerOf* que relaciona a classe *Party* com uma *IssueEntity*, o que permite se obter todas as *IssueEntities* criadas por uma pessoa ou organização. Não consta no exemplo de instanciação, porém, um médico poderia estar relacionado com o registro do problema *Gripe* por meio da propriedade *hasMaker*.

QC7. Quais ações (*workflow*) a serem tomadas para solucionar o problema?

Com já mencionado, *Procedure* é a classe que representa a solução de um problema e é composta por um ou mais *Steps*. Cada *Step* ativa uma ação a ser executada. Além disso, o *Step* possui uma *Transition* de saída, que irá promover a transição para o próximo *Step* do *Procedure*.

Sendo assim, para se obter todas as ações a serem sequencialmente executadas, primeiramente deve se obter o primeiro *Step* do *Procedure*, através da propriedade *hasFirstStep*. A partir desse primeiro *Step*, é possível, através da propriedade *hasOutcomming*, obter a *Transition* que irá acionar o próximo *Step*.

A propriedade *hasTarget*, que relaciona uma *Tansition* com um *Step*, indica o *Step* destino da transição.

Através das classes e propriedades aqui mencionadas, é possível obter todos os *Steps*, ordenados, que deverão ser executados.

Para se obter as ações referente a cada *Step*, basta fazer uso da propriedade *activates*, que relaciona o *Step* com a ação a ser executada.

QC8. Um problema causa outro problema?

A propriedade *causes* relaciona a classe *Issue* com ela própria. Esse relacionamento expressa que um *Issue* causa outro *Issue*. A propriedade *directlyCauses* também pode expressar um relacionamento de causalidade, porém com a característica de ser uma causalidade direta, ou seja, indicando que um *Issue* causa diretamente outro.

Assim, a propriedade *causes* expressa que o *Issue* causa outro *Issue* de maneira direta ou indireta.

Com essas propriedades é possível saber se um problema causa outro, direta ou indiretamente.

QC9. Um problema depende de outro problema?

A propriedade *dependsOn* relaciona a classe *Issue* com ela própria. Esse relacionamento expressa que um *Issue* depende de outro *Issue*. A propriedade *directlyDependsOn* também pode expressar um relacionamento de dependência, porém com a característica de ser uma dependência direta, ou seja, indicando que um *Issue* depende diretamente de outro.

Assim, a propriedade *dependsOn* expressa que o *Issue* depende de outro *Issue* de maneira direta ou indireta.

Com essas propriedades é possível saber se um problema depende de outro, direta ou indiretamente.

QC10. Quais os possíveis problemas, a partir de um conjunto de sintomas?

Fortemente relacionada à questão QC2, esta questão pode ser respondida por meio da propriedade *indicates* que relaciona um *Symptom* com o *Issue* que ele indica. Dado um conjunto de *Symptoms*, é possível obter todos os *Issues indicados* por estes *Symptoms*. Para melhorar ainda mais, os *Issues* retornados podem ser ordenados pela quantidade de *Symptoms* presentes no referido conjunto, obtendo-se, assim, uma lista de *Issues* ordenados do mais provável para o menos provável, usando-se para tal a propriedade inversa *indicatedBy*, que relaciona o *Issue* com seus *Symptons*.

QC11. Em quais possíveis classes (de problema) uma ocorrência no espaço/tempo pode ser classificada, a partir dos sintomas identificados?

A classe *Issue* tem como objetivo descrever problemas. Para que possamos registrar semanticamente as ocorrências no espaço/tempo de problemas, faz-se necessário criar subclasses da classe *Issue* que representem os problemas e essas subclasses terão como instancias as ocorrências.

Diante disso, visando permitir a classificação automática de ocorrências de problemas a partir de seus sintomas, a título de exemplo de classificação intrínseca em um domínio específico, foi criada uma subclasse da classe *Issue* denominada *Gripe*, definida como equivalente ao conjunto de recursos que apresentam pelo menos um dos sintomas (instâncias da classe *Symptom*): *Febre* ou *Coriza*. Em outras palavras, qualquer ocorrência que apresente pelo menos um destes dois sintomas, a máquina, automaticamente, a classificará como sendo uma instância da classe *Gripe*. A Figura 5~~Figura 5~~ demonstra esse exemplo.

Formatado: Cor da fonte: Preto

As relações na cor verde são obtidas automaticamente quando a máquina classifica a ocorrência como instância da classe Gripe. Perceba que o ser humano apenas precisa informar os sintomas (relacionamentos na cor preta) e a máquina decide, com base na ontologia, quais possíveis doenças se aplica àquela ocorrência.

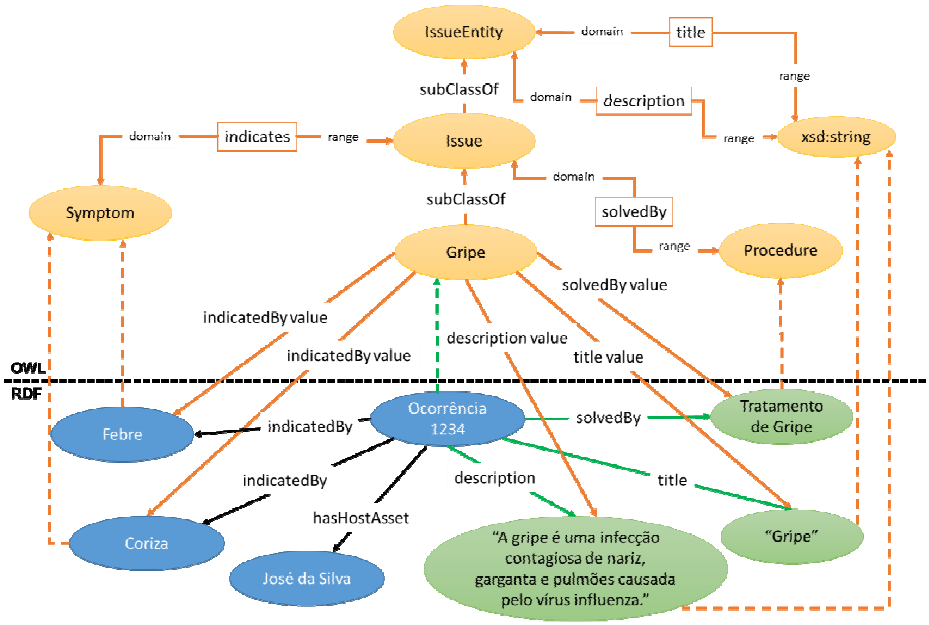

Figura 5 - Exemplo de instanciação com a subclasse Gripe

Fonte: Os autores (2015).

7 Considerações finais e trabalhos futuros

A *Issue Procedure Ontology* (IPO) tem o objetivo de representar o domínio de problemas, relacionando sintomas, problemas e soluções. Para sua construção, foi elaborado um conjunto de questões de competência visando definir algumas perguntas-chave que, depois de finalizada, a ontologia deveria prover meios para responder. Assim a ontologia IPO contém 16 classes, 48 propriedades e 2 indivíduos implementados utilizando a linguagem OWL 2.0 e todos os termos foram apresentados na seção 3.

A fim de se avaliar se a ontologia atende às questões de competência, foram apresentados exemplos realistas de instanciação evidenciando o poder de expressividade da ontologia.

A ontologia fornece os mecanismos necessários para que a máquina seja capaz de identificar os possíveis problemas a partir de sintomas, bem como sugerir possíveis soluções a partir dos problemas identificados. Diferentemente de um sistema de informação, onde a máquina fornece informações para que o ser humano resolva o problema, a IPO representa um passo fundamental em direção a construção de sistemas de conhecimento, onde a máquina forneça não apenas informações, mas também forneça possíveis soluções, de forma autônoma, para um problema em questão.

Como a ontologia IPO é uma ontologia extensível (*core ontology)*, para se aplicar em contexto real de forma mais fidedigna, é recomendado especializá-la para se obter uma maior adequação ao domínio a ser

abordado. No decorrer deste artigo, foram expostos diversos exemplos relacionados com o domínio problemas médicos e, paralelamente a ontologia, está sendo desenvolvido um estudo de caso realista dentro deste domínio, abordando algumas doenças, com seus sintomas e tratamentos, onde a ontologia IPO é estendida por subclasses descrevendo especificamente cada uma destas doenças.

Como importante trabalho futuro, esta ontologia precisa ser inserida em alguns contextos reais, sendo utilizada por um período considerável de tempo para que seja paulatinamente ajustada às necessidades dos usuários. Um processo contínuo de aperfeiçoamento peculiar a construção de ontologias.

Referências

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic web. *Scientific American*, v. 284, n. 5, p. 28-37, 2001.

BRICKLEY, D.; MILLER, L. *FOAF vocabulary specification 0.99*. 2014. Disponível em: <http://xmlns.com/foaf/spec/>. Acesso em: 11 maio 2015.

CARTLIDGE, A. *et al.* **An introductory overview of ITIL® V3**. 1ª ed. Wokingham: It Service Management Forum Limited, 2007. 56 p. Disponível em: <http://www.itsmf.org.rs/sites/default/files/itSMF%20ITIL%20V3%20Introduction%20Overview.pdf>. Acesso em: 20 maio 2015. 978-0-9551245-8-7.

CYGANIAK, R.; WOOD, D.; LANTHALER, M. *RDF 1.1 concepts and abstract syntax*. **W3C Recommendation**, 2014. Disponível em: <http://www.w3.org/TR/rdf11-concepts/>. Acesso em: 11 maio 2015.

DCMI USAGE BOARD. *DCMI metadata terms*. 2012. Disponível em: <http://dublincore.org/documents/dcmi-terms/>. Acesso em: 11 maio 2015.

HITZLER, P. *et al.* OWL 2 Web Ontology Language Primer. 2ª ed. **W3C Recommendation**, 2012. Disponível em: <http://www.w3.org/TR/2012/REC-owl2-primer-20121211/>. Acesso em: 05 maio 2015.

JACYNTHO, M. D. DE A. *Um modelo de bloqueio multigranular para RDF*. Rio de janeiro: Pontifícia Universidade Católica do Rio de Janeiro; Departamento de Informática, 2012.

LICHTNOW, D.; DE OLIVEIRA, J. P. M. Relato e Considerações sobre o Desenvolvimento de uma Ontologia para Avaliação de Sites da Área de Saúde. *Cadernos de Informática*, v. 4, n. 1, p. 7-46, 2009.

MAGALHÃES, I. L.; PINHEIRO, W. B. Gerenciamento de serviços de TI na prática: uma abordagem com base na Itil. São Paulo: Novatec, 2007.

MAXIMIANO, A. C. A. *Introdução à administração.* São Paulo: Atlas, 2004.

MAYO CLINIC STAFF. *Pneumonia.* 2013. Disponível em: <http://www.mayoclinic.org/diseases-conditions/pneumonia/basics/definition/con-20020032>. Acesso em: 10 jan. 2015.

MICHAELIS, H. *Michaelis:* dicionário brasileiro da língua portuguesa. 2015. Disponível em: <http://michaelis.uol.com.br/>. Acesso em: 30 mar. 2015.

MILES, A.; BECHHOFER, S. *SKOS Simple Knowledge Organization System Namespace Document-HTML*. W3C Recommendation. 2009. Disponível em: <http://www.w3.org/2009/08/skos-reference/skos.html>. Acesso em: 11 maio 2015.

NOY, N. F.; MCGUINNESS, D. L. *Ontology development 101*: a guide to creating your first ontology. 2001. Stantford Medical Informatics Technical Report, SMI-2001-0880. Disponível em: <http://protege.stanford.edu/publications/ontology_development/ontology101.pdf>. Acesso em: 24 abril 2015.

PEREIRA, J. R.; DE SOUZA, M. A.; DA COSTA, H. R. Gerenciamento de problema: uma abordagem com base na Itil. *Pensar Tecnologia*, v. 1, n. 2, p. 23-36, 2012.

PICKLER, M. E. V. Web semântica: ontologias como ferramentas de representação do conhecimento. *Perspectivas em Ciência da Informação*, v. 12, n. 1, p. 65-83, 2007.

REZENDE, D. A. Tecnologia da informação integrada à inteligência empresarial. São Paulo: Atlas, 2002.

YU, L. A developer's guide to the semantic web. Berlin: Springer, 2011.