UNIVERSIDADE DO VALE DO RIO DOS SINOS - UNISINOS
CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA INTERDICIPLINAR DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA
NÍVEL MESTRADO

JOÃO PABLO SILVA DA SILVA

UM SISTEMA MULTIAGENTE BASEADO EM ONTOLOGIAS PARA APOIO ÀS INSPEÇÕES DE GARANTIA DA QUALIDADE DE SOFTWARE

SÃO LEOPOLDO
2010

João Pablo Silva da Silva

UM SISTEMA MULTIAGENTE BASEADO EM ONTOLOGIAS PARA APOIO ÀS INSPEÇÕES DE GARANTIA DA QUALIDADE DE SOFTWARE

Dissertação apresentada com requisito parcial para obtenção do título de Mestre em Computação Aplicada, pelo Programa Interdisciplinar de Pós-graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos.

Orientador: Prof. Dr. Sérgio Crespo

São Leopoldo

2010

Ficha Catalográfica

S586s Silva, João Pablo Silva da
Um sistema multiagente baseado em ontologias para apoio às inspeções de garantia da qualidade de software. / por João Pablo Silva da Silva. – 2010.
104 f. : il. ; 30cm.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos, Programa Interdisciplinar de Pós-Graduação em Computação Aplicada, São Leopoldo, RS, 2010.

"Orientação: Prof. Dr. Sérgio Crespo, Ciências Exatas e Tecnológicas".

1. Agente inteligente – Software. 2. Ontologia – Computação. 3. Garantia da qualidade – Software. I. Título.

CDU 004.89

Catalogação na Publicação:
Bibliotecária Camila Rodrigues Quaresma - CRB 10/1790

João Pablo Silva da Silva

UM SISTEMA MULTIAGENTE BASEADO EM ONTOLOGIAS PARA APOIO ÀS INSPEÇÕES DE GARANTIA DA QUALIDADE DE SOFTWARE

Dissertação apresentada com requisito parcial para obtenção do título de Mestre em Computação Aplicada, pelo Programa Interdisciplinar de Pós-graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos.

Aprovado em __/__/____.

BANCA EXAMINADORA

_____

Nome do Componente - Instituição do Componente

_____

Nome do Componente - Instituição do Componente

_____

Nome do Componente - Instituição do Componente

Esta Dissertação de Mestrado é
dedicada à minha Esposa,
Familiares e Amigos.

AGRADECIMENTOS

Inicialmente, agradeço ao Grupo Meta, por viabilizar e inspirar o objeto de pesquisa trabalhado ao longo desta dissertação. Também agradeço à UNISINOS, por prover meios para que o trabalho em questão fosse desenvolvido.

Faço um agradecimento especial ao meu orientador, Dr. Sérgio Crespo, por acreditar e conduzir os trabalhos rumo aos seus resultados. Complementarmente, agradeço aos professores Dr. João Gluz e Dr. Jorge Barbosa, por incentivar o desenvolvimento da proposta.

Por fim, agradeço ao colega de pesquisa Pablo Dall'Oglio, pela constante troca de ideias e experiências sempre pertinentes para o atendimento dos objetivos estabelecidos. Também agradeço à coleta de Grupo Meta Sabrina Dalcin pelo apoio dado na revisão deste trabalho.

RESUMO

O presente trabalho apresenta um sistema de apoio às inspeções de garantia da qualidade de software, baseado em ontologias e agentes, que objetiva a automação de atividades específicas em um processo de inspeção. A ontologia, elaborada através do método *Ontology Development 101*, mapeia os conceitos mínimos e necessário para representar o domínio de conhecimento de inspeções de garantia da qualidade. O agente, desenvolvido através de um método iterativo e incremental, e especificado com *Unified Modeling Language*, implementa um conjunto de critérios para a manipulação de indivíduos na ontologia. O protótipo de interface web viabiliza a interação com o agente e a ontologia, respectivamente. O sistema é verificado através de testes unitários e integrado, e é validado através de experimentos em um ambiente real de uso. A coleta de dados para comparação se dá em dois momentos, antes do uso do sistema e após uso do sistema. Por fim, conclui-se que o sistema dá um ganho significativo na cobertura das inspeções e garante a manutenção, tanto dos valores de aderência ao processo, quanto dos valores de esforço das inspeções.

Palavras-chave: Garantia da Qualidade. Ontologia. Agente.

ABSTRACT

This work presents a support system for inspection of software quality assurance, based on ontologies and agents, which aims at automation of specific activities in a process inspection. The ontology, developed with the Ontology Development 101 method, maps the minimum and necessary concepts to represent the domain knowledge of quality assurance inspections. The agent, development with iterative and incremental method, and specified with the Unified Modeling Language, implements a set of criteria for the handling of individuals in the ontology. The web interface prototype enables the interaction with the agent and ontology, respectively. The system is verified through individual and integrated testing, and is validated through experiments on a real environment of use. The collection of data for comparison occurs in two moments before the use of the system and after using the system. Finally, it is concluded that the system provides a significant gain in coverage of inspections and the maintenance of both the values of adherence to process and the values of the inspection effort.

Keywords: Quality Assurance. Ontology. Agent.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AIP	Agentes de Inspeção de Processo
API	*Application Programming Interfaces*
CMMI	*Capability Maturity Model Integration*
DAML	*DARPA Agent Markup Language*
DL	*Description Logic*
FIPA	*Foundation for Intelligent Physical Agents*
IEC	*International Electrotechnical Commission*
ISO	*International Organization for Standardization*
JADE	*Java Agent Development*
Java EE	*Java Enterprise Edition*
JSF	*Java Server Faces*
JSP	*Java Server Pages*
LGPL	*Library Gnu Public Licence*
ML	*Maturity Level*
MVC	*Model-View-Controller*
OIL	*Ontology Inference Layer*
OIP	Ontologia de Inspeção de Processo
OnSSPKR	*Ontology Supported Software Process Knowledge Representation*
OPD	*Organizacional Process Definition*
OPF	*Organizational Process Focus*
OWL	*Web Ontology Language*
PA	*Process Area*
PPQA	*Process and Product Quality Assurance*
RDF	*Resource Description Framework*
RDFS	*RDF Schema*
SEI	*Software Egineering Institute*
SG	*Specific Goal*
SGBD	Sistema Gerenciador de Banco de Dados
SP	*Specific Practice*
SPO	*Software Process Ontology*
SQL	*Structured Query Language*
SWEBOK	*Software Engineering Body of Knowledge*
TAI	Total de Artefatos Inspecionáveis
TIRA	Total de Itens de Revisão Aplicáveis
TIRAp	Total de Itens de Revisão Aplicáveis
TIRAt	Total de Itens de Revisão Atendidos
URI	*Uniform Resource Identifier*
VAL	*Validation*
VER	*Verification*
W3C	*Word Wide Web Consortium*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Partindo da premissa que a qualidade de um produto de software é altamente influenciada pela qualidade do processo de construção deste software, torna-se estratégico para fornecedores monitorar e controlar a execução de processos de desenvolvimento. É dentro deste contexto que o presente trabalho é introduzido, levantando potenciais formas de se explorar o domínio de aplicação de garantia da qualidade, tanto cientificamente, quanto tecnologicamente.

Este Capítulo apresenta na Seção 1.1 a contextualização e motivação para o trabalho, evidenciando o objeto de pesquisa. A Seção 1.2 traz as principais contribuições obtidas com o trabalho. Na Seção 1.3 é feito o detalhamento do objetivo geral e dos objetivos específicos. A Seção 1.4 apresenta a metodologia de trabalho aplicada. A Seção 1.5 faz o fechamento caracterizando os demais Capítulos.

1.1 CONTEXTO E MOTIVAÇÃO

O mercado corporativo consumidor de software está cada vez mais exigente quanto à qualidade dos produtos e serviços comprados. Tal característica, demanda dos fornecedores de software investimentos em suas estruturas de desenvolvimento para garantir a qualidade de seus produtos e serviços. Neste contexto, a engenharia de software ganha força, pois estabelece as capacidades necessárias para uma cadeia produtiva de software, desde suas definições iniciais, até a respectiva entrada em produção (SOMMERVILLE, 2001). A engenharia de software é fundamentada nos princípios da engenharia convencional, para obter softwares economicamente viáveis, confiáveis e eficientes (PRESSMAN, 2001), sendo definida como a aplicação de uma abordagem sistemática, disciplinada e quantificável, no desenvolvimento, operação e manutenção de um software (ABRAN et al., 2004).

Essencialmente, abordagens de engenharia de software se apoiam em compromissos com a qualidade, tendo em suas bases os processos de software (PRESSMAN, 2001). Porém, processos de software não são autossuficientes, ou seja, estes somente são funcionais, quando as pessoas que o executam estão comprometidas com os mesmos. Seguindo esta linha, modelos de referência de qualidade introduziram em seus *frameworks* de engenharia de software a disciplina de garantia da qualidade. A garantia da qualidade de software se

caracteriza por dar transparência à execução de um processo de software e por viabilizar a tomada de ação quando desvios são identificados. Atividades de garantia da qualidade, essencialmente, vistoriam de forma objetiva a execução do processo em projetos de software, externando as não conformidades que expõe a riscos, a qualidade do produto final (CHRISSIS; KONRAD; SHRUM, 2007).

O mercado corporativo também contribui para o constante crescimento das informações disponibilizadas na rede mundial de computadores. Tal crescimento torna inviável o processamento destas informações por humanos, sendo necessário o apoio de sistemas inteligentes com tal capacidade. Para que computadores possam processar informações, faz-se necessário descrevê-las e organizá-las segundo um formalismo que viabilize a sua manipulação e interoperabilidade. Tal necessidade impulsiona a web semântica (SHADBOLT; HALL; BERNERS-LEE, 2006). No mesmo sentido, pesquisadores têm usado tecnologias de web semântica para resolver problemas de engenharia de software que remetam à necessidade de representar, manipular ou compartilhar conhecimento (DIETRICH; JONES, 2007).

Dentre as tecnologias relacionadas com a web semântica, destacam-se as ontologias. Ontologias são representações formais, explícitas e compartilhadas, que viabilizam a manipulação de conhecimento por sistemas computacionais (GRUBER, 1993). Soluções baseadas em ontologias podem ser aplicadas em diversas áreas de conhecimento, inclusive na engenharia de software. Publicações sobre mapeamento de padrões de projeto (DIETRICH; ELGAR, 2005), colaboração e compartilhamento de informações (WONGTHONGTHAM; CHANG; DILLON, 2006), reuso de software (GHIOTTO et al., 2006) e engenharia de requisitos (KAIYA; SAEKI, 2005), reforçam esta tendência.

Ontologias, em sua essência, são representações de conhecimento, as quais precisam ser manipuladas por algum sistema para se tornarem funcionais. Como alternativa tecnológica a esta necessidade, tem-se os sistemas inteligentes baseados em agentes. Agentes de software são programas de computador dotados de autonomia, pró-atividade e sociabilidade (RUSSELL; NORVIG, 2004). Tais características capacitam a construção de sistemas inteligentes baseados em agentes para a manipulação de ontologias, como pode ser observado em diversas publicações científicas (GUARINO, 1995) (HOSS; CARVER, 2006) (WANG; LEE, 2007) (WANG; LEE, 2008).

A motivação deste trabalho está na exploração do uso de ontologias e agentes na solução de problemas de engenharia de software, dentro de um ambiente real de desenvolvimento. Acredita-se que o uso de dados reais, coletados a partir da implantação de

práticas de garantia da qualidade em uma estrutura de desenvolvimento de software de uma organização, viabiliza a elaboração de uma solução de automação, baseada em ontologias e agentes, sob uma ótica teórico-prática aplicada. Em resumo, fica caracterizado como objeto de pesquisa deste trabalho, o uso de agentes e ontologias na automação de inspeções de garantia da qualidade de software, com base em um cenário real de aplicação.

1.2 PRINCIPAIS CONTRIBUIÇÕES

O presente trabalho é caracterizado por uma abordagem prática de implementação de garantia da qualidade em um ambiente real de desenvolvimento de software, apoiado por um sistema de automação baseado em ontologias e agentes. Neste contexto, podem-se listar como principais contribuições os seguintes itens:

- a estruturação de uma estratégia de implementação de garantia da qualidade de software;
- a modelagem do domínio de conhecimento de garantia da qualidade de software através de uma ontologia;
- o projeto de agentes de software capazes de automatizar atividades específicas de inspeções de garantia da qualidade.

1.3 OBJETIVO GERAL E ESPECÍFICOS

Este trabalho tem por objetivo geral a construção de um sistema de apoio às inspeções de garantia da qualidade de software, baseado em ontologias e agentes, capaz de aumentar a cobertura destas inspeções, sem impactar no esforço despendido para a execução das atividades. O objetivo geral pode ser decomposto nos seguintes objetivos específicos:

- realizar a análise de viabilidade técnica do projeto;
- modelar o domínio de inspeções de garantia da qualidade;
- projetar agentes de apoio às atividades de inspeção;
- construir um protótipo de interface para os agentes e a ontologia;
- experimentar o sistema em um ambiente real de desenvolvimento.

1.4 METODOLOGIA DE TRABALHO

Para viabilizar o pleno atendimento dos objetivos citados na Seção 1.3, faz-se necessário o estabelecimento de uma metodologia de trabalho, a qual pode ser caracterizada por:

- formulação do objeto de pesquisa com base na observação de execuções de inspeções de garantia da qualidade em um ambiente real;
- pesquisa por alternativas de soluções para o problema levantado em bases de produções científicas;
- análise de viabilidade através de estudo e práticas com potenciais tecnologias aplicáveis ao problema;
- análise e estabelecimento de uma visão de solução baseada no objeto de pesquisa e estudos realizados;
- projeto e implementação da solução estabelecida, usando o ferramental selecionado para o trabalho;
- verificação e validação da solução com base em análise dos resultados obtidos em um cenário real de aplicação;
- registro em documento da solução elaborada, seus resultados e respectivas análise comparativas.

1.5 ORGANIZAÇÃO DO DOCUMENTO

O restante deste trabalho está organizado de acordo com a seguinte sequência de Capítulos:

- *Capítulo 2.* Apresenta uma fundamentação teórica e tecnológica para os conceitos e ferramentas necessárias para o entendimento da solução apresentada, sendo abordados ontologias, agentes e aplicações web.
- *Capítulo 3.* Apresenta uma revisão detalhada de garantia da qualidade de software, desde seus conceitos e definições, até uma estratégia de implementação das práticas em uma estrutura de desenvolvimento.

- *Capítulo 4.* Apresenta uma coleção de publicações pesquisadas em bases específicas da área que, de alguma forma, estão relacionadas com a visão de solução para o problema levantado.
- *Capítulo 5.* Apresenta a análise, projeto e implementação da ontologia de inspeções de garantia da qualidade, dos agentes de inspeções de garantia da qualidade e do protótipo de interface para interação com os agentes e a ontologia.
- *Capítulo 6.* Apresenta a contextualização do ambiente utilizado para estudo de caso, detalha a estratégia de verificação e validação utilizada e realiza uma análise estatística dos resultados obtidos.
- *Capítulo 7.* Apresenta o fechamento do trabalho através de suas conclusões, sendo descritas as vantagens e desvantagens da solução apresentada, as principais contribuições do trabalho e as potenciais evoluções através de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA E TECNOLÓGICA

Para que os objetivos propostos neste trabalho sejam plenamente atendidos, faz-se necessário o uso de conceitos e tecnologias relacionados às ontologias e agentes, logo, é demandado um entendimento básico que viabilize a compreensão das soluções apresentadas. Complementarmente, torna-se desejável a contextualização de tecnologias de desenvolvimento de aplicações web, utilizadas para implementar um protótipo de interface com usuários. Com base nisto, o presente Capítulo propõe uma fundamentação teórica e tecnológica para nivelar o conhecimento relacionado ao domínio da solução deste trabalho.

Inicialmente, o Capítulo apresenta na Seção 2.1 uma revisão teórica e tecnológica sobre ontologias, onde são apresentados conceitos e definições, um método de engenharia de ontologias, linguagens de especificação e de consulta, e um *framework* para codificação de aplicações. Na sequência, o Capítulo traz na Seção 2.2, uma visão geral de agentes, onde estes são definidos e caracterizados, sendo também apresentada, uma plataforma para codificação e execução de agentes. A Seção 2.3 faz uma breve contextualização de aplicações web, caracterizando uma plataforma de desenvolvimento e um padrão de projeto aplicável a esta. A Seção 2.4 faz o fechamento do capítulo.

2.1 ONTOLOGIAS

O termo “Ontologia” vem de um ramo da filosofia que estuda o ser e sua existência, sendo definido como a teoria da natureza da existência. O termo foi inicialmente adotado na computação por pesquisadores de inteligência artificial, os quais identificaram sua aplicabilidade e construíram modelos computacionais com algum tipo de raciocínio automatizado (GRUBER, 2010). No início da década de 90, ontologias começaram a ser tratadas como parte integrante de sistemas baseados em conhecimento, sendo definida como uma especificação explícita de conceitualizações. Conceitualizações são abstrações, uma visão simplificada daquilo que se quer representar por algum motivo (GRUBER, 1993).

No contexto da ciência da computação e informação, uma ontologia define um conjunto de primitivas representacionais de um determinado domínio de conhecimento. As primitivas representacionais são fundamentalmente classes, atributos e relacionamentos, incluindo seus significados e restrições que garantam logicamente aplicações consistentes.

Ontologias, tipicamente, são especificadas com linguagens que permitam alguma forma de abstração a partir de estruturas de dados e estratégias de implementação. Por isso, ontologias se encontram em nível semântico, ao contrário de esquemas de bases de dados que se encontram em um nível lógico e físico. (GRUBER, 2010). Com base na definição de ontologias, três princípios fundamentais podem ser caracterizados (WONGTHONGTHAM; CHANG; DILLON, 2007):

- ontologias são representações formais, logo, são passíveis de compreensão e manipulação por um computador;
- ontologias são representações explícitas, ou seja, suas características são declaradas de forma compreensível a qualquer um;
- ontologias são representações compartilhadas, onde qualquer envolvido tem acesso ao conhecimento ali representado.

A Figura 1 apresenta um fragmento[1] de uma ontologia de vinhos, onde se pode observar a organização hierárquica de diversos conceitos relacionados ao domínio de conhecimento em questão. O primeiro nível define um conceito genérico que representa todas as coisas do mundo, a partir deste conceito é possível derivar qualquer outro. O segundo nível da hierarquia apresenta uma coleção de conceitos relacionados ao domínio dos vinhos. Os níveis subsequentes, nada mais são que a especialização de conceitos em subconceitos, até que se tenha a granularidade necessária para representar o conhecimento envolvido.

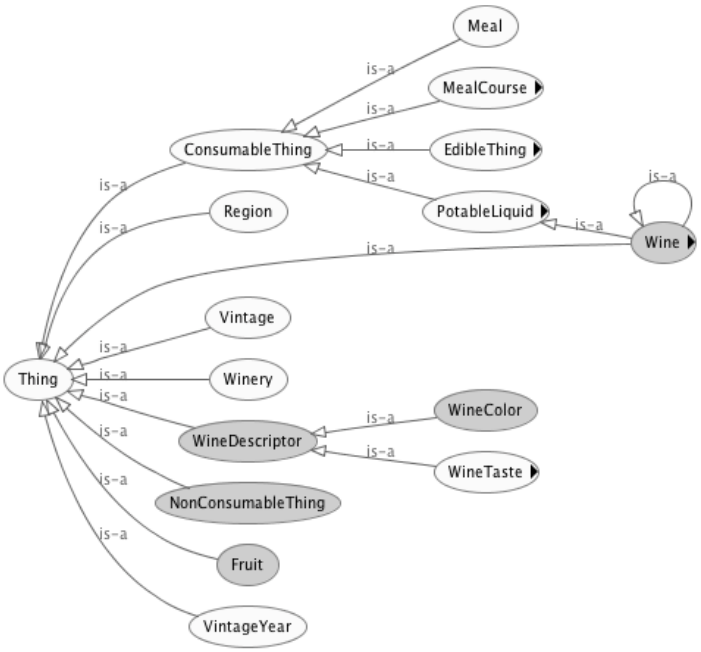

FIGURA 1 - Exemplo de uma ontologia de vinho.

[1] Ontologia completa disponível em: http://www.w3c.org/TR/owl-guide/wine.rdf.

2.1.1 Engenharia de Ontologias

Devido a sua natureza, a construção de ontologias requer um método de engenharia de conhecimento. Tal método deve considerar que não há uma única forma de modelar um domínio, que a aplicação em questão influencia na modelagem e que ontologias são desenvolvidas através de processos iterativos. Seguindo estas premissas, a *Ontology Development 101* destaca-se por sua simplicidade e objetividade, sem perder a capacidade de cobertura das principais questões referentes à construção de uma ontologia. A Figura 2 apresenta a metodologia através de um diagrama iterativo (NOY; MCGUINNESS, 2001).

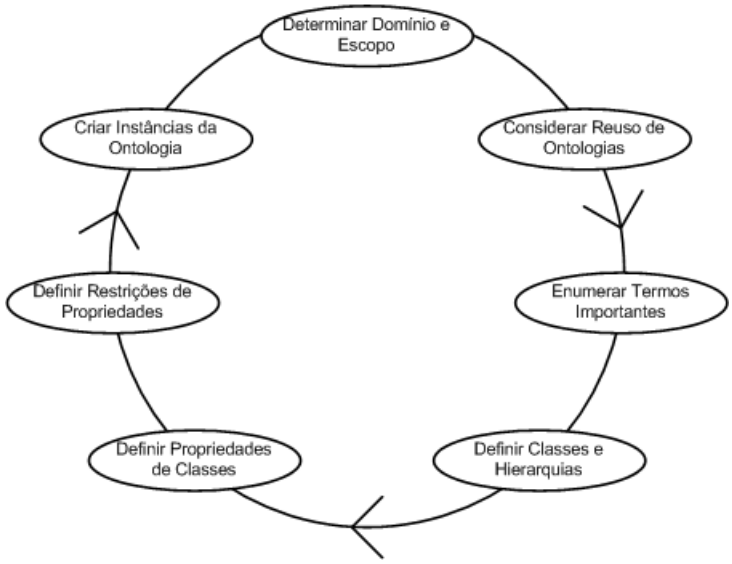

FIGURA 2 - Um método de engenharia de ontologias.

A Figura 2 apresenta inicialmente a etapa *Determinar Domínio e Escopo*, onde é definido o domínio de conhecimento que se quer representar e qual o escopo da aplicação que se deseja construir. A segunda etapa apresentada é *Considerar Reuso de Ontologias*, a qual faz pensar sobre questões de reuso de ontologias já definidas. A terceira etapa é *Enumerar Termos Importantes*, que leva ao levantamento de termos do domínio sem a preocupação de classificá-los como conceito ou propriedade (NOY; MCGUINNESS, 2001).

A quarta etapa apresentada na Figura 2 é *Definir Classes e Hierarquias*, onde, a partir da listagem de termos, eliminam-se redundâncias e definem-se as classes da ontologia organizadas em uma hierarquia. A quinta etapa é *Definir Propriedades de Classes*, sendo definidas propriedades que complementam a descrição dos conceitos da ontologia. A sexta etapa é *Definir Restrições de Propriedades*, onde são estabelecidas restrições para cada propriedade. Por fim, é apresentada a etapa *Criar Instâncias da Ontologia*, que estabelece a criação de indivíduos para a ontologia (NOY; MCGUINNESS, 2001).

2.1.2 OWL

Ontologias fazem parte da pilha de padrões da *Word Wide Web Consortium* (W3C) para Web Semântica, sendo usadas para definir os vocabulários conceituais, nos quais dados são compartilhados entre aplicações, serviços de busca são disponibilizados, bases de conhecimento reutilizáveis são publicadas e serviços que facilitam a interoperabilidade entre múltiplos e heterogêneos sistemas e bases de dados são implementados (GRUBER, 2010).

A *Web Ontology Language* (OWL) é uma recomendação da W3C que objetiva prover uma representação eficiente para ontologias. Uma característica importante da linguagem, herdada de *Resource Description Framework* (RDF), é a capacidade de interligar ontologias distribuídas em diversos sistemas. Em outras palavras, uma determinada ontologia pode referenciar conceitos de outra ontologia. A OWL foi especificamente projetada para as necessidades da Web e Web Semântica (SHADBOLT; HALL; BERNERS-LEE, 2006).

OWL é uma linguagem utilizada para descrever um determinado domínio de conhecimento através de suas estruturas de representação. Pode-se dizer que a OWL é utilizada na formalização de um domínio através da definição de classes e de propriedades para estas classes, na definição de indivíduos e afirmações sobre as propriedades destes indivíduos, no raciocínio sobre essas classes e esses indivíduos de acordo com a semântica formal definida pela linguagem. A OWL é subdividida em três sublinguagens (MCGUINNESS; HARMELEN, 2010):

- *OWL Lite:* atende as necessidades básicas de classificação de hierarquias e restrições simples, tais como definição de cardinalidades. Dadas suas características, facilita a migração de taxonomias existentes, mas limita as capacidades da linguagem.
- *OWL Description Logic (DL):* provê maior expressividade que *OWL Lite* em termos de computação. Todas as conclusões são computáveis e todas as computações têm tempo finito. É assim chamada dada a correspondência com a lógica de descrição e inclui todas as construções da linguagem, porém impõe algumas restrições, tais como herança múltipla e classes como instância de outra classe.
- *OWL Full:* provê maior expressividade computacional que *OWL Lite* e a liberdade sintática existente no RDF, mas não há garantias computacionais nas construções.

OWL Full permite que uma ontologia aumente o significado do vocabulário RDF ou OWL.

Dada a relação de composição das sublinguagens, pode-se dizer que toda ontologia *OWL Lite* válida é uma ontologia *OWL DL* válida, consequentemente toda ontologia *OWL DL* válida é uma ontologia *OWL Full* válida. Também, pode-se afirmar que toda conclusão *OWL Lite* válida é uma conclusão *OWL DL* válida, logo, toda conclusão *OWL DL* válida é uma conclusão *OWL Full* válida. A Figura 3 apresenta esta relação de composição das sublinguagens OWL (MCGUINNESS; HARMELEN, 2010).

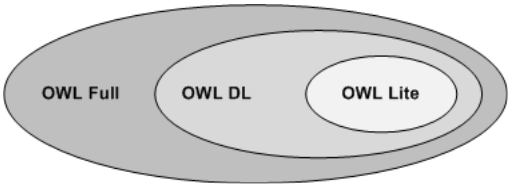

FIGURA 3 - Composição das sublinguagens OWL.

A Figura 4 mostra um fragmento de um arquivo OWL que define a ontologia de vinhos anteriormente apresentada. No primeiro bloco de código, pode-se notar a definição de uma propriedade do tipo objeto, na sequência se observa a definição de uma propriedade do tipo dados. O terceiro bloco define uma classe e todas as suas restrições. Por fim, o último bloco de código mostra a criação de indivíduos para ontologia.

```
...
<owl:ObjectProperty rdf:about="#adjacentRegion">
    <rdf:type rdf:resource="&owl;SymmetricProperty"/>
    <rdfs:range rdf:resource="#Region"/>
    <rdfs:domain rdf:resource="#Region"/>
</owl:ObjectProperty>
...
<owl:DatatypeProperty rdf:about="#yearValue">
    <rdfs:range rdf:resource="&xsd;positiveInteger"/>
    <rdfs:domain rdf:resource="#VintageYear"/>
</owl:DatatypeProperty>
...
<owl:Class rdf:about="#AlsatianWine">
    <owl:equivalentClass>
        <owl:Class>
            <owl:intersectionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#Wine"/>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="#locatedIn"/>
                    <owl:hasValue rdf:resource="#AlsaceRegion"/>
                </owl:Restriction>
            </owl:intersectionOf>
        </owl:Class>
    </owl:equivalentClass>
</owl:Class>
...
<owl:Thing rdf:about="#AlsaceRegion">
    <rdf:type rdf:resource="#Region"/>
    <locatedIn rdf:resource="#FrenchRegion"/>
</owl:Thing>
...
```

FIGURA 4 - Fragmento de código OWL.

2.1.3 Jena

Jena é um *framework* Java para programação de aplicações para a Web Semântica, o qual provê ambientes para programação de RDF, RDF *Schema* (RDFS) e OWL, além de incluir mecanismos de inferência baseados em regras. O *framework* foi desenvolvido pela *Hewlett-Packard Labs* com o objetivo de tornar fácil o desenvolvimento de aplicações que usem linguagens e modelos baseados na Web Semântica (MCBRIDE, 2002). Jena 1 foi lançado em 2000 e dentre suas diversas contribuições destacam-se as ferramentas para manipulação de grafos RDF e o suporte a ontologias através de DARPA *Agent Markup Language + Ontology Inference Layer* (DAML+OIL). O *framework* Jena 2 foi lançado em 2003 e se caracterizou pelo suporte a RDFS e OWL, além de prover mecanismos de inferência para ambas às linguagens (CARROLL et al., 2003).

Arquiteturalmente, Jena é organizado em uma estrutura de camadas, conforme visualizado na Figura 5. A camada principal do *framework* é a *Graph*, a qual é baseada na sintaxe abstrata de RDF e provê estruturas de armazenamento de triplas (sujeito + predicado + objeto), além de mecanismos de inferência para RDFS e OWL. A segunda camada é a *EnhGraph*, a qual se caracteriza por ser uma camada intermediária que provê pontos de extensão para visões de grafos (*Graph*) ou de nodos (*Node*) contidos em grafos. A última camada é a *Model* ou *Ontology Models*, a qual provê aos programadores as interfaces necessárias para o desenvolvimento de aplicações (CARROLL et al., 2003).

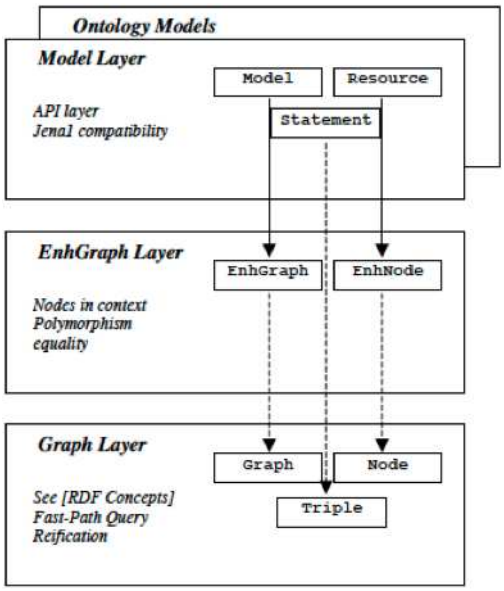

FIGURA 5 - Visão geral da arquitetura do *framework* Jena.

Outra característica importante do *framework* Jena é a disponibilização aos programadores de interfaces para motores de inferência, tanto para RDFS quanto para OWL.

Como pode ser observado na Figura 6, o mecanismo de inferência está baseado em uma estrutura chamada *Reasoner*, a qual combina um ou mais grafos em um grafo único e o disponibiliza para *InfGraph*. Através da camada *Ont/Model API* as consultas são realizadas no grafo resultante. Por fim, o *ModelFactory* serve como gerenciador, tanto da camada de consultas, quanto para a criação e registros dos motores de inferência (CARROLL et al., 2003).

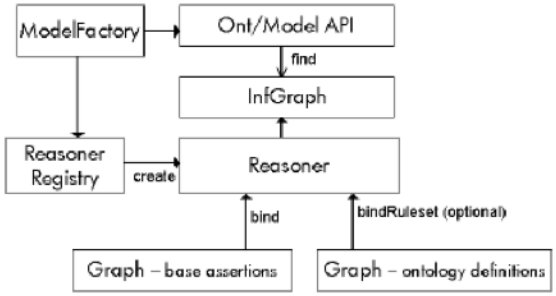

FIGURA 6 - Mecanismos de inferência do *framework* Jena.

2.1.4 SPARQL

SPARQL é uma linguagem inspirada na *Structured Query Language* (SQL), que viabiliza a busca de informações em grafos RDF. Uma característica importante da linguagem é a capacidade de compor consultas tendo vários modelos RDF como fonte de dados. Complementarmente, a SPARQL também viabiliza consultas em modelos especificados através de OWL, facilitando a recuperação de indivíduos e propriedades de uma ontologia. A Tabela 1 apresenta as principais cláusulas da SPARLQL (PRUD'HOMMEAUX; SEABORNE, 2010).

TABELA 1 - Principais cláusulas SPARQL.

Cláusula	Descrição	Sintaxe
PREFIX	Possibilita a criação de abreviaturas para referenciar a *Uniform Resource Identifier* (URI) dos recursos do grafo.	`PREFIX name: <http://uri.resource>`
SELECT	Possibilita a seleção das variáveis que se deseja obter como resultado da consulta.	`SELECT ?var_1 ?var_2 ?var_n`
FROM	Possibilita definir o grafo padrão para consulta.	`FROM <http://uri.resource>`
FROM NAMED	Possibilita que uma mesma consulta busque dados em mais de um grafo.	`FROM NAMED <http://uri.resource>`
WHERE	Possibilita a definição das regras e condições para atribuição de valores para as variáveis.	`WHERE {?data pre:resource ?var}.`
FILTER	Possibilita a filtragem do resultado, comparando variáveis com algum valor.	`FILTER (?var <= 10)` `FILTER (?var = 'SPARQL')`

A Figura 7 apresenta um exemplo de consulta simples em SPARQL estruturada da seguinte forma: na parte superior estão os dados do grafo RDF, no meio está a consulta SPARQL propriamente dita e na parte inferior está o resultado obtido. Observando a consulta, pode ser visto na cláusula *SELECT* a seleção da variável *?title*, a qual, na cláusula *WHERE*, é inicializada com dado recebido do recurso *<http://purl.org/dc/elements/1.1/title>* (PRUD'HOMMEAUX; SEABORNE, 2010).

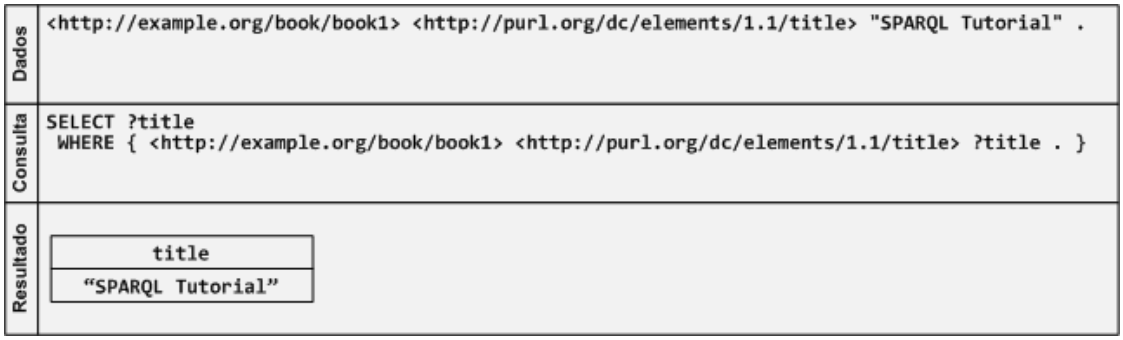

FIGURA 7 - Exemplo de uma consulta simples SPARQL.

A Figura 8 apresenta um exemplo de consulta SPARQL que retorna múltiplos valores como resultado. Inicialmente, fica caracterizada na consulta o uso da cláusula PREFIX, abreviando a URI do grafo RDF. Também pode ser vista na cláusula WHERE, a definição de duas regras para atribuição de valores, onde todos os registros de *?x* que possuírem nome e e-mail serão atribuídos às variáveis *?name* e *?mbox*, respectivamente (PRUD'HOMMEAUX; SEABORNE, 2010).

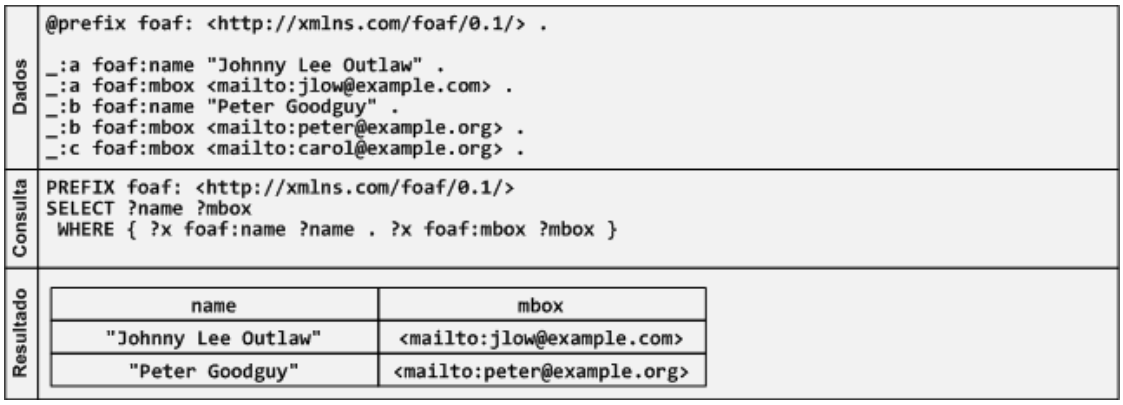

FIGURA 8 - Exemplo SPARQL com múltiplos retornos.

A Figura 9 tem como resultado o mesmo apresentado na Figura 7, porém este é obtido de uma forma mais elaborada, já que há mais dados no grafo da Figura 9 do que no grafo da Figura 7. Para se obter o mesmo resultado, além de se realizar a atribuição de recursos às variáveis na cláusula *WHERE*, também é necessário criar regras de filtragem de registros. Estas regras são criadas com a cláusula *FILTER*, inclusa na cláusula *WHERE* da consulta SPARQL. Pode-se notar o uso de uma expressão regular para filtrar o resultado da consulta,

mas esta não é a única forma, pois a cláusula *FILTER* permite ainda o uso de expressões aritméticas e relacionais (PRUD'HOMMEAUX; SEABORNE, 2010).

Dados

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/book/> .
@prefix ns: <http://example.org/ns#> .

:book1 dc:title "SPARQL Tutorial" .
:book1 ns:price 42 .
:book2 dc:title "The Semantic Web" .
:book2 ns:price 23 .
```

Consulta

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title
 WHERE { ?x dc:title ?title
         FILTER regex(?title, "^SPARQL") }
```

Resultado

title
“SPARQL Tutorial”

FIGURA 9 - Exemplo SPARQL com filtro de resultado.

2.2 AGENTES

Apesar de agentes ainda não possuírem uma definição de consenso geral na comunidade cientifica, uma das mais aceitas diz que estes são sistemas de computador situados em algum ambiente, capazes de tomar ações de forma autônoma dentro deste ambiente, a fim de atingir seus objetivos específicos. A Figura 10 mostra o conceito apresentando a relação dos agentes com o seu ambiente. Pode-se perceber que o arco de entrada é caracterizado por sensores que percebem o ambiente, enquanto que, o arco de saída é caracterizado por ações que transformam o ambiente (WOOLDRIDGE, 2002).

FIGURA 10 - Relação entre agente e ambiente.

Agentes, de forma complementar, ainda podem ser caracterizados por três propriedades específicas (SILVA, 2005):

- *Autonomia:* habilidade do software de agir independentemente, sem intervenção direta humana ou de outros agentes.
- *Pró-atividade:* agentes não agem simplesmente em resposta ao seu ambiente, eles são capazes de exibir pró-atividade, comportamento dirigido a objetivos e tomada de ações quando necessário.

- *Sociabilidade:* habilidade de participar de muitos relacionamentos, interagindo com outros agentes ao mesmo tempo ou em tempos diferentes.

A variedade de ambientes existentes é muito grande, porém pode-se identificar um número limitado de dimensões, das quais os ambientes podem ser organizados em categorias. Tais dimensões determinam o projeto apropriado de agentes e a aplicabilidade de cada uma das principais técnicas de implementação. A seguir são listadas as categorias de dimensões de ambientes (RUSSELL; NORVIG, 2004):

- *Completamente Observável versus Parcialmente Observável.* Um ambiente é completamente observável quando os agentes tem acesso ao estado completo do ambiente em todo instante de tempo. Por outro lado, um ambiente é parcialmente observável quando há ruídos, sensores imprecisos ou porque simplesmente não é possível conhecer algum estado.
- *Determinístico versus Estocástico.* O ambiente é determinístico quando seu próximo estado é completamente determinado pelo estado atual e pela ação do próprio agente, caso contrário ele é considerado estocástico. Em princípio, um agente não precisa se preocupar com a incerteza em um ambiente completamente observável e determinístico, mas um ambiente parcialmente observável pode parecer estocástico, principalmente se este for complexo, o que dificulta o controle dos aspectos não observados.
- *Episódico versus Sequencial.* Em ambiente episódico a experiência do agente é dividida em episódios atômicos, onde cada episódio consiste na percepção do agente e na execução de uma ação. Cada ação ou decisão é totalmente independente, ou seja, não é influenciado ou influencia outras ações ou decisões. Já em um ambiente sequencial, cada ação ou decisão tem impacto nas ações subsequentes do agente.
- *Estático versus Dinâmico.* Se um ambiente puder se alterar enquanto o agente está deliberando, este ambiente é dinâmico, caso contrário, ele é estático. Ambientes estáticos são mais fáceis de manipular, pois o agente não precisa continuar observando o mundo enquanto decide por uma determinada ação, enquanto que ambientes dinâmicos perguntam constantemente ao agente o que ele deseja fazer.
- *Discreto versus Contínuo.* A distinção entre ambiente discreto e contínuo é definida pelo modo que tempo é tratado, pelas percepções e pelas ações do agente. Um ambiente discreto possui um número finito de estados, percepções e ações,

enquanto que, em um ambiente contínuo não é possível enumerar os estados, percepções ou ações envolvidas.

- *Agente Único versus Multiagente.* A diferença entre ambientes de um único agente e multiagentes está na quantidade de agentes que interagem com o ambiente e entre si. Um ambiente de único agente possui um agente específico, o qual tem percepções e realiza ações para um determinado fim. Por outro lado, em um ambiente multiagente os agentes, além de possuir percepções e ações específicas, interagem entre si para atingir um objetivo comum.

Outra questão importante, a considerar no desenvolvimento de agentes, é a sua estrutura, ou seja, como estes são construídos e organizados internamente. Pode-se dizer que a estrutura de um agente é a união de uma *arquitetura* e um *programa*. O programa é a implementação da função que mapeia percepções e ações. Já a arquitetura são os sensores e atuadores que possibilitam interagir com o ambiente. A seguir, são listados os quatro tipos básicos de programas de agentes que incorporam a essência da maioria dos programas inteligentes (RUSSELL; NORVIG, 2004).

- *Agentes Reativos Simples.* É o tipo mais simples de agentes, os quais selecionam ações com base na percepção atual, ignorando o restante do histórico de percepções.
- *Agentes Reativos Baseados em Modelos.* O agente mantém estados internos dependentes do histórico de percepções, refletindo pelo menos algum dos estados não observáveis a partir do estado atual. Este conhecimento sobre os possíveis estados é chamado de modelo do mundo.
- *Agentes Baseados em Objetivos.* O fato de conhecer os estados possíveis nem sempre é suficiente para decidir o que fazer. É importante que o agente além do conhecer o universo de estados também tenha alguma informação sobre seus objetivos que descreva as situações desejáveis.
- *Agentes Baseados em Utilidade.* Somente os objetivos não são suficientes para gerar agentes de alta qualidade. Uma função de utilidade mapeia um estado, ou uma sequência de estados, em um valor que descreve o grau de "felicidade" do agente, assim este pode tomar suas decisões com base neste grau, buscando sempre estados os estados mais funcionais.

2.2.1 JADE

O *framework Java Agent Development* (JADE) foi criado em 1998 a partir de uma iniciativa *Telecom Italia*, motivada pela necessidade de validar a recém-criada especificação *Foundation for Intelligent Physical Agents* (FIPA). JADE teve seu código-fonte aberto em 2000 e começou a ser liberada sob a *Library Gnu Public Licence* (LGPL), encontrando-se atualmente na versão 3.6.1. JADE é uma plataforma de software que provê uma camada *middleware* de funcionalidades básicas, as quais são independentes de uma aplicação específica e podem simplificar a distribuição de soluções orientadas a agentes. Basicamente, o *framework* provê aos programadores o seguinte núcleo de funcionalidades (BELLIFEMINE; CAIRE; GREENWOOD, 2007):

- sistemas totalmente distribuídos habitados por agentes, onde cada agente roda em uma *thread* independente, potencialmente em máquinas remotas e diferentes;
- compatibilidade total com as especificações FIPA, participando de todos os eventos de interoperabilidade;
- transporte eficiente de mensagens assíncronas via uma interface de programação, onde a plataforma garante a escolha do melhor meio de comunicação;
- implementação de páginas brancas e páginas amarelas, onde sistemas podem ser implementados para representar domínios e subdomínios de um grafo de diretórios;
- um conjunto de ferramentas de suporte gráficas que facilitam o trabalho de depuração e monitoramento realizado pelos programadores;
- suporte às ontologias e linguagens de conteúdo, com verificação e codificação automática baseadas em *Extensible Markup Language* (XML) e RDF;
- bibliotecas de protocolos de interação que estabelecem modelos de comunicação orientados a realização de um ou mais objetivos.
- integração com diversas tecnologias orientadas a web, tais como: *servlets*, *applets* e *web services*; e suporte para plataformas móveis e ambientes com redes sem fio.
- interface de processos para controle de uma plataforma e seus componentes distribuídos fora da aplicação.
- *kernel* desenhado para permitir que programadores extendam o seu comportamento de acordo com suas necessidades específicas.

A Figura 11 apresenta uma síntese da arquitetura dos elementos da plataforma JADE, a qual é formada por contêineres de agentes que podem ser distribuídos por toda a rede.

Agentes vivem em um contêiner, sendo estes, processos Java providos pelo *run-time* JADE, tendo disponível todos os serviços necessários para sua execução. Um contêiner especial chamado *main* estabelece o ponto de partida da plataforma, sendo este o primeiro a ser executado e todos os demais são registrados nele. Também pode ser percebida, a interface FIPA provida pela plataforma aos agentes que nela rodam, sendo possível interagir com agentes oriundos de outras plataformas (BELLIFEMINE; CAIRE; GREENWOOD, 2007).

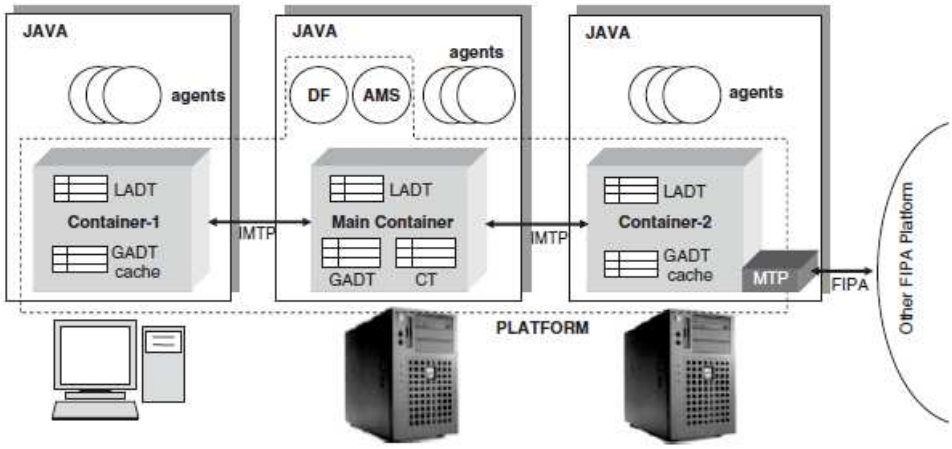

FIGURA 11 - Relacionamento entre os principais elementos arquiteturais JADE.

2.3 APLICAÇÕES WEB

Java Enterprise Edition (Java EE) é uma plataforma Java para desenvolvimento rápido e facilitado de aplicações empresariais, a qual provê para os desenvolvedores um conjunto de *Application Programming Interfaces* (API) que reduzem o tempo e a complexidade do desenvolvimento e aumentam o desempenho dos sistemas. A Java EE é mantida pela *Sun Microsystens* e, atualmente, as versões 5 e 6 são disponibilizadas pela empresa (SUN, 2007).

Dentre os *frameworks* providos pela plataforma, cabe destacar os que dão suporte ao desenvolvimento de aplicações web. Para Java EE, uma aplicação web é uma extensão dinâmica de aplicações servidoras, podendo ser: a) orientadas a apresentação, onde um conteúdo dinâmico é gerado e apresentado através de linguagens de marcação; b) orientadas a serviços, onde serviços específicos são disponibilizados aos seus clientes através de *web services* (SUN, 2007).

2.3.1 JSF

Java Server Faces (JSF) é um *framework* de componentes de interface de usuário no lado servidor, baseado no padrão de projeto *Model-View-Controller* (MVC), que provê as ferramentas necessárias para desenvolvimento de aplicações web. O MVC, demonstrado na Figura 12, é um padrão de projetos que organiza a arquitetura de uma aplicação em três estruturas distintas, onde a *Model*, representa o modelo de domínio da aplicação, definindo todo o universo de objetos necessários para a implementação das regras de negócio da aplicação. A *View* provê a visualização do modelo, através da implementação de interfaces com o usuário. Através destas interfaces o usuário pode manipular os objetos de domínio, definidos na *Model*. Por fim, a *Controller* garante a ligação entre a *Model* e a *View*. É através da *Controller* que se estabelecem os fluxos de navegação da *View* e quais objetos da *Model*, podem ser manipulados pelos usuários da aplicação (REENSKAUG, 1979).

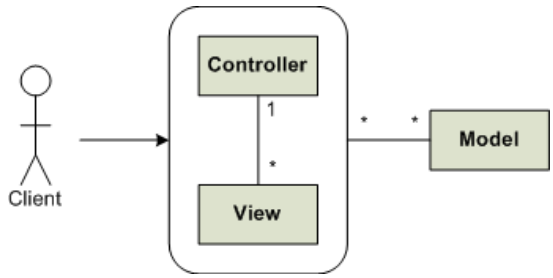

FIGURA 12 - O padrão de projeto MVC.

JSF tem como estruturas básicas uma API para representação de componentes de interface de usuário e manipulação de seus estados, e uma biblioteca de *tags Java Server Pages* (JSP) customizadas para expressar os componentes JSP básicos e interagir com os objetos no lado servidor. A Figura 13 apresenta o funcionamento básico do JSF, onde pode ser vista a caracterização do padrão MVC. No lado esquerdo, tem-se o cliente (*Browser*) e no lado direito o servidor (*Web Container*). O lado cliente faz uma requisição para o lado servidor, que através de suas estruturas de controle define o fluxo de paginas JSP para visualização e interage com os objetos de domínio do modelo. Após, os resultados gerados são retornados para o lado cliente (SUN, 2007).

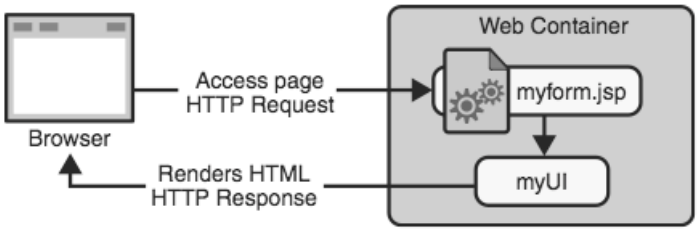

FIGURA 13 - Funcionamento básico do JSF.

2.4 FECHAMENTO DO CAPÍTULO

Ontologias é uma ferramenta robusta e eficaz para representação do conhecimento. Quando bem elaboradas, proveem aos sistemas inteligentes que as consomem uma grande capacidade de representação e inferência. Porém, sua construção não é trivial, pois demanda de uma grande capacidade de abstração do Engenheiro de Conhecimento, o qual muitas vezes, acaba modelando um determinado domínio de conhecimento como quem modela uma base de dados. De forma complementar, as ferramentas OWL, SPARQL e Jena se mostram plenas no que se refere à definição e manipulação de ontologias, entretanto, é necessário um conhecimento intermediário de XML e orientação a objetos.

Agentes é uma ótima ferramenta para construção de sistemas inteligentes, dadas as suas características de automação, pró-atividade e sociabilidade. Agentes bem projetados são eficazes, e é justamente o projeto de um agente o caminho crítico para o seu sucesso. A caracterização do ambiente e a escolha do tipo de programa para o agente é importante, pois uma escolha errada pode levar ao não atendimento dos objetivos pretendidos ou ao nível de complexidade de programação não justificável ao final da implementação. A plataforma JADE é robusta e prática para construção de agentes e, assim como Jena, demanda de bons conhecimentos sobre orientação a objetos.

A plataforma Java EE é uma solução plenamente consolidada para desenvolvimento de aplicações web. O uso de JSF impõe ao desenvolvedor a implementação do padrão de projeto MVC, o que sob a ótica da engenharia de software é uma grande vantagem, pois não é viável a tentativa de quebra do padrão. Por outro lado, esta inflexibilidade demanda de pleno entendimento, tanto do padrão, quanto da própria tecnologia, o que gera uma improdutividade inicial. Para compensar esta dificuldade, a plataforma dispõe de uma boa documentação de suporte.

3 GARANTIA DA QUALIDADE DE SOFTWARE APLICADA

A garantia da qualidade de software esta no caminho crítico desta dissertação, ou seja, tanto o problema levantado, quanto a solução apresentada tem esta disciplina como área fim. Em função disto, este Capítulo foi dedicado ao entendimento da garantia da qualidade de software, dos conceitos direta ou indiretamente relacionados e de como se pode implementar a disciplina.

O presente Capítulo apresenta na Seção 3.1 uma revisão sobre processo de software, considerando que este é o alicerce da garantia da qualidade. A Seção 3.2 apresenta os conceitos e definições de garantia da qualidade de software e como a disciplina é vista sob a ótica dos modelos de referência. A Seção 3.3 traz uma abordagem prática para a implementação da disciplina em um ambiente de desenvolvimento de software. A Seção 3.4 faz o fechamento do capítulo.

3.1 PROCESSO DE SOFTWARE

Abordagens de engenharia de software, essencialmente, apoiam-se em compromissos com a qualidade, tendo como principal alicerce uma camada de processos de software, os quais estabelecem *frameworks* capazes de efetivamente entregar o produto requerido (PRESSMAN, 2001). Refinando esta definição, pode-se dizer que um processo de software é um conjunto de atividades e resultados associados, os quais levam a um produto de software (SOMMERVILLE, 2001). A Figura 14 apresenta a estrutura de um *framework* de processo.

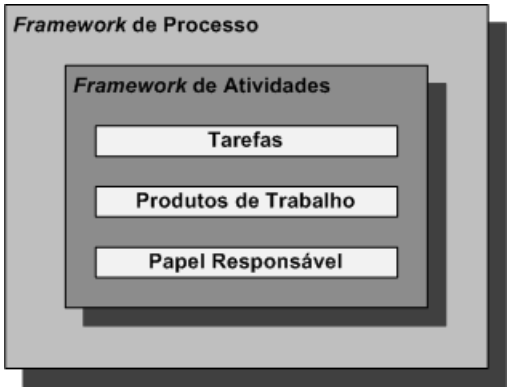

FIGURA 14 - Um *framework* genérico de processo.

O *framework* genérico de processo apresentado na Figura 14 é composto por *frameworks* de atividades aplicáveis a qualquer projeto de desenvolvimento de software. Cada *framework* de atividades é composto por uma coleção de tarefas e produtos de trabalho

(PRESSMAN, 2001). De forma complementar, cabe observar que as tarefas e produtos de trabalho estão diretamente relacionados a um papel responsável.

O *Capability Maturity Model Integration* (CMMI) é um modelo de referência altamente conceituado no mundo corporativo, que relaciona um conjunto de práticas orientadas à qualidade de software. O modelo classifica os processos de software segundo sua maturidade, a qual remete a melhoria contínua dos processos da organização através de um conjunto de áreas de processo, *Process Area* (PA), organizados segundo critérios definidos pelo modelo (CHRISSIS; KONRAD; SHRUM, 2007).

A Figura 15 mostra a hierarquia de níveis de maturidade, *Maturity Level* (ML) estabelecido pelo CMMI. O nível *Inicial* se caracteriza por processos *ad hoc* que tende ao caos, onde o sucesso em projetos de desenvolvimento, quando obtido, é reflexo de esforços individuais. O nível *Gerenciado* garante que o projeto de desenvolvimento é planejado e executado, onde os recursos têm as habilidades necessárias e a aderência ao processo é garantida. O nível *Definido* é caracterizado por padrões, procedimentos, ferramentas e métodos, onde cada projeto estabelece seu processo a partir do processo padrão da organização (CHRISSIS; KONRAD; SHRUM, 2007).

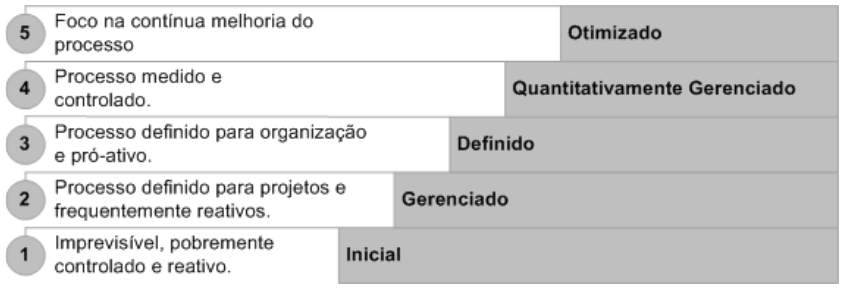

FIGURA 15 - Os cinco níveis de maturidade de um processo.

Ainda é apresentado na Figura 15 o nível *Quantitativamente Gerenciado*, que se caracteriza pelo pleno uso de medidas no planejamento e monitoramento, onde objetivos quantitativos são definidos com base em critérios orientados a qualidade e produtividade do processo. Por fim, o nível *Otimizado* é caracterizado pela melhoria contínua baseada no pleno entendimento quantitativo do processo, ou seja, processos estratégicos são identificados, medidos e melhorados (CHRISSIS; KONRAD; SHRUM, 2007).

3.2 GARANTIA DA QUALIDADE DE SOFTWARE

Assumindo a premissa que a qualidade de um produto de software é altamente influenciada pela qualidade do processo que construiu este software, faz-se necessário lançar

mão de mecanismos de monitoramento capazes de dar uma clara visão sobre a execução dos processos de software. O *Software Engineering Body of Knowledge* (SWEBOK) diz que o papel da garantia da qualidade é se certificar que um processo é executado como planejado, provendo para a organização um conjunto de indicadores de processo. Em outras palavras, a garantia da qualidade é responsável por monitorar a execução do processo, e os resultados gerados a partir deste monitoramento devem ser externados para as gerências responsáveis pela execução de projetos de software (ABRAN et al., 2004).

Para fins de desambiguação, cabe diferenciar o conceito de garantia da qualidade do conceito de controle da qualidade. A garantia da qualidade é formada por um conjunto sistemático e planejado de atividades que, quando seguidas, garantem aos clientes que os produtos ou serviços atendem às suas expectativas. Pode-se dizer então, que garantia da qualidade remete ao como fazer o software, que por sua vez remete ao conceito de processo de software. Por outro lado, o controle da qualidade é a comparação de um determinado produto ou serviço com suas respectivas especificações, onde todo e qualquer desvio identificado é devidamente registrado e endereçado. O controle da qualidade é uma atividade inserida no processo de software que garante o pleno funcionamento do produto ou serviço de acordo com os requisitos previamente levantados (BASTOS et al., 2007).

O CMMI trata a garantia da qualidade como uma PA, exigida para se atingir o nível 2 de maturidade. A garantia da qualidade de processo e de produto, *Process and Product Quality Assurance* (PPQA), é uma PA de suporte do CMMI, que provê às gerências da organização uma visão objetiva sobre a execução dos processos e produtos de trabalho relacionados. Cabe observar que não se deve confundir produto de software com produto de trabalho. Produto de software é a meta do projeto, ou seja, é o programa ou sistema a ser desenvolvido pela equipe, enquanto que, produto de trabalho são todos os artefatos meio do projeto, ou seja, documentos de gerenciamento e engenharia, tais como: Plano de Projetos, Modelo de Casos de Uso, etc. Fundamentalmente, PPQA envolve a avaliação objetiva da execução dos processos e produtos de trabalho contra descrições de processo, procedimentos e padrões (CHRISSIS; KONRAD; SHRUM, 2007).

PPQA busca também, a identificação e o acompanhamento de não conformidades, garantindo seu devido fechamento. Não conformidades são desvios na execução do processo que podem expor o projeto de desenvolvimento a riscos relacionados ao escopo, custo, tempo ou qualidade. Garantir o fechamento significa monitorar e apoiar a equipe de desenvolvimento na resolução do problema e não realizar a correção propriamente dita. Por fim, PPQA explicita informações sobre a aderência dos projetos de desenvolvimento aos

processos definidos, provendo a fundamentação necessária para a tomada de decisão por parte das gerências competentes, avaliação e melhoria contínua dos processos de desenvolvimento (CHRISSIS; KONRAD; SHRUM, 2007).

Como todas as demais PAs do CMMI, PPQA é organizada em objetivos específicos, *Specific Goal* (SG), que por sua vez são organizados em práticas específicas, *Specific Practice* (SP), como visto na Figura 16. O primeiro objetivo definido (SG 1) diz respeito à avaliação objetiva da aderência dos processos e produtos de trabalho elaborados contra descrições de processo, procedimentos e padrões. Este objetivo é composto pelas práticas para avaliação objetiva de processos (SP 1.1) e de produtos de trabalho e serviços (SP 1.2). O segundo objetivo (SG 2) propõe uma percepção clara e objetiva do projeto em termos de desvios e suas correções. O objetivo é composto pela prática de comunicação e monitoramento das não conformidades até que estas sejam fechadas (SP 2.1) e pela prática de geração de registros para uso como base de conhecimento em ações de melhoria contínua (SP 2.2) (CHRISSIS; KONRAD; SHRUM, 2007).

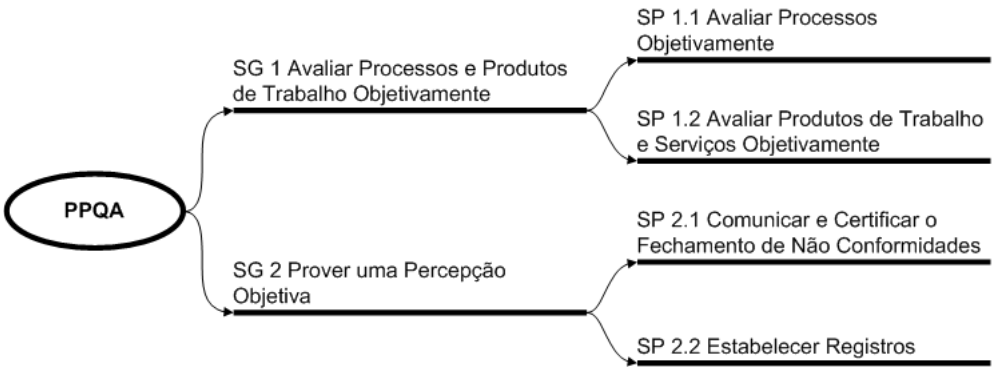

FIGURA 16 - Estrutura hierárquica de PPQA.

É importante não confundir os objetivos da PA de garantia da qualidade com as PAs de verificação, *Verification* (VER), e validação, *Validation* (VAL), também definidas pelo CMMI. Ao contrário de PPQA, VER e VAL são PAs orientadas ao produto de software e não ao processo de software. VER tem como propósito garantir que os produtos estejam aderentes as suas especificações, e VAL tem como propósito garantir que os produtos sejam funcionais quando inseridos em seu ambiente real de uso (CHRISSIS; KONRAD; SHRUM, 2007).

3.3 ESTRATÉGIA DE IMPLEMENTAÇÃO DE PPQA

PPQA define um conjunto de objetivos e práticas considerados importantes para estabelecer a capacidade e a maturidade de uma organização, no que se refere à disciplina de

garantia da qualidade. Como em outros modelos de referência, o CMMI não estabelece como as práticas da PA podem ser de fato implementadas em uma organização. Cabe à própria organização interpretar as práticas e definir como estas podem de fato ser implementadas. Tal necessidade, demanda dos envolvidos conhecimentos e capacidades de engenharia e de gerenciamento.

Como alternativa para implementação de PPQA, é estabelecida uma estratégia fundamentada em três princípios. O primeiro diz respeito ao pleno atendimento dos objetivos da PA, garantindo o sucesso em avaliações oficiais para obtenção de laudos de maturidade CMMI. O segundo se refere à relação custo/benefício, buscando minimizar o esforço necessário para execução das atividades sem perder a cobertura necessária para garantir o monitoramento do processo. O terceiro diz respeito à percepção da equipe de desenvolvimento sobre a importância da PA, a qual deve sustentar a garantia da qualidade como uma ferramenta de suporte ao desenvolvimento.

Podem ser observadas na Figura 17, três frentes distintas para implementar PPQA. A primeira remete ao mapeamento de pontos críticos de um processo através do levantamento dos principais produtos de trabalho, e a geração de uma coleção de itens de revisão que garantam a verificação dos fatores de sucesso para o projeto. A segunda remete à execução das inspeções em projetos através da definição de escopo, seleção e análise de amostras, além de corroboração e publicação dos resultados. A terceira estabelece dois indicadores fundamentais para o monitoramento, tanto dos projetos de desenvolvimento, quanto das próprias atividades de garantia da qualidade.

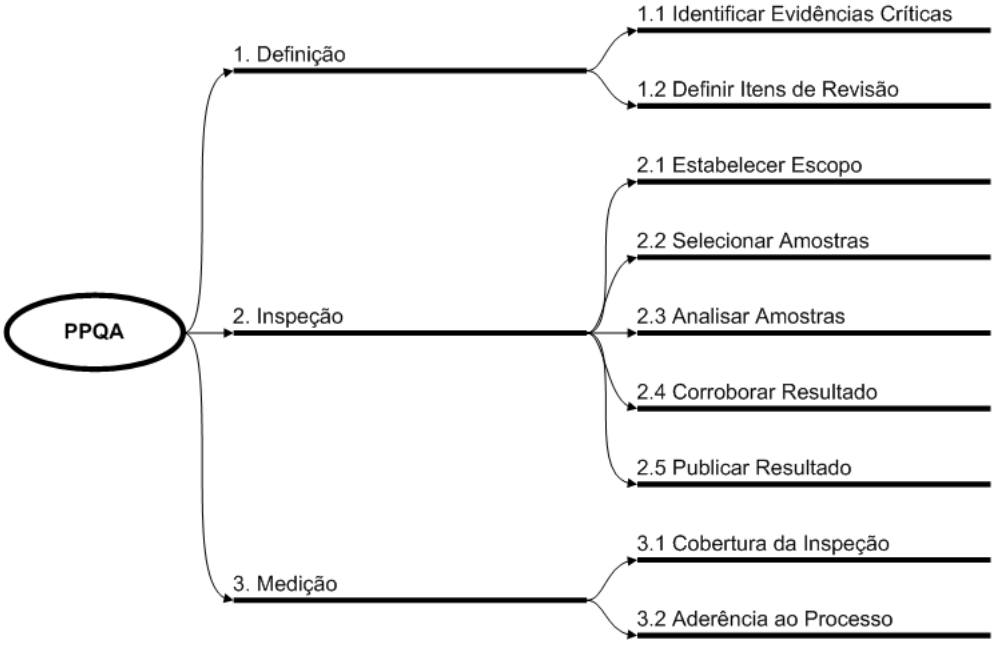

FIGURA 17 - Visão geral da estratégia de implementação de PPQA.

3.3.1 Definição

A primeira frente apresentada diz respeito às definições básicas realizadas antes de dar início às inspeções de projeto propriamente ditas. Inicialmente, cabe observar que as atividades de definição não são atividades recorrentes como as atividades de inspeção de projetos. Uma vez estabelecidas às definições necessárias, estas tendem a se estabilizar. Entretanto, como previsto pelo CMMI, tais definições devem ser revisadas sempre que ocorrer algum tipo de modificação significativa nos processos, padrões ou procedimentos da organização.

3.3.1.1 Identificar Evidências Críticas

A identificação de evidências críticas é uma atividade fundamental para as definições básicas da implementação de PPQA, pois uma forma de se conseguir uma boa relação custo/benefício é identificar os artefatos que estão no caminho crítico da qualidade do projeto, ou seja, documentos que se construídos de forma incorreta impactam diretamente na qualidade do produto solicitado pelo cliente. Por exemplo, se uma Lista de Requisitos não for devidamente estruturada e escrita, a equipe de desenvolvimento pode não entender o escopo e construir algo não solicitado, o que impacta no orçamento do projeto. Para identificar as evidências críticas, faz-se necessário um estudo, tanto do processo de desenvolvimento, quanto do histórico de projetos da organização. Assim, torna-se possível mapear quais e quando os produtos de trabalho devem ser devidamente inspecionados.

3.3.1.2 Definir Itens de Revisão

Uma vez mapeados os produtos de trabalho pertencentes ao caminho crítico da qualidade, faz-se necessário enumerar para cada um, todas as características que garantam a sua qualidade. Se nas inspeções for possível identificar tais características nos artefatos gerados, pode-se concluir que estes estão aderentes ao processo definido. Cada característica enumerada vira um item de revisão, o qual sempre deve ter uma avaliação binária: Atende ou

Não Atende. Por exemplo, pode-se enumerar as seguintes características, consequentemente itens de revisão, de um Plano de Gerenciamento de Riscos:

- os riscos identificados estão descritos de forma clara e objetiva;
- cada risco está devidamente qualificado;
- está definido para cada risco uma ação de resposta devidamente planejada;
- para cada risco está definida pelo menos uma ação de contingência.

3.3.2 Inspeção

A segunda frente estabelecida caracteriza as inspeções dos projetos de software. Dados os objetivos de uma implementação de PPQA, esta se torna a frente mais importante, pois é nela que todas as práticas da PA são plenamente atendidas. Neste contexto, pode-se dizer que Definição e Medição são frentes de apoio à Inspeção. As atividades definidas dentro de Inspeção a tornam capaz de dar aos interessados uma plena e transparente visão da execução dos projetos de desenvolvimento, no que se refere à aderência aos processos definidos pela organização. Cabe ainda salientar, que a inspeção é altamente recorrente, ou seja, são aplicadas em um projeto quantas vezes for necessário para garantir a execução do processo.

3.3.2.1 Estabelecer Escopo

É natural pensar que os projetos executam seus processos de forma contínua, logo, as inspeções devem cobrir o exato ponto de execução do processo. Para tal, faz-se necessário avaliar o status do projeto e decidir quais itens de revisão são aplicáveis neste momento. Esta é uma atividade repetida a cada inspeção e de rápida execução, pois o maior esforço deve ser concentrado na inspeção propriamente dita. Para exemplificar, suponha-se que um projeto, ao executar o *Unified Process* (LARMAN, 2004), esteja na fase de Construção, logo, não faz sentido aplicar itens de revisão que cobrem a fase de Concepção, pois isto é algo já superado pelo projeto e eventuais correções em artefatos desta fase não agregam mais valor.

3.3.2.2 Selecionar Amostras

Uma vez estabelecido o escopo da inspeção, faz-se necessário colher amostras dos produtos de trabalho para efetivamente executar a inspeção. Nesta atividade, cabe a aplicação de técnicas de amostragem para garantir um resultado plausível com a realidade do projeto. Por exemplo, ao buscar amostras de Atas de Reunião, poderia ser adotada uma amostragem estratificada por datas, sendo selecionado o artefato mais antigo, o mediano e o mais novo. Já, para a seleção de amostras de Diagramas de Caso de Uso, a amostragem pode ser estratificada por pacote e redator, colhendo um artefato de cada pacote para cada Analista de Sistemas envolvido no projeto.

3.3.2.3 Analisar Amostras

A análise das amostras colhidas caracteriza a inspeção propriamente dita. É neste momento que se busca no artefato as características de qualidade, itens de revisão, enumeradas na Definição. Ao ponderar sobre cada item de revisão se deve chegar a uma conclusão sobre seu atendimento. É importante observar que a decisão de Atende ou Não Atende, é sempre tomada sobre o conjunto de amostras, ou seja, caso uma das amostras não atenda ao item de revisão em questão, todas as demais são desqualificadas e o item é considerado Não Atende. Por outro lado, se todas as amostras atenderem ao item de revisão este é considerado Atende. Tal critério garante a avaliação objetiva requerida pelo CMMI para o atendimento das práticas SP 1.1 e SP 1.2 apresentadas na Seção 3.2.

3.3.2.4 Corroborar Resultado

A equipe de desenvolvimento não deve ter a sensação de que o resultado de uma inspeção é algo externo ao projeto e imposto à equipe. Tal situação leva a PA ao descrédito e, consequentemente, vira custo sem agregar valor. Uma forma simples de contornar este problema é corroborar o resultado junto aos representantes da equipe, tais como: Gerente de Projeto, Líder Técnico, etc. Ao apresentar o resultado, deve-se obter o consenso de todos, sempre usando uma abordagem argumentativa. Isto reforça a necessidade de ter como

Analista de Qualidade um profissional com conhecimento e experiência em engenharia e gerenciamento.

3.3.2.5 Publicar Resultado

Como exigido pelo CMMI, o resultado das inspeções deve ser de conhecimento de todos os envolvidos, logo, faz-se necessário publicá-lo para a equipe do projeto e para as gerências diretamente superiores ao projeto. Assim, todos tem ciência sobre os riscos levantados, sobre as não conformidades endereçadas e sobre as suas responsabilidades para com o processo.

3.3.3 Medição

A frente de Medição dá suporte à frente de Inspeção, pois nela são definidos dois indicadores básicos para avaliação dos projetos e suas inspeções. O CMMI recomenda o uso de medidas para monitorar processos e projetos. No que se refere à PPQA, a lógica é a mesma, ou seja, faz-se necessário ter um conjunto mínimo de indicadores capazes de prover uma objetiva e clara percepção sobre a cobertura obtida durante as inspeções e sobre a aderência do projeto ao processo definido.

3.3.3.1 Cobertura da Inspeção

A cobertura da inspeção indica o quão boa foi a Definição, que estabelece o conjunto de itens de revisão, e o quão criterioso foi o Analista de Qualidade ao definir o escopo, pois este é o responsável por identificar itens aplicáveis e não aplicáveis. O cálculo deste indicador se dá através da razão entre o Total de Itens de Revisão Aplicáveis (TIRA) e o Total de Artefatos Inspecionáveis (TAI). Para entender o número gerado, é preciso saber que quanto maior for o grau obtido melhor está a cobertura da inspeção em questão. Um projeto que tenha obtido grau 4 tem melhor cobertura que um que tenha obtido grau 2, porque o primeiro tem em média 4 itens de revisão para cada artefato, enquanto que, o segundo tem em média 2 itens de revisão para cada artefato. A Tabela 2 exemplifica este cenário.

TABELA 2 - Exemplo de cálculo de cobertura da inspeção.

Projeto 1		**Projeto 2**	
TIRA	40	TIRA	20
TAI	10	TAI	10
TIRA / TAI	04	TIRA / TAI	02

3.3.3.2 Aderência ao Processo

A aderência ao processo indica o quão próximo está o projeto da execução que o processo entende como ideal. Este indicador gera um valor percentual, onde 100% indica total aderência e 0% indica nenhuma aderência. Para calcular, faz-se necessário obter a razão entre o Total de Itens de Revisão Atendidos (TIRAt) e o Total de Itens de Revisão Aplicáveis (TIRAp). Um projeto que tenha em seu escopo de inspeção 10 itens de revisão e tenha recebido Atende em 5 itens de revisão, tem como grau de aderência o valor de 0,5, ou seja, 50% de aderência ao processo. A Tabela 3 apresenta um exemplo para o cálculo deste indicador.

TABELA 3 - Exemplo de cálculo de aderência ao processo.

TIRAt	**TIRAp**	**TIRAt / TIRAp**
05	10	0,5

3.4 FECHAMENTO DO CAPÍTULO

O presente Capítulo foi aberto apresentando uma visão geral de processo de software. Tal contextualização é pertinente quando se entende que o objetivo da garantia da qualidade de software é justamente garantir a execução de processos de desenvolvimento quaisquer que sejam. Na sequência, a garantia da qualidade de software é amplamente discutida, o que é pertinente dado os objetivos deste trabalho. Cabe ressaltar a importância de se entender a diferença entre garantia e controle de qualidade, pois não é raro ter confusões neste sentido.

A grande contribuição do Capítulo está na estratégia de implementação de garantia da qualidade, pois ela foi elaborada com base em necessidades reais de uma organização de desenvolvimento de software e traz detalhes práticos sobre o atendimento dos objetivos e práticas definidas pelo CMMI para PPQA. Os resultados gerados por esta estratégia, apresentados subsequentemente neste trabalho, são base comparativa para os resultados obtidos com a pesquisa desenvolvida e apresentada ao longo desta dissertação.

4 TRABALHOS RELACIONADOS

Com o objetivo de validar o tema de pesquisa desta dissertação, foi realizada uma pesquisa por trabalhos relacionados em bases de publicações[2] conceituadas no âmbito da ciência da computação. Considerando o universo de publicações encontradas, foram selecionadas as cinco mais relevantes para os objetivos deste trabalho, as quais de alguma forma influenciaram o desenvolvimento desta pesquisa.

Este Capítulo apresenta na Seção 4.1 uma tese de doutorado que define uma ontologia para processo de software. Na Seção 4.2 é apresentado um artigo que estabelece uma ontologia para modelos de processos de software. A Seção 4.3 traz um artigo que define um padrão para representação de conhecimento de processos de software. A Seção 4.4 descreve um artigo que trata de agentes inteligentes baseados em ontologias de PPQA. Na Seção 4.5 é apresentado um artigo sobre *web services* para avaliações de garantia da qualidade. A Seção 4.6 apresenta uma análise comparativa entre os trabalhos relacionados. Por fim, a Seção 4.7 faz o fechamento do capítulo.

4.1 ONTOLOGIA DE PROCESSO DE SOFTWARE

De todos os trabalhos analisados para o desenvolvimento desta dissertação, este é o pioneiro no uso de ontologias como ferramenta base para engenharia de software. A tese em questão objetiva estruturar a aquisição, organização, reuso e compartilhamento de conhecimento sobre processo de software. Para tal, foi estabelecida uma ontologia para prover um vocabulário e um conjunto de axiomas que fixam a semântica dos termos neste domínio. Uma das principais características da ontologia proposta é dar suporte ao desenvolvimento de ferramentas baseadas no conhecimento sobre processos de software, ou seja, otimizar o tempo do desenvolvedor através da automação de tarefas específicas (FALBO, 1998).

A ontologia de processo de software proposta está fundamentada nos principais conceitos relacionados ao domínio do conhecimento, sendo estes: *atividades*, as tarefas a serem realizadas; *artefatos*, os produtos de software produzidos; *procedimentos*, condutas bem estabelecidas e ordenadas para execução das atividades; *recursos*, pessoas, ferramentas ou equipamentos necessários para execução das atividades; *processos*, coleções de atividades

[2] Ver http://ieeexplore.ieee.org/Xplore/guesthome.jsp.

relacionadas executadas durante o desenvolvimento do produto. Estruturalmente a ontologia de processo de software é decomposta em três ontologias de base (FALBO, 1998):

- *Ontologia de Atividade.* Considerada a mais relevante por se tratar do cerne de qualquer modelo de processo de software. Entende-se que atividades podem ser compostas por outras atividades, ou seja, ocorrer em vários níveis de abstração. A Figura 18 (A) apresenta a taxonomia definida para a ontologia de atividade.
- *Ontologia de Procedimento.* Estabelece os conceitos relacionados à execução de uma atividade dentro de um processo de software. Também vale ressaltar que a ontologia de procedimento leva em consideração na sua definição, tanto a tecnologia, quanto o paradigma utilizado no processo de software. A Figura 18 (B) apresenta a taxonomia definida para a ontologia de procedimento.
- *Ontologia de Recurso.* Define uma característica derivada do papel que um elemento do domínio de conhecimento desempenha em uma atividade, logo, as propriedades de um recurso são determinadas pelas atividades relacionadas, o que estabelece uma forte ligação da ontologia de recurso com a de atividade. Em outras palavras, a ontologia de recurso define conceitos relacionados aos elementos agentes que atuam ou apoiam a realização de uma atividade. A Figura 18 (C) apresenta a taxonomia da ontologia de recursos.

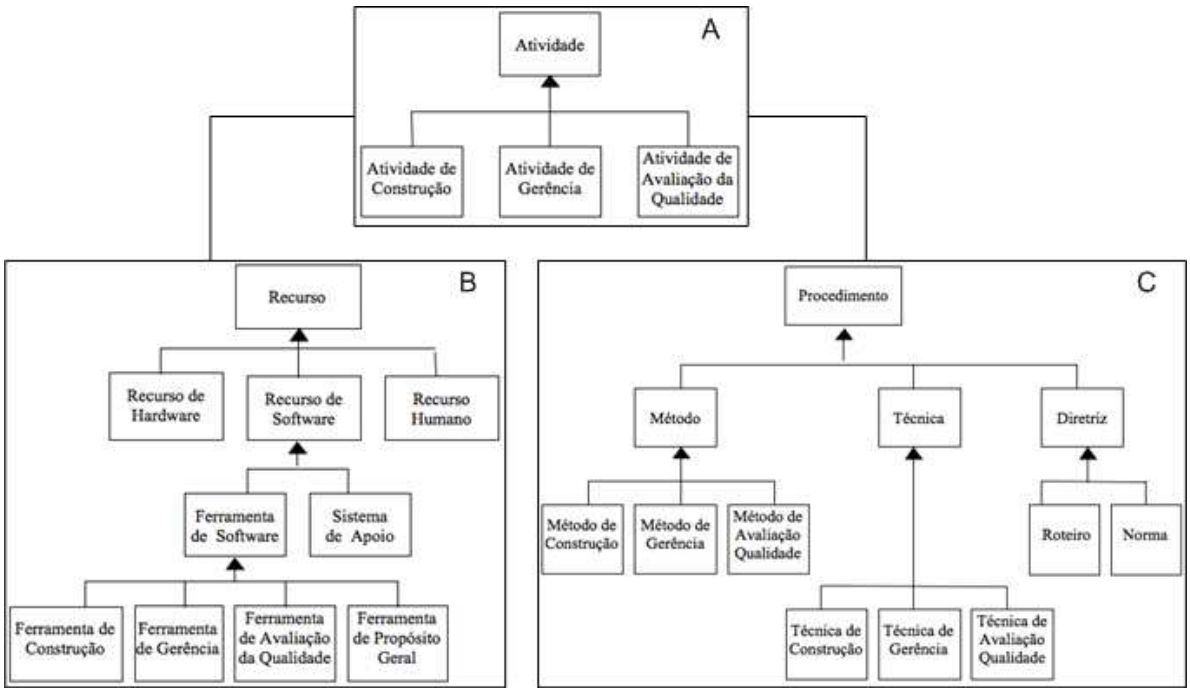

FIGURA 18 - Visão geral da ontologia de processo de software.

4.2 ONTOLOGIA DE MODELOS DE PROCESSO DE SOFTWARE

O presente artigo ressalta a existência de diversos modelos de referência de qualidade de software, tais como o CMMI e a *International Organization for Standardization / International Electrotechnical Commission* (ISO/IEC) 15504, dentre outros. Esses modelos conduzem as organizações na melhoria de seus processos de software, buscando sempre a maximização da qualidade dos softwares desenvolvidos. Entretanto, o uso de tais modelos nas organizações é dificultado devido as suas características, propósitos, orientações e estruturas. A seguir são apresentados alguns dos problemas encontrados ao implantar modelos de referência de qualidade de software (LIAO; QU; LEUNG, 2005):

- *Descrição Formal de Modelo de Processo.* Os modelos de processos existentes são modelos empíricos e descritivos, sem uma estrutura formal de representação.
- *Compatibilidade e Transformabilidade.* Modelos não compatíveis e sem capacidade de transformação causam muitos problemas para as organizações no que se refere à análise e modelagem de processos.
- *Comparação entre Atributos de Processos.* Análise quantitativa e comparação de atributos de processo são fundamentais para avaliar um modelo. Porém, a coleta de dados para as análise necessárias é dificultada pela falta de padrão entre os modelos.

Com base neste cenário, é proposta uma abordagem baseada em ontologias para expressar processos de software em um nível conceitual. A *Software Process Ontology* (SPO) é semanticamente forte, definindo a estrutura de modelos de processo em nível de esquema. A Figura 19 apresenta um grafo RDF da ontologia, onde podem ser observadas as sua capacidades no que diz respeito à representação de modelos de referência de qualidade.

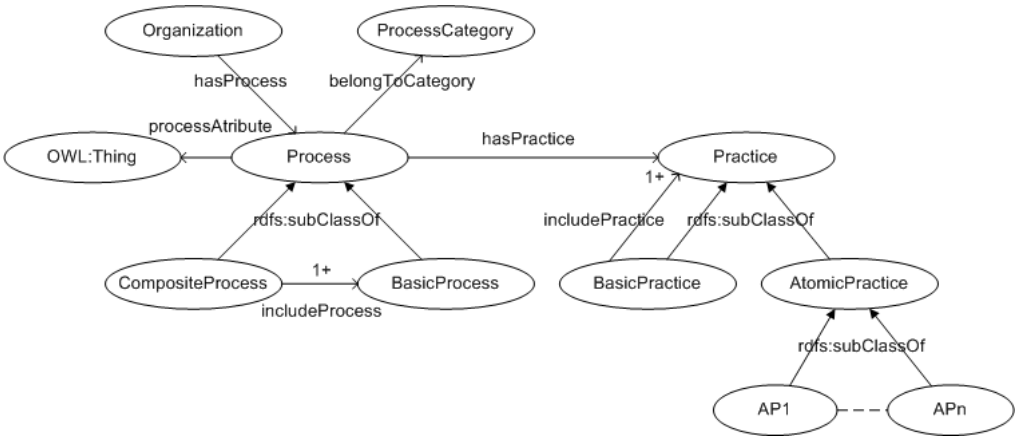

FIGURA 19 - Ontologia de modelos de processo de software.

A Figura 19 traz o processo (*Process*) como conceito central. Cada processo é de uma organização (*Organization*), pertence a uma categoria (*ProcessCategory*) e possui práticas específicas (*Practice*). Os processos são especializados em básicos (*BasicProcess*) e compostos (*CompositeProcess*). As práticas também são especializadas em básicas (*BasicPractice*) e atômicas (*AtomicPractice*) (LIAO; QU; LEUNG, 2005).

4.3 REPRESENTAÇÃO DO CONHECIMENTO EM PROCESSO DE SOFTWARE

O artigo em questão registra a constatação de que, mesmo com os esforços científicos até então publicados, muito trabalho está por fazer no que se refere à representação de conhecimento em processos de software. O artigo traz a tona algumas limitações relacionadas à representação de conhecimento de software, enumerando os principais problemas (HE et al., 2006):

- *Incompleteza.* A maioria dos estudos é fundamentada em experiências, entretanto o conhecimento de processo de software deveria ser sistematicamente analisado e coletado.
- *Ambiguidade de Tipos de Conhecimento de Processo de Software.* Conhecimento de processo de software pode ser dividido em: experiências, artefatos e habilidades. Muitos estudos são fundamentados com base em dois tipos, falhando em relação ao terceiro.
- *Efetividade de Ferramentas de Suporte.* Representar processos sem o ferramental necessário é uma árdua tarefa. Prover o suporte necessário é um grande desafio.

Apoiado por esta constatação foi proposto um *framework* chamado *Ontology Supported Software Process Knowledge Representation* (OnSSPKR) que, além de ser capaz de representar o conhecimento de processo de software, é capaz de prover suporte para auditorias em projetos existentes e para gerentes de projeto no planejamento de novos projetos. A Figura 20 mostra o grafo da ontologia de processo baseada em experiências, onde se observa o processo (*Process*) como conceito central, o qual é de uma organização (*Organization*) e pertence a uma categoria (*ProcessCategory*). O conceito de processo é especializado em marcos, arquitetura e fase (*Milestone*, *Architecture* e *Phase*). Outra especialização de processo é em atividade (*Activity*), a qual também é especializada em atômica e básica (*AtomicActivity* e *BasicActivity*) (HE et al., 2006).

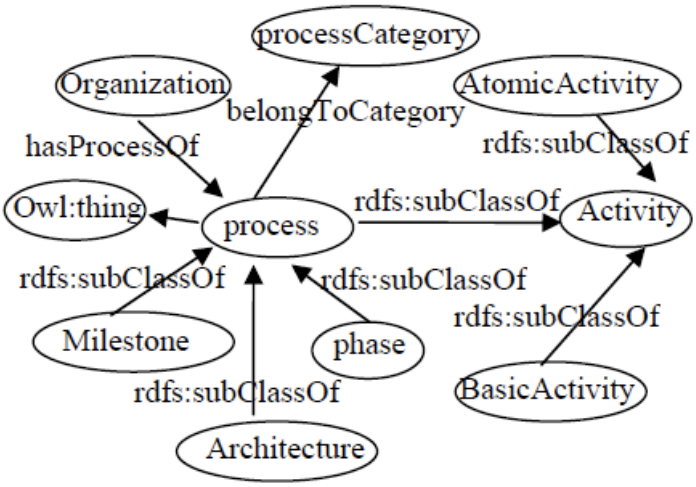

FIGURA 20 - Representação de conhecimento em processos de software.

4.4 AGENTE INTELIGENTE BASEADO EM UMA ONTOLOGIA DE PPQA

A proposta deste artigo é introduzida relatando que os maiores problemas encontrados em projetos de software estão relacionados a estouro de orçamento, atrasos e baixa qualidade nas entregas. Também se afirma que a qualidade de um produto de software é dependente da qualidade do processo que o construiu. Em função disto, a PA do CMMI referente à garantia da qualidade de processo e de produto, PPQA, é importante e estratégica para as organizações, pois através dela é possível viabilizar um incremento controlado e contínuo na qualidade dos produtos de software (WANG; LEE, 2007).

Fundamentado na PA de PPQA do CMMI, o artigo apresenta uma proposta de implementação de um agente inteligente *fuzzy*, baseado em uma ontologia de PPQA, que objetiva dar suporte para avaliações de níveis de maturidade CMMI. O agente é estruturalmente composto por um mecanismo de processamento de documentos, mecanismo de raciocínio e mecanismos de sumarização. A ontologia de PPQA é baseada nos conceitos fundamentais da PA e está organizada em cinco camadas conceituais, conforme mostrado pela Figura 21 (WANG; LEE, 2007).

A Figura 21 apresenta em sua parte superior a *Camada Quem*, onde conceitos relacionados aos papéis do processo são definidos. A próxima é a *Camada Quando*, que estabelece os diversos momentos do processo de desenvolvimento. Na sequencia, tem-se a *Camada O que*, que define os produtos de trabalho do processo mapeado. Por fim, é apresentada a *Camada Onde* e a *Camada Como*, que estabelecem os ambientes do processo e práticas do processo, respectivamente (WANG; LEE, 2007).

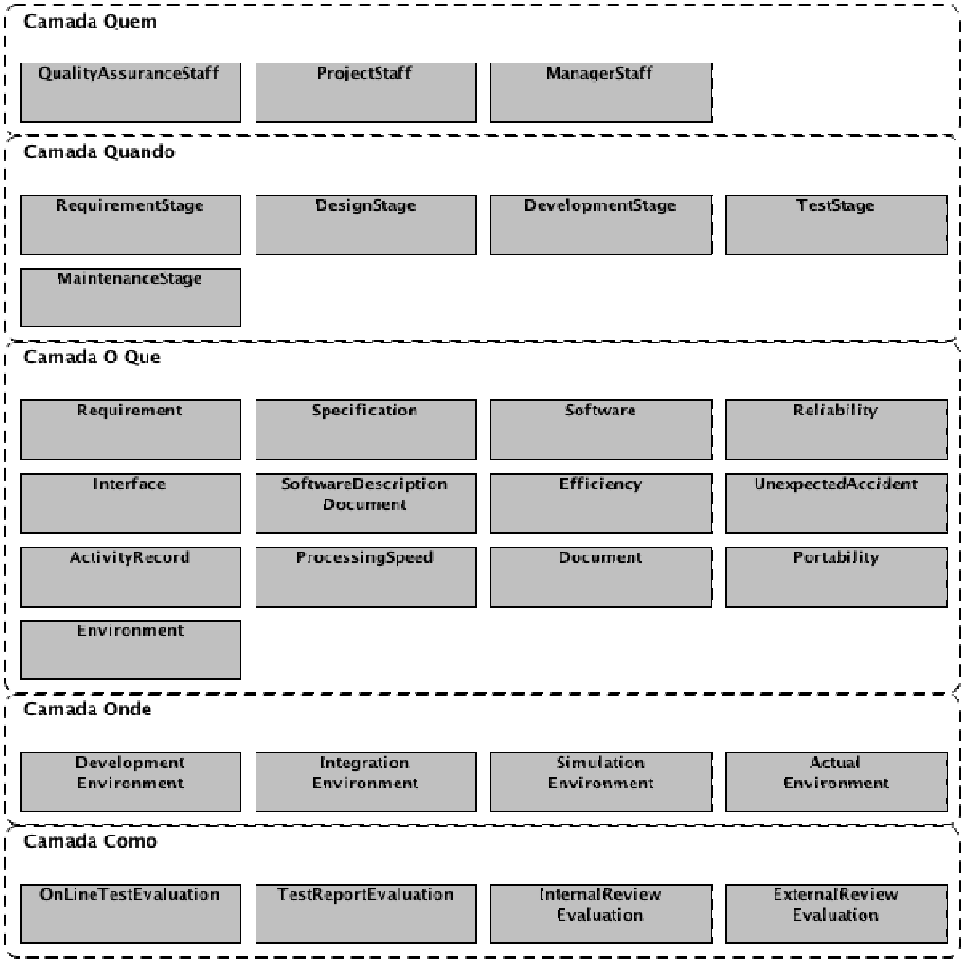

FIGURA 21 - Organização em camadas da ontologia de PPQA.

4.5 WEB SERVICE INTELIGENTE PARA AVALIAÇÕES DE PPQA

De todos os trabalhos analisados, este artigo é a mais recente publicação relacionada com o tema desta dissertação. Neste artigo, é proposto um *web service* inteligente para avaliações de PPQA, através de um serviço de avaliação de processos e produtos de trabalho. Também é disponibilizado um serviço de comunicação, que externa aos interessados os resultados obtidos nas avaliações realizadas. Uma das principais características da publicação é o uso de um agente de raciocínio baseado em ontologia. O objetivo da ferramenta implementada é dar suporte para as gerências no que se refere ao monitoramento da aderência aos processos definidos. De forma complementar, serve como evidência de implementação de PPQA em avaliações CMMI. A Figura 22 mostra a estrutura básica da proposta do artigo (WANG; LEE, 2008).

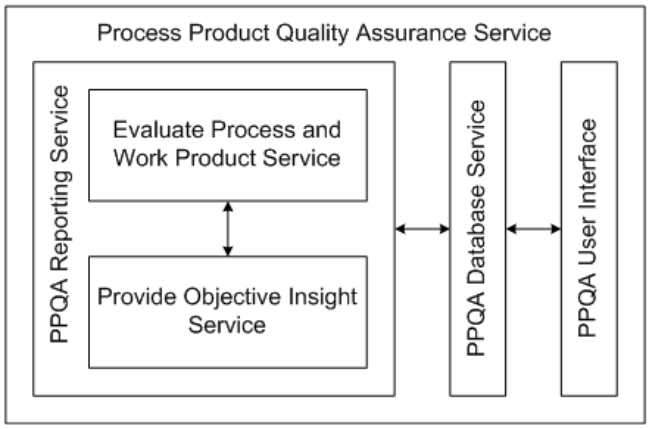

FIGURA 22 - Visão geral da estrutura do *web service* de PPQA.

A Figura 22 apresenta a estrutura do *web service* proposto, onde podem ser observados os componentes do serviço de reporte, sendo este composto por: um serviço de avaliação de processo e produto de trabalho; e um serviço de percepção objetiva sobre a aderência ao processo. Complementarmente, o serviço de reporte interage com o serviço de armazenamento, que, por sua vez, é consumido pelas estruturas de interface de usuário (WANG; LEE, 2008).

4.6 COMPARATIVO ENTRE TRABALHOS RELACIONADOS

Os trabalhos apresentados neste Capítulo foram comparados sob duas perspectivas. A primeira diz respeito ao uso de ontologias para a resolução de problemas de engenharia de software, mais especificamente inspeções de garantia da qualidade de software. A segunda diz respeito ao uso de agentes de software para a automação de atividades de inspeções de garantia da qualidade com base em modelos definidos a partir de ontologias.

Para comparação das ontologias, utilizou-se como critério base a capacidade de responder ao um conjunto de consultas relacionadas com garantia da qualidade, onde se buscou em cada modelo conceitos que, em sua essência, representam indivíduos que satisfazem às consultas. Por exemplo, se uma ontologia não possui algum conceito que remeta a projetos de software, considera-se que esta não é capaz de responder sobre inspeções realizadas em projetos de software.

Para comparação dos agentes, utilizou-se como critério um conjunto de funcionalidades consideradas fundamentais para um sistema de apoio à PPQA, pois estas caracterizam as capacidades de manipulação de ontologias e automação de atividades de inspeção de garantia da qualidade. A avaliação se deu com base nas especificações e

implementações detalhadas em cada trabalho relacionado. Por exemplo, se um trabalho não apresenta como característica ou resultado a capacidade de automatizar o escopo de uma inspeção, considerou-se que este não implementa tal funcionalidade.

A Tabela 4 apresenta um quadro comparativo dos trabalhos apresentados neste Capítulo, segundo as duas perspectivas já mencionadas. Cabe observar que a avaliação realizada é empírica, pois as publicações apresentam apenas abstrações de seus modelos, o que inviabiliza uma experimentação prática. Quadros com valor (Sim) indicam que o critério foi atendido. Quadros com valor (Não) indicam que o critério não foi atendido. Quadros com valor (---) indicam que não foi possível avaliar.

TABELA 4 - Comparativo entre trabalhos relacionados.

	Critérios	**T1**	**T2**	**T3**	**T4**	**T5**
Ontologias	Responde consultas sobre tarefas, papéis ou produtos de trabalho pertencentes a um processo.	Sim	Sim	Sim	Sim	---
	Responde consultas sobre itens de revisão aplicáveis a um produto de trabalho.	Não	Não	Não	Não	---
	Responde consultas sobre inspeções realizadas em projeto de desenvolvimento.	Não	Não	Não	Não	---
	Responde consultas sobre escopo de uma inspeção de garantia da qualidade.	Sim	Não	Sim	Sim	---
	Responde consultas sobre resultado de uma inspeção de garantia da qualidade.	Não	Não	Não	Não	---
	Responde consultas sobre papéis responsáveis por cada não conformidade.	Não	Não	Não	Não	---
Agentes	Capacidade de prover a manutenção de um cadastro de processos de software.	Não	Não	Não	Sim	Não
	Capacidade de prover a manutenção de um cadastro de projetos de software.	Não	Não	Não	Não	Sim
	Capacidade de prover a manutenção de um cadastro de características de qualidade.	Não	Não	Não	Sim	Sim
	Capacidade de prover a manutenção de um cadastro de inspeções de garantia da qualidade.	Não	Não	Não	Não	Sim
	Capacidade de definir automaticamente o escopo de inspeções de garantia da qualidade.	Não	Não	Não	Não	Não
	Capacidade de enquadrar automaticamente os resultados obtidos na verificação de artefatos.	Não	Não	Não	Não	Não
	Capacidade de designar automaticamente os papéis responsáveis pelas não conformidades.	Não	Não	Não	Não	Não
	Capacidade de calcular automaticamente o indicador de aderência ao processo.	Não	Não	Não	Sim	Sim

LEGENDA:
T1 - Ontologia de Processo de Software (FALBO, 1998)
T2 - Ontologia de Modelos de Processo de Software (LIAO; QU; LEUNG, 2005)
T3 - Representação do Conhecimento em Processo de Software (HE et al., 2006)
T4 - Agente Inteligente Baseado em uma Ontologia de PPQA (WANG; LEE, 2007)
T5 - Web Service Inteligente para Avaliações de PPQA (WANG; LEE, 2008)

4.7 FECHAMENTO DO CAPÍTULO

O primeiro trabalho apresentado, apesar de ser o mais antigo dos analisados, tem seu valor na essência da proposta, ou seja, é sabido há algum tempo que a engenharia de software demanda de ferramentas inteligentes capazes de dar suporte aos desenvolvedores. Tal suporte, deve realmente aumentar a produtividade e qualidade do desenvolvimento de software e, para tal, não basta que estas sejam meramente ferramentas de desenho ou codificação. Faz-se necessário ter ferramentas capazes de inferir e identificar comportamento e padrões, deixando para os desenvolvedores a criatividade inerente ao desenvolvimento de software.

O segundo trabalho, ao estabelecer uma ontologia para representação de modelos de referência de qualidade de software, demonstra que as diferentes abordagens de cada modelo convergem para alguns pontos comuns. Tal característica é um fator dificultador para organizações que desejam implantar modelos de qualidade de software, pois nem sempre se tem a ciência dos pontos de intersecção, levando a implementações redundantes e burocráticas. Já o terceiro trabalho, refina a ideia de se ter um modelo para representação de conhecimento em processos de software. Pode-se dizer neste caso que processos de software também têm intersecções e que é de grande serventia para as organizações conseguir identificar redundâncias em seus processos para garantir a melhoria contínua.

O quarto trabalho destaca-se pela definição das camadas de uma inspeção de garantia da qualidade, *Quem*, *Quando*, *O que* e *Onde*, e a implementação de um agente o qual comprova a possibilidade de uso de máquinas de inferência para resolver problemas de engenharia de software. O quinto trabalho é o que mais se aproxima da proposta desta dissertação, pois traz a implementação de um *web service* de apoio às inspeções de garantia da qualidade. Como diferença entre as abordagens, caracteriza-se o foco dado ao tema, o artigo tem por objetivo principal ser uma ferramenta de apoio ás gerências e avaliações CMMI, enquanto que, esta dissertação tem por objetivo ser uma ferramenta de apoio aos responsáveis pelas inspeções de projetos de desenvolvimento.

5 SISTEMA DE APOIO ÀS INSPEÇÕES DE GARANTIA DA QUALIDADE

Inspeções de garantia da qualidade são atividades subjetivas altamente dependentes de interação humana. Entretanto, entende-se como viável a construção de um sistema computacional capaz de automatizar algumas das atividades executadas ao longo destas inspeções. Para a construção deste sistema, faz-se necessário o uso de modelos de representação de conhecimento e programas com a capacidade de manipular as instâncias destes modelos.

Este Capítulo apresenta na Seção 5.1 uma visão geral do sistema proposto. Na Seção 5.2 é apresentada uma ontologia para o domínio de inspeções de garantia da qualidade e o detalhamento de sua construção. Na Seção 5.3 é a especificada a arquitetura dos agentes de suporte às inspeções de garantia da qualidade. Na Seção 5.4 é apresentado o projeto do protótipo de interface para uso do sistema. A Seção 5.5 faz uma análise comparativa com trabalhos relacionados. A Seção 5.6 traz o fechamento do capítulo.

5.1 VISÃO GERAL DO SISTEMA

Inspeções de garantia da qualidade são atividades fundamentalmente humanas, pois estas são intrinsecamente subjetivas e seus resultados são fruto de análises baseadas em percepções e bom senso. Não é raro se obter resultados distintos em inspeções realizadas por Analistas de Qualidade diferentes, mesmo que estes tenham o mesmo escopo de inspeção, itens de revisão, e verifiquem a mesma amostra, artefatos do projeto. Isto ocorre porque uma inspeção depende da interpretação da realidade do projeto alvo e, esta realidade, pode ser vista de forma diferente por cada pessoa envolvida.

O sistema proposto neste trabalho não tem a pretensão de eliminar o envolvimento humano nas atividades de garantia da qualidade, tão menos de eliminar a subjetividade da inspeção propriamente dita. Este trabalho estabelece, na forma de um sistema computacional, uma ferramenta de apoio às inspeções de garantia da qualidade capaz automatizar atividades bem específicas, as quais garantem a maximização da cobertura das inspeções de garantia da qualidade, sem impactar no esforço despendido na execução desta. A Figura 23 apresenta um diagrama de contexto de casos de uso que explicita as funcionalidades do sistema, caracterizando quais dependem de interação humana e quais são automatizadas.

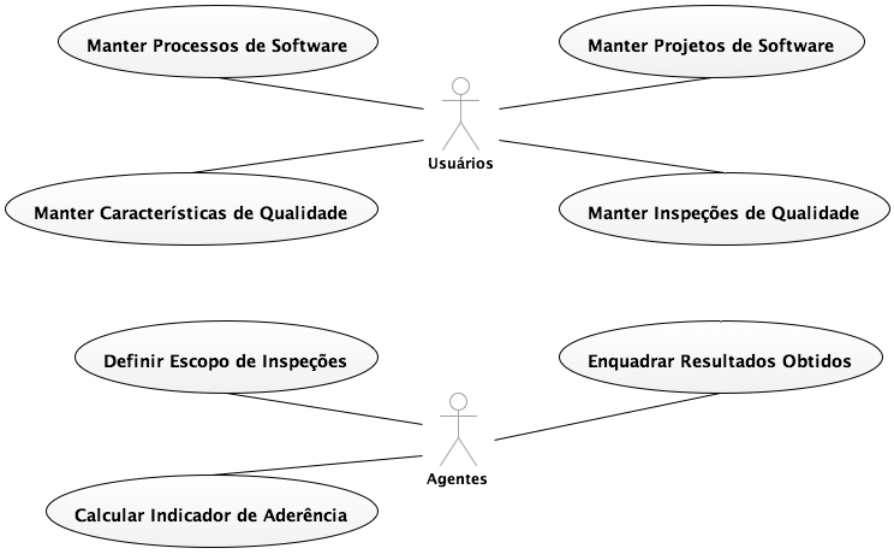

FIGURA 23 - Casos de uso para o sistema de apoio às inspeções.

Inicialmente, pode ser observado na Figura 23 um conjunto de casos de uso que caracterizam as ações dependentes de interação humana. O caso de uso *Manter Processo de Software* estabelece a manutenção de processos de software, que são o critério base para realização de inspeções. O caso de uso *Manter Projetos de Software* define a manutenção dos projetos alvo das inspeções. O caso de uso *Manter Características de Qualidade* estabelece a manutenção dos itens de revisão aplicáveis em um projeto durante as inspeções. O caso de uso *Manter Inspeções de Qualidade* define a manutenção das inspeções propriamente ditas.

A Figura 23 também apresenta os casos de uso que não dependem de interação humana, automatizando ações de uma inspeção de garantia da qualidade. O caso de uso *Definir Escopo de Inspeções* estabelece as regras para definição do conjunto de itens de revisão aplicáveis em uma inspeção. O caso de uso *Enquadrar Resultados Obtidos* define as regras para enquadrar o resultado de avaliação de itens de inspeção como: não aplicável, não atendido ou atendido. Por fim, o caso de uso *Calcular Indicador de Aderência* estabelece a fórmula de cálculo do indicador de acordo com as definições apresentadas na Seção 3.3.3.

Para viabilizar tecnicamente o sistema apresentado, faz-se necessário lançar mão de ferramentas específicas, sendo estas: ontologias e agentes. As ontologias são usadas neste trabalho para formalizar o domínio de conhecimento relacionado ao problema. A escolha de modelos baseados em conhecimento ao invés de modelos baseados em dados se justifica nas tendências da web semântica, onde estruturas formais, explícitas e compartilhadas de representação garantem a manipulação destas representações por sistemas inteligentes. Os agentes são utilizados para implementar as regras de comportamento do sistema, garantindo o encapsulamento do modelo de domínio de conhecimento. Isto se justifica através das características apresentadas por sistemas orientados a agentes, tais como: autonomia e pró-

atividade, as quais viabilizam trabalhos futuros orientados a cognição e aprendizado aplicados à engenharia de software.

Em termos práticos, o uso das tecnologias citadas depende de algumas ferramentas específicas, organizadas em uma pilha conforme a Figura 24. Na camada inferior, tem-se OWL, utilizada para especificar a ontologia, e SPARQL, utilizada para consultar indivíduos na ontologia. Na camada seguinte, observa-se o uso do *framework* Jena, a qual provê um conjunto de classes e métodos para manipulação de modelos OWL e execução de consultas SPARQL. Na próxima camada, aparece a plataforma JADE, a qual provê recursos para implementação e execução de agentes. Por fim, a camada superior traz JSF e *Facelets*, para implementação do protótipo de aplicação web.

FIGURA 24 - Pilha de tecnologias do sistema de apoio às inspeções.

5.2 ONTOLOGIA DE INSPEÇÃO DE PROCESSO

A Ontologia de Inspeção de Processo (OIP) é uma estrutura de representação de conhecimento que cobre o domínio de garantia da qualidade sob uma perspectiva prática. O modelo está organizado em três subdomínios: *processo de software*, que garante a ciência sobre o processo a ser executado; *projeto de software*, que caracteriza o objeto de inspeção propriamente dito; e *inspeção de garantia da qualidade*, que caracteriza a verificação de um projeto contra seu processo de desenvolvimento.

A OIP foi elaborada segundo o método *Ontology Development 101* (NOY e MCGUINNESS, 2001), privilegiando o mapeamento dos conceitos mínimos e necessários para o domínio de conhecimento, não sendo considerados conceitos correlacionados ou complementares. Para o pleno atendimento dos objetivos deste trabalho, faz-se necessário que as consultas na OIP respondam as seguintes questões:

- Quais são as tarefas, papéis ou produtos de trabalho pertencentes a um processo?
- Quais são os itens de revisão aplicáveis a um produto de trabalho?
- Quais são as inspeções realizadas em projeto de desenvolvimento?

- Qual é o escopo de uma inspeção de garantia da qualidade?
- Qual é o resultado de uma inspeção de garantia da qualidade?
- Quem são os papéis responsáveis por cada não conformidade?

É importante observar que ao longo da construção da OIP se considerou a reutilização das ontologias definidas nos trabalhos relacionados apresentados no Capítulo 4. Entretanto, a análise destes modelos mostrou que tal reuso não se justificaria em função de dois aspectos: a) os modelos não mapeiam os principais conceitos do domínio de inspeções de garantia da qualidade de software; b) a extensão dos modelos existentes aumentaria a complexidade de manipulação da ontologia sem agregar valor aos resultados do trabalho.

5.2.1 Definição dos Modelos da OIP

A Figura 25 apresenta um grafo da hierarquia de conceitos da OIP. No primeiro nível, de acordo com os padrões OWL, tem-se uma classe que representa todas as coisas (*Thing*), a qual é superclasse de todas as demais. Nos níveis subsequentes são apresentadas as classes do subdomínio de processo de software (*Process*, *Phase*, *Discipline*, *Role*, *WorkProduct* e *Task*), as classes do subdomínio de projeto de software (*Project*) e as classes do subdomínio de inspeções de garantia da qualidade (*RevisionItem*, *Inspection*, *Scope*, *Finding*, *NoApply*, *Positive* e *Negative*).

FIGURA 25 - Grafo de hierarquia de classes da OIP.

5.2.1.1 Subdomínio de Processo de Software

A implementação de práticas de garantia da qualidade necessita de um critério, ou seja, procedimentos e padrões que indiquem para as equipes como se deve executar um projeto de software. A Tabela 5 apresenta a especificação em DL dos conceitos de processos de software necessários para a plena cobertura do domínio de garantia da qualidade.

TABELA 5 - Subdomínio de processo de software.

Especificação em DL
Process ⊆ Thing Process ⊆ =1 hasName String Process ⊆ =1 hasDescription String Process ⊆ =1 hasVersion String
Phase ⊆ Thing Phase ⊆ =1 hasName String Phase ⊆ =1 hasDescription String Phase ⊆ =1 isPhaseOf Process
Discipline ⊆ Thing Discipline ⊆ =1 hasName String Discipline ⊆ =1 hasDescription String
Role ⊆ Thing Role ⊆ =1 hasName String Role ⊆ =1 hasDescription String
WorkProduct ⊆ Thing WorkProduct ⊆ =1 hasName String WorkProduct ⊆ =1 hasDescription String
Task ⊆ Thing Task ⊆ =1 hasName String Task ⊆ =1 hasDescription String Task ⊆ =1 isTaskOfDiscipline Discipline Task ⊆ ≥1 isTaskOfPhase Phase Task ⊆ =1 hasRole Role Task ⊆ ≥1 hasWorkProduct WorkProduct

A primeira classe apresentada na Tabela 5 representa o universo de processos de software (*Process*) que podem ser inseridos na OIP, a qual é uma subclasse da classe que representa todas as coisas (*Thing*) e tem restrições que garantem que todos os indivíduos possuam um nome, uma descrição e uma versão (*hasName*, *hasDescription* e *hasVersion*). Na sequência é especificada a classe que representa a organização de um processo em fases (*Phase*), a qual também é uma subclasse de todas as coisas (*Thing*). A classe possui as restrições que garantem que seus indivíduos tenham um nome, uma descrição (*hasName* e *hasDescription*) e que pertençam a um processo (*isPhaseOf*).

A Tabela 5 também apresenta uma classe que representa o agrupamento de tarefas de um mesmo fim em disciplinas (*Discipline*), que é subclasse de todas as coisas (*Thing*). As

restrições para esta classe garantem que todos os indivíduos possuam um nome e uma descrição (*hasName* e *hasDescription*). Na sequência, tem-se a classe que representa os papéis (*Roles*) responsáveis pela execução das tarefas de um processo. Esta classe é uma subclasse de todas as coisas (*Thing*) e suas restrições garantem que todos os indivíduos possuam um nome e uma descrição (*hasName* e *hasDescription*).

Por fim, a Tabela 5 traz a especificação da classe produto de trabalho (*WorkProduct*) que representa os artefatos de saída de uma determinada tarefa, a qual é subclasse de todas as coisas (*Thing*). A classe possui restrições que garantem que todos os indivíduos possuam um nome e uma descrição (*hasName* e *hasDescription*). A última classe representa as tarefas (*Task*) de um processo de software, sendo esta também, subclasse de todas as coisas (*Thing*). As restrições definidas para a classe garantem que seus indivíduos tenham um nome e uma descrição (*hasName* e *hasDescription*), pertençam a uma disciplina e a no mínimo uma fase (*isTaskOfDiscipline* e *isTaskOfPhase*), possuam um papel responsável (*hasRole*) e no mínimo um produto de trabalho como saída (*hasWorkProduct*).

5.2.1.2 Subdomínio de Projeto de Software

A institucionalização de práticas de garantida da qualidade se dá sempre em função de projetos de desenvolvimento de software. Isto se fundamenta no fato de que são os projetos que executam os processos, logo, os projetos são os alvos das inspeções. A Tabela 6 apresenta a especificação do conceito de projeto de software dentro do contexto de garantia da qualidade aplicada.

TABELA 6 - Subdomínio de projeto de software.

Especificação em DL
Project ⊆ Thing
Project ⊆ =1 hasName String
Project ⊆ =1 hasDescription String
Project ⊆ =1 hasManager String

Com base na abordagem de mínimo e necessário adotada durante a construção da OIP, o subdomínio de projeto de software (*Project*) é o mais simples, pois para atender às necessidades de garantia da qualidade, basta que a ontologia seja capaz de identificar e caracterizar os projetos alvos para realização de inspeções. A classe é uma subclasse de todas as coisas (*Thing*) e suas restrições garante que seus indivíduos tenham um nome, uma descrição e um gerente responsável (*hasName, hasDescription* e *hasManager*).

5.2.1.3 Subdomínio de Inspeções de Garantia da Qualidade

As inspeções de garantia da qualidade caracterizam o ápice das práticas de garantia da qualidade de software. Esta afirmação se baseia no fato de que é através da execução inspeções que o propósito da disciplina é atendido. Em outras palavras, são as inspeções que avaliam projetos contra processos e dão uma percepção clara e objetiva sobre como os produtos de software estão sendo construídos. A Tabela 7 apresenta a especificação para os conceitos de inspeções de garantia da qualidade demandados pela ontologia.

TABELA 7 - Subdomínio de inspeção de garantia da qualidade.

Especificação em DL
RevisionItem ⊆ Thing RevisionItem ⊆ =1 hasName String RevisionItem ⊆ =1 hasDescription String RevisionItem ⊆ ≥1 isRevisionItemOf WorkProduct
Inspection ⊆ Thing Inspection ⊆ =1 hasName String Inspection ⊆ =1 hasDescription String Inspection ⊆ =1 hasAnalyst String Inspection ⊆ =1 hasDate Date Inspection ⊆ =1 hasProject Project Inspection ⊆ =1 hasPhase Phase
Scope ⊆ Thing Scope ⊆ =1 hasTarget String Scope ⊆ =1 hasRevisionItem String Scope ⊆ =1 hasStatus String Scope ⊆ =1 isScopeOf Inspection
Finding ⊆ Thing Finding ⊆ =1 isFindingOf Scope
NotApply ⊆ Finding NotApply ⊆ =1 hasJustification String
Positive ⊆ Finding Positive ⊆ =1 hasEvidence String
Negative ⊆ Finding Negative ⊆ =1 hasIssue String Negative ⊆ =1 hasResponsible String

A Tabela 7 apresenta inicialmente a classe de itens de revisão (*RevisionItem*), a qual representa os critérios de qualidade aplicáveis a um determinado produto de trabalho. A classe é uma subclasse de todas as coisas (*Thing*) e suas restrições garantem que cada indivíduo tenha um nome e uma descrição (*hasName* e *hasDescription*) e pertença a no mínimo um produto de trabalho (*isRevisionItemOf*). A próxima classe representa as inspeções de garantia da qualidade (*Inspection*) propriamente ditas. Esta classe é uma subclasse de todas as coisas (*Thing*) e as restrições garantem que todos os indivíduos possuam um nome, uma descrição,

um analista responsável e uma data de execução (*hasName*, *hasDescription*, *hasAnalyst* e *hasDate*), e também estejam relacionados a um projeto e uma fase (*hasProject* e *hasPhase*). A classe subsequente representa o escopo (*Scope*) de uma inspeção, ou seja, estabelece uma relação dos itens de revisão com produtos de trabalho aplicados em uma inspeção de projetos de desenvolvimento. A classe é uma subclasse de todas as coisas (*Thing*) e suas restrições garantem que cada indivíduo tenha um alvo, um item de revisão e um status (*hasTarget*, *hasRevisionItem* e *hasStatus*) e pertença a uma inspeção (*isScopeOf*).

O último grupo de classes apresentadas pela Tabela 7 está relacionado com a representação dos resultados das inspeções de garantia da qualidade. A primeira classe deste grupo representa os resultados (*Finding*[3]) de uma inspeção de forma generalizadora, sendo esta classe uma subclasse de todas as coisas (*Thing*) e com uma restrição que garante que seus indivíduos pertençam a um escopo (*isFindingOf*). A próxima classe representa os itens não aplicáveis em uma inspeção (*NotApply*), a qual é uma subclasse de resultado (*Finding*) e possui uma restrição que garante que os indivíduos desta classe tenham uma justificativa (*hasJustification*). Na sequencia, tem-se a classe que representa os itens em conformidade (*Positive*), que é subclasse de resultado (*Finding*) e a restrição existente garante que os indivíduos tenham uma evidência (*hasEvidence*). Por fim, tem-se a classe que representa itens em não conformidade (*Negative*), a qual também é subclasse de resultado (*Finding*) e suas restrições garantem que seus indivíduos tenham uma descrição de não conformidade e um conjunto de responsáveis (*hasIssue* e *hasResponsible*).

5.2.2 Implementação dos Modelos da OIP

A OIP foi implementada segundo modelo de especificação OWL acessível através dos recursos disponibilizados pelo *framework* Jena para manipulação de ontologias. A Figura 26 mostra o componente *Ontology*, criado para encapsular as atividades de consulta e manutenção da OIP. No componente, pode ser vista a estrutura *Query*, a qual expõe uma interface para realização de consultas SPARQL. Também pode ser vista a estrutura *Model*, que expõe uma interface para manutenção, tanto do modelo, quanto dos indivíduos da ontologia. Para a persistência da OIP, também se usou recursos do *framework* Jena, o qual possibilita o armazenamento de ontologias em banco de dados. Tal capacidade garante um

[3] Termo usado pelo CMMI para caracterizar cada percepção dos avaliadores em uma inspeção (CHRISSIS; KONRAD; SHRUM, 2007).

melhor desempenho em consultas SPARQL e mais segurança, tanto na manipulação da ontologia, quanto para evitar perda de informações. A Figura 26 apresenta o componente *Database*, como parte do nodo que representa o Sistema Gerenciador de Banco de Dados (SGBD) PostgreSQL, escolhido para o uso neste trabalho.

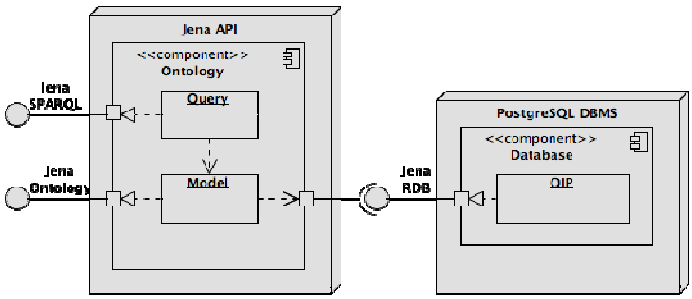

FIGURA 26 - Componente Jena para manipulação da OIP.

A recuperação de indivíduos da OIP é feita através dos recursos do *framework* Jena para execução de consultas SPARQL. A seguir é apresentada uma série de consultas que respondem as questões definidas na Seção 5.2, com o objetivo de demonstrar as capacidades da OIP no que se refere à representação do domínio de conhecimento mapeado. A Figura 27 apresenta uma consulta que responde quais são as tarefas, papéis ou produtos de trabalho pertencentes a um processo. Para tal, faz-se necessário recuperar todos os indivíduos das classes *Phase*, *Task*, *Role* e *WorkProduct*, relacionados a um indivíduo da classe *Process*.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX oip: <http://www.unisinos.br/OntoInspProcess.owl#>
SELECT ?PhaseHasName ?TaskHasName ?RoleHasName ?WorkProductHasName
 WHERE { ?Process rdf:type oip:Process .
         ?Process oip:hasName ?ProcessHasName .
         ?Phase rdf:type oip:Phase .
         ?Phase oip:hasName ?PhaseHasName .
         ?Phase oip:isPhaseOf ?PhaseIsPhaseOf .
         ?Task rdf:type oip:Task .
         ?Task oip:hasName ?TaskHasName .
         ?Task oip:isTaskOfPhase ?TaskIsTaskOfPhase .
         ?Task oip:hasRole ?TaskHasRole .
         ?Task oip:hasWorkProduct ?TaskHasWorkProduct .
         ?Role rdf:type oip:Role .
         ?Role oip:hasName ?RoleHasName .
         ?WorkProduct rdf:type oip:WorkProduct .
         ?WorkProduct oip:hasName ?WorkProductHasName .
         FILTER (?WorkProduct = ?TaskHasWorkProduct) .
         FILTER (?Role = ?TaskHasRole) .
         FILTER (?Phase = ?TaskIsTaskOfPhase) .
         FILTER (?Process = ?PhaseIsPhaseOf) .
         FILTER (?Process = oip:I125927142137995) }
ORDER BY ?ProcessHasName ?PhaseHasName ?TaskHasName
```

PhaseHasName	TaskHasName	RoleHasName	WorkProductHasName
"Construction"	"Apply Change Request"	"Team"	"Change Applied"
"Construction"	"Build System"	"Programmer"	"Build"
"Construction"	"Codify System"	"Programmer"	"Source Code"
"Construction"	"Create Baseline"	"Team"	"Baseline Applied"
...			
"Delivery"	"Apply Change Request"	"Team"	"Change Applied"
"Delivery"	"Build System"	"Programmer"	"Build"
"Delivery"	"Codify System"	"Programmer"	"Source Code"
"Delivery"	"Create Baseline"	"Team"	"Baseline Applied"
...			
"Planning"	"Analyze Requirements"	"Analyst"	"Requirements Model"
"Planning"	"Apply Change Request"	"Team"	"Change Applied"
"Planning"	"Create Baseline"	"Team"	"Baseline Applied"
"Planning"	"Elaborate Change Request"	"Manager"	"Change Request"
...			
"Specification"	"Apply Change Request"	"Team"	"Change Applied"
"Specification"	"Approve Change Request"	"Manager"	"Change Request Approved"
"Specification"	"Construct Wireframes"	"Analyst"	"Wireframe"
"Specification"	"Create Baseline"	"Team"	"Baseline Applied"
...			

FIGURA 27 - Consulta que retorna fases, tarefas, papéis e produtos de trabalho.

A Figura 28 apresenta a consulta que responde quais são os itens de revisão aplicáveis a um produto de trabalho. Neste caso, recuperam-se todos os indivíduos das classes *WorkProduct* e *RevisionItem*.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX oip: <http://www.unisinos.br/OntoInspProcess.owl#>
SELECT ?WorkProductHasName ?RevisionItemHasDescription
 WHERE { ?WorkProduct rdf:type oip:WorkProduct .
         ?WorkProduct oip:hasName ?WorkProductHasName .
         ?RevisionItem rdf:type oip:RevisionItem .
         ?RevisionItem oip:hasDescription ?RevisionItemHasDescription .
         ?RevisionItem oip:isRevisionItemOf ?RevisionItemIsRevisionItemOf .
         FILTER (?WorkProduct = ?RevisionItemIsRevisionItemOf) }
ORDER BY ?WorkProductHasName ?RevisionItem
```

WorkProductHasName	RevisionItemHasDescription
"Action Plan"	"The artifact was continuously updated."
"Action Plan"	"The correct resource was used."
"Action Plan"	"The issue was properly open and assigned."
"Analysis Meeting Minutes"	"The artifact was continuously updated."
...	
"Change Request"	"The correct resource was used."
"Change Request Approved"	"The agreement was recorded."
"Change Request Approved"	"The artifact was continuously updated."
"Change Request Approved"	"The correct resource was used."
...	
"Lesson Learned"	"The artifact was continuously updated."
"Lesson Learned"	"The correct resource was used."
"Milestone Meeting Minutes"	"The discussions and decisions were recorded."
"Milestone Meeting Minutes"	"The artifact was continuously updated."
...	
"Requirements Model"	"The artifact was continuously updated."
"Requirements Model"	"The traceability was kept in de model."
"Requirements Validation Report"	"The agreement was recorded."
"Requirements Validation Report"	"The artifact was continuously updated."
...	

FIGURA 28 - Consulta que retorna os itens de revisão dos produtos de trabalho.

A Figura 29 responde quais são as inspeções realizadas em projeto de desenvolvimento. Para satisfazer esta consulta é necessário selecionar os indivíduos das classes *Project* e *Inspection*, relacionando-os para poder saber qual inspeção é de qual projeto.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX oip: <http://www.unisinos.br/OntoInspProcess.owl#>
SELECT ?ProjectHasName ?InspectionHasName ?InspectionHasDescription
 WHERE { ?Project rdf:type oip:Project .
         ?Project oip:hasName ?ProjectHasName .
         ?Inspection rdf:type oip:Inspection .
         ?Inspection oip:hasName ?InspectionHasName .
         ?Inspection oip:hasDescription ?InspectionHasDescription .
         ?Inspection oip:hasProject ?InspectionHasProject .
         FILTER (?Project = ?InspectionHasProject) }
ORDER BY ?ProjectHasName ?InspectionHasName
```

ProjectHasName	InspectionHasName	InspectionHasDescription
"Project A"	"Inspection A1"	"Inspection of planning phase."
"Project A"	"Inspection A3"	"Inspection of construction phase."
"Project A"	"Inspection A4"	"Inspection of delivery phase."
"Project B"	"Inspection B1"	"Inspection of planning phase."
...		
"Project C"	"Inspection C4"	"Inspection of delivery phase."
"Project D"	"Inspection D1"	"Inspection of planning phase."
"Project D"	"Inspection D3"	"Inspection of construction phase."
"Project D"	"Inspection D4"	"Inspection of delivery phase."
...		
"Project E"	"Inspection E4"	"Inspection of delivery phase."
"Project F"	"Inspection F1"	"Inspection of planning phase."
"Project F"	"Inspection F3"	"Inspection of construction phase."
"Project F"	"Inspection F4"	"Inspection of delivery phase."
...		
"Project G"	"Inspection G4"	"Inspection of delivery phase."
"Project H"	"Inspection H1"	"Inspection of planning phase."
"Project H"	"Inspection H3"	"Inspection of construction phase."
"Project H"	"Inspection H4"	"Inspection of delivery phase."
...		

FIGURA 29 - Consulta que retorna as inspeções de projetos.

A Figura 30 responde qual é o escopo de uma inspeção de garantia da qualidade, dado um momento do projeto. Para tal, faz-se necessário selecionar os indivíduos da classe *Scope*, que estão relacionados a um determinado indivíduo da classe *Inspection*.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX oip: <http://www.unisinos.br/OntoInspProcess.owl#>
SELECT ?InspectionHasName ?ScopeHasTarget ?ScopeHasRevisionItem
 WHERE { ?Inspection rdf:type oip:Inspection .
         ?Inspection oip:hasName ?InspectionHasName .
         ?Scope rdf:type oip:Scope .
         ?Scope oip:hasTarget ?ScopeHasTarget .
         ?Scope oip:hasRevisionItem ?ScopeHasRevisionItem .
         ?Scope oip:hasStatus ?ScopeHasStatus .
         ?Scope oip:isScopeOf ?ScopeIsScopeOf .
         FILTER (?Inspection = ?ScopeIsScopeOf) .
         FILTER (?ScopeIsScopeOf = oip:i126567439906025) }
ORDER BY ?InspectionHasName ?ScopeHasTarget ?ScopeHasRevisionItem";
```

InspectionHasName	ScopeHasTarget	ScopeHasRevisionItem
"Inspection J4"	"Action Plan"	"The artifact was continuously updated."
"Inspection J4"	"Action Plan"	"The correct resource was used."
"Inspection J4"	"Baseline Applied"	"The artifact was continuously updated."
"Inspection J4"	"Baseline Applied"	"The artifacts used were designed."
...		
"Inspection J4"	"Configuration Item"	"The artifact was continuously updated."
"Inspection J4"	"Configuration Item"	"The artifacts used were designed."
"Inspection J4"	"Configuration Item"	"The correct resource was used."
"Inspection J4"	"Configuration Plan"	"The artifact was continuously updated."
...		
"Inspection J4"	"Milestone Meeting Minutes"	"The artifact was continuously updated."
"Inspection J4"	"Milestone Meeting Minutes"	"The correct resource was used."
"Inspection J4"	"Product Accepted"	"The agreement was recorded."
"Inspection J4"	"Product Accepted"	"The artifact was continuously updated."
...		
"Inspection J4"	"System Documentation"	"The artifact was continuously updated."
"Inspection J4"	"System Documentation"	"The correct resource was used."
"Inspection J4"	"Test Errors Report"	"The artifact was continuously updated."
"Inspection J4"	"Test Errors Report"	"The correct resource was used."
...		

FIGURA 30 - Consulta que retorna o escopo de inspeções.

A Figura 31 apresenta uma consulta que responde qual é o resultado de uma inspeção de garantia da qualidade. Pode ser observada na consulta a recuperação de indivíduos das classes *Inspection*, *Scope* e *Positive*. Neste caso em específico são recuperados indivíduos que representam itens em conformidade. Para recuperar indivíduos em não conformidade ou não aplicáveis basta aplicar a mesma consulta nas respectivas classes *Negative* ou *NotApply*.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX oip: <http://www.unisinos.br/OntoInspProcess.owl#>
SELECT ?ScopeHasTarget ?ScopeHasRevisionItem
 WHERE { ?Inspection rdf:type oip:Inspection .
         ?Inspection oip:hasName ?InspectionHasName .
         ?Scope rdf:type oip:Scope .
         ?Scope oip:hasTarget ?ScopeHasTarget .
         ?Scope oip:hasRevisionItem ?ScopeHasRevisionItem .
         ?Scope oip:isScopeOf ?ScopeIsScopeOf .
         ?Positive rdf:type oip:Positive .
         ?Positive oip:isFindingOf ?PositiveIsFindingOf .
         FILTER (?Inspection = ?ScopeIsScopeOf) .
         FILTER (?Scope = ?PositiveIsFindingOf) .
         FILTER (?ScopeIsScopeOf = oip:" + id + ") }
ORDER BY ?InspectionHasName ?ScopeHasTarget ?ScopeHasRevisionIte
```

ScopeHasTarget	ScopeHasRevisionItem
"Action Plan"	"The artifact was continuously updated."
"Action Plan"	"The correct resource was used."
"Action Plan"	"The issue was properly open and assigned."
"Baseline Applied"	"The artifact was continuously updated."
...	

FIGURA 31 - Consulta que retorna o resultado de uma inspeção.

A Figura 32 responde quem são os papéis responsáveis por cada não conformidade, buscando os indivíduos das classes *Inspection*, *Scope* e *Negative*. De forma complementar é

apresentado o atributo *hasResponsible*, que contém os papéis responsáveis pela não conformidade.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX oip: <http://www.unisinos.br/OntoInspProcess.owl#>
SELECT ?InspectionHasName ?ScopeHasTarget ?ScopeHasRevisionItem ?NegativeHasResponsible
 WHERE { ?Inspection rdf:type oip:Inspection .
         ?Inspection oip:hasName ?InspectionHasName .
         ?Scope rdf:type oip:Scope .
         ?Scope oip:hasTarget ?ScopeHasTarget .
         ?Scope oip:hasRevisionItem ?ScopeHasRevisionItem .
         ?Scope oip:isScopeOf ?ScopeIsScopeOf .
         ?Negative rdf:type oip:Negative .
         ?Negative oip:hasResponsible ?NegativeHasResponsible .
         ?Negative oip:isFindingOf ?NegativeIsFindingOf .
         FILTER (?Inspection = ?ScopeIsScopeOf) .
         FILTER (?Scope = ?NegativeIsFindingOf) .
         FILTER (?ScopeIsScopeOf = oip:" + id + ") }
ORDER BY ?InspectionHasName ?ScopeHasTarget ?ScopeHasRevisionItem

-------------------------------------------------------------------------------------
|InspectionHasName|ScopeHasTarget  |ScopeHasRevisionItem      | NegativeHasResponsible|
=====================================================================================
|"Inspection J4"  |"Action Plan"   |"The artifact was         | "[Manager]"           |
|                 |                | continuously updated."   |                       |
|"Inspection J4"  |"Lesson Learned"|"The discussions and      | "[Manager]"           |
|                 |                | decisions were recorded."|                       |
|"Inspection J4"  |"Lesson Learned"|"The artifact was         | "[Manager]"           |
|                 |                | continuously updated."   |                       |
|"Inspection J4"  |"Lesson Learned"|"The correct resource     | "[Manager]"           |
|                 |                | was used."               |                       |
-------------------------------------------------------------------------------------
```

FIGURA 32 - Consulta que retorna o responsável por não conformidade.

5.3 AGENTES DE INSPEÇÃO DE PROCESSO

Os Agentes de Inspeção de Processo (AIP) são agentes de software responsáveis pela manipulação da OIP, capazes de perceber ações externas, analisá-las segundo regras de comportamento e atuar sobre os indivíduos da ontologia. O ambiente no qual o AIP está inserido caracteriza-se por ser *observável*, pois os estados são conhecidos; *determinístico*, pois o próximo estado é definido em função do anterior; *episódico*, pois há etapas bem estabelecidas; *estático*, pois não há mudança enquanto o AIP atua; *discreto*, pois existem um número definido de estados, ações e percepções; e *multiagente*, pois mais de um agente atua sobre o ambiente.

O AIP é um sistema multiagente, composto por três agentes com responsabilidades distintas, estruturalmente definidos como *reativos simples*. Os agentes realizam ações baseadas apenas na sua percepção atual do ambiente e foram construídos através de um processo de desenvolvimento iterativo e incremental (LARMAN, 2004). Não se utilizou métodos específicos de engenharia de software orientada a agentes (GIORGINI et al., 2004) porque o AIP não tem características cognitivas, logo, um método de engenharia de software convencional mostrou-se mais produtivo para o seu desenvolvimento.

Apesar de sua simplicidade, entende-se que o AIP atende às três propriedades de sistemas multiagentes, visto que o AIP é: *autônomo* quando realiza a criação de indivíduos

por conta própria; *pró-ativo* quando se identifica e estabelece responsáveis por uma não conformidade; *sociável* quando estabelecem interações através da recuperação e criação de indivíduos nos subdomínios de processo de software, projetos de software e inspeções de garantia da qualidade.

A Figura 33 apresenta a arquitetura do AIP, a qual está em conformidade com arquitetura base definida pela plataforma JADE. Pode ser observada no componente AIP a definição dos seguintes agentes: mantenedor de processo (*ProcessMaintainer*), responsável por manipular os indivíduos pertencentes ao domínio de processo de software; mantenedor de projeto (*ProjectMaintainer*), responsável por manipular os indivíduos pertencentes ao domínio de projeto de software; e mantenedor de inspeção (*InspectionMaintainer*), responsável por manipular os indivíduos do domínio de inspeções de garantia da qualidade.

Ainda pode ser observada no componente AIP da Figura 33, a definição serviços providos por cada agente. Tais serviços caracterizam os comportamentos dos agentes, sendo definidos os seguintes: execução de consultas SPARQL (*OntologyQuerying*); configuração de propriedades do tipo dado em indivíduos (*DataPropertySetting*); criação de indivíduos (*IndividualCreation*); remoção de indivíduos (*IndividualDeletion*); adição de propriedades do tipo objeto (*ObjectPropertyAddition*); configuração de propriedades do tipo objeto (*ObjectPropertySetting*).

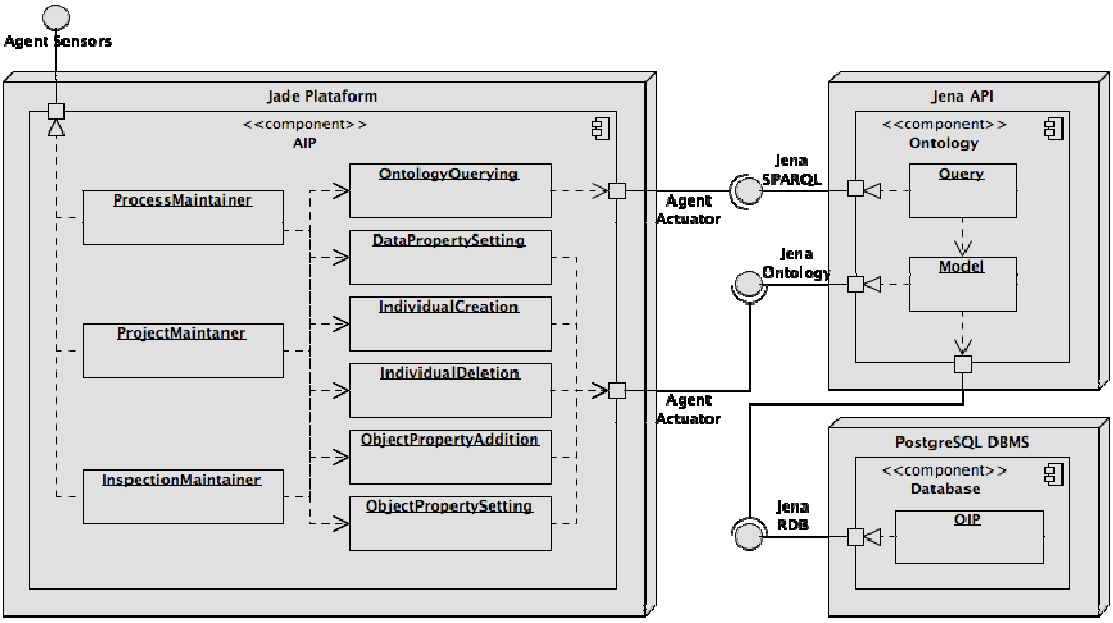

FIGURA 33 - Arquitetura de componentes do AIP.

Complementarmente, pode-se observar na Figura 33 a existência de uma interface que caracteriza os sensores do AIP. Esta interface se conecta ao protótipo web com o objetivo de capturar os eventos demandantes de ações dos agentes. Por fim, tem-se a definição de dois atuadores conectados ao componente de manutenção de ontologias (*Ontology*). O primeiro

atuador realiza consultas SPARQL e o segundo realiza manutenções no universo de indivíduos da OIP.

5.3.1 Especificação Funcional do AIP

Os agentes que constituem o sistema possuem responsabilidades e serviços de acordo com o papel assumido por cada um. A Figura 34 apresenta um diagrama de contexto de casos de uso que explicita de forma resumida o papel, as responsabilidades e os serviços de cada agente. Os papéis são caracterizados pelos atores que representam os agentes no diagrama, as responsabilidades são caracterizadas de acordo com o subdomínio da OIP mantido por cada agente e os serviços são caracterizados pelos casos de uso iniciados pelos agentes.

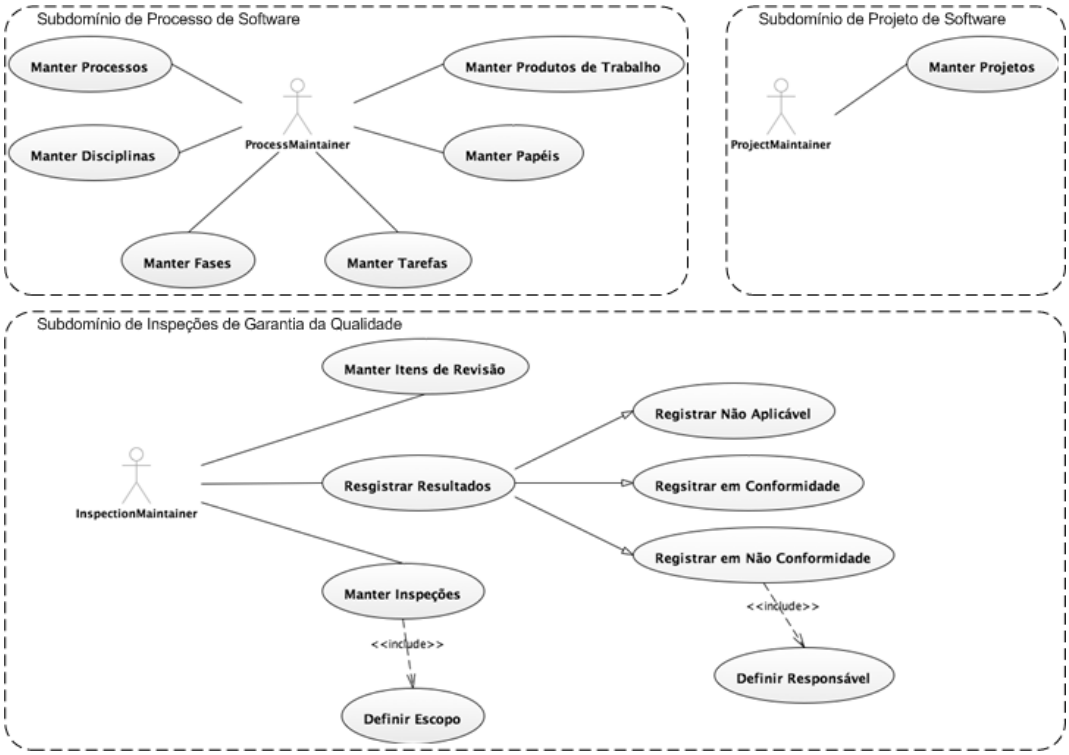

FIGURA 34 - Papéis, responsabilidades e serviços do AIP.

A Figura 34 apresenta o ator *ProcessMaintainer*, o qual é responsável pelo subdomínio de processo de software e provê serviços para manutenção das classes deste subdomínio. A Tabela 8 detalha as regras de comportamento deste agente através da definição de suas percepções e ações.

TABELA 8 - Regras de comportamento para *ProcessMaintainer*.

Caso de Uso	Percepção	Ação
Manter Disciplinas	Solicitada a listagem de disciplinas.	Busca todos os indivíduos da classe *Discipline*.
	Solicitada a inclusão de uma disciplina.	Adiciona um indivíduo na classe *Discipline*. Define valores para as propriedades *hasName* e *hasDescription*.

Caso de Uso	Percepção	Ação
	Solicitada a visualização de uma disciplina.	Identifica o indivíduo e busca os valores de suas propriedades.
	Solicitada a exclusão de uma disciplina.	Identifica o indivíduo e o remove da ontologia.
Manter Fases	Solicitada a listagem de fases.	Busca todos os indivíduos da classe *Phase*.
	Solicitada a inclusão de uma fase.	Adiciona um indivíduo na classe *Phase*. Define valores para as propriedades *hasName* e *hasDescription*. Relaciona a fase com um processo através da propriedade *isPhaseOf*.
	Solicitada a visualização de uma fase.	Identifica o indivíduo e busca os valores de suas propriedades.
	Solicitada a exclusão de uma fase.	Identifica o indivíduo e o remove da ontologia.
Manter Papéis	Solicitada a listagem de papéis.	Busca todos os indivíduos da classe *Role*.
	Solicitada a inclusão de um papel.	Adiciona um indivíduo na classe papel *Role*. Define valores para as propriedades *hasName* e *hasDescription*.
	Solicitada a visualização de um papel.	Identifica o indivíduo e busca os valores de suas propriedades.
	Solicitada a exclusão de um papel.	Identifica o indivíduo e o remove da ontologia.
Manter Processos	Solicitada a listagem de processos.	Busca todos os indivíduos da classe *Process*.
	Solicitada a inclusão de um processo.	Adiciona um indivíduo na classe *Process*. Define valores para as propriedades *hasName*, *hasDescription* e *hasVersion*.
	Solicitada a visualização de um processo.	Identifica o indivíduo e busca os valores de suas propriedades. Busca todos os indivíduos da classe *Task* vinculados ao processo em questão.
	Solicitada a exclusão de um processo.	Identifica o indivíduo e o remove da ontologia.
Manter Produtos de Trabalho	Solicitada a listagem de produtos de trabalho.	Busca todos os indivíduos da classe *WorkProduct*.
	Solicitada a inclusão de um produto de trabalho.	Adiciona um indivíduo na classe *WorkProduct*. Define valores para as propriedades *hasName* e *hasDescription*.
	Solicitada a visualização de um produto de trabalho.	Identifica o indivíduo e busca os valores de suas propriedades.
	Solicitada a exclusão de um produto de trabalho.	Identifica o indivíduo e o remove da ontologia.
Manter Tarefas	Solicitada a listagem de tarefas.	Busca todos os indivíduos da classe *Task*.
	Solicitada a inclusão de uma tarefa.	Adiciona um indivíduo na classe *Task*. Define valores para as propriedades *hasName*, *hasDescription*. Relaciona a tarefa com disciplina através da propriedade *isTaskOfDiscipline*. Relaciona a tarefa com fase através da propriedade *isTaskOfPhase*. Relaciona a tarefa com um papel através da propriedade *hasRole*. Relaciona a tarefa com um produto de trabalho através da propriedade *hasWorkProduct*.
	Solicitada a visualização de uma tarefa.	Identifica o indivíduo e busca os valores de suas propriedades.
	Solicitada a exclusão de uma tarefa.	Identifica o indivíduo e o remove da ontologia.

A Figura 34 também apresenta o ator *ProjectMaintainer*, o qual é responsável pelo subdomínio de projeto de software e provê serviços para manutenção das classes deste subdomínio. A Tabela 9 detalha as regras de comportamento deste agente através da definição de suas percepções e ações.

TABELA 9 - Regras de comportamento para *ProjectMaintainer*.

Caso de Uso	Percepção	Ação
Manter Projetos	Solicitada a listagem de projetos.	Busca todos os indivíduos da classe *Project*.
	Solicitada a inclusão de um projeto.	Adiciona um indivíduo na classe *Project*. Define valores para as propriedades *hasName*, *hasDescription* e *hasManager*.
	Solicitada a visualização de um projeto.	Identifica o indivíduo e busca os valores de suas propriedades.
	Solicitada a exclusão de um projeto.	Identifica o indivíduo e o remove da ontologia.

A Figura 34 por fim apresenta o ator *InspectionMaintainer*, o qual é responsável pelo subdomínio de inspeções de garantia da qualidade e provê serviços para manutenção das classes deste subdomínio. A Tabela 10 detalha as regras de comportamento deste agente através da definição de suas percepções e ações.

TABELA 10 - Regras de comportamento para *InspectionMaintainer*.

Caso de Uso	Percepção	Ação
Definir Escopo	Incluída uma nova inspeção.	Busca por todos os itens de revisão, associados a produtos de trabalhos, mantidos por tarefas, pertencentes à fase estabelecida. Cria um indivíduo na classe *Scope* para cada registro da busca. Define valores para as propriedades *hasTarget*, *hasRevisionItem*, *hasStatus*. Relaciona o escopo com uma inspeção através da propriedade *isScopeOf*.
Definir Responsável	Incluído um novo item em não conformidade.	Identifica o produto de trabalho relacionado, busca no processo os papéis responsáveis por este. Atribui o resultado à propriedade *hasResponsible*.
Manter Inspeções	Solicitada a listagem de inspeções.	Busca todos os indivíduos da classe *Inspection*.
	Solicitada a inclusão de uma inspeção.	Adiciona um indivíduo na classe *Inspection*. Define valores para as propriedades *hasName*, *hasDescription*, *hasAnalyst*, *hasDate*. Relaciona a inspeção com um projeto através da propriedade *hasProject*. Relaciona a inspeção com uma fase através da propriedade *hasPhase*. Inclui caso de uso *Definir Escopo*.
	Solicitada a visualização de uma inspeção.	Identifica o indivíduo e busca os valores de suas propriedades. Busca todos os indivíduos da classe *Scope* vinculados à inspeção em questão. Calcula a aderência do projeto ao processo.
	Solicitada a exclusão de uma inspeção.	Identifica o indivíduo e o remove da ontologia. Identifica os indivíduos da classe *Scope* relacionados com a inspeção excluída e os remove da ontologia.
Manter Itens de Revisão	Solicitada a listagem de itens de revisão.	Busca todos os indivíduos da classe *RevisionItem*.
	Solicitada a inclusão de um item de revisão.	Adiciona um indivíduo na classe *RevisionItem* Define valores para as propriedades *hasName*, *hasDescription*.

Caso de Uso	Percepção	Ação
		Relaciona o item de revisão com produtos de trabalho através da propriedade *isRevisionItemOf*.
	Solicitada a visualização de um item de revisão.	Identifica o indivíduo e busca os valores de suas propriedades.
	Solicita a exclusão de um item de revisão.	Identifica o indivíduo e o remove da ontologia.
Registrar em Conformidade	Incluído um novo resultado em conformidade.	Extende o caso de uso *Registrar Resultado*. Adiciona um indivíduo na classe *Positive*. Define valores para a propriedade *hasEvidence*. Atualiza o status do escopo relacionado para *Close*.
Registrar em Não Conformidade	Incluído um novo resultado em não conformidade.	Extende o caso de uso *Registrar Resultado*. Adiciona um indivíduo na classe *Negative*. Define valores para a propriedade *hasIssue*. Atualiza o status do escopo relacionado para *Close*.
Registrar Não Aplicável	Incluído um novo resultado.	Extende o caso de uso *Registrar Resultado*. Adiciona um indivíduo na classe *NotApply*. Define valores para a propriedade *hasJustification*. Atualiza o status do escopo relacionado para *Close*.
Registrar Resultado	Solicitada a inclusão de um resultado.	Relaciona o resultado com um escopo através da propriedade *isFindingOf*.

5.3.2 Especificação Técnica do AIP

Os três agentes que compõe o AIP seguem o padrão de arquitetura proposto pela plataforma JADE. A Figura 35 apresenta uma parte do modelo de classes do AIP, onde pode ser visto à esquerda a classe *Agent*, que pertence a JADE e define a estrutura base para a implementação de agentes. Na sequencia, tem-se uma classe abstrata *Maintainer*, a qual é uma estrutura generalizadora que implementa métodos comuns aos três agentes. A classe *ProcessMaintainer* implementa o agente que manipula os indivíduos do subdomínio de processos de software. Basicamente, este agente realiza operações de inclusão, recuperação e exclusão de indivíduos.

A Figura 35 também a apresenta a classe *ProjectMaintainer*, que implementa o agente responsável por manipular os indivíduos do subdomínio de projetos de software. Esta é a menor classe porque, como já observado, a OIP mapeia somente um conceito deste domínio. Por fim, tem-se a classe *InspectionMaintainer*, que implementa o agente que manipula os indivíduos do subdomínio de inspeções de garantia da qualidade. Esta é a classe mais importante, pois suas manipulações não são meramente cadastrais, ou seja, é nesta classe que são implementados os métodos que garantem a autonomia e pró-atividade do AIP, de acordo com as regras de comportamento definidas na Tabela 10.

FIGURA 35 - Diagrama de classes dos agentes do sistema.

No que se refere ao comportamento do AIP, o diagrama de classes da Figura 36 apresenta um conjunto de seis classes que implementam todo o comportamento demandado pelos três agentes definidos. Cabe observar que a classe *OneShotBehaviour* é provida pela plataforma JADE e estabelece um padrão de comportamento para as demais classes. Inicialmente, observam-se as classes *IndividualCreation* e *IndividualDeletion* que possibilitam a criação e remoção de indivíduos na ontologia, respectivamente.

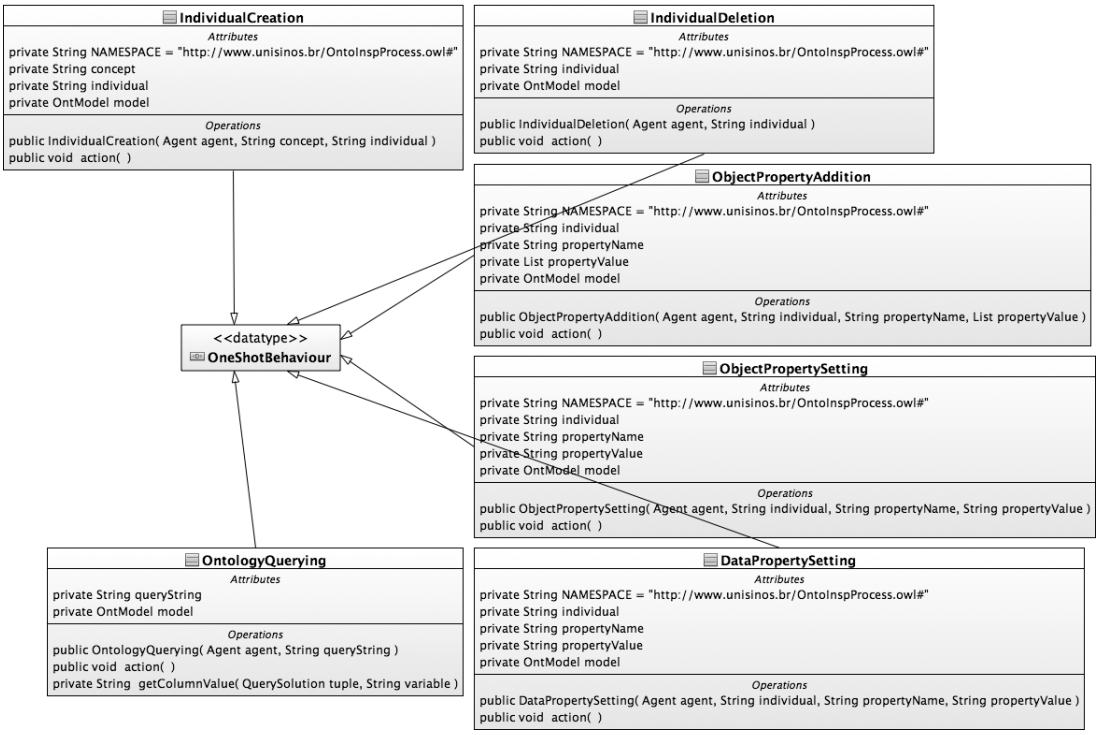

FIGURA 36 - Diagrama de classes de comportamento dos agentes.

De forma complementar, observa-se na Figura 36 que a classe *ObjectPropertyAddition* possibilita a adição de valores de propriedades do tipo objeto em indivíduos já existentes na OIP. A classe *ObjectPropertySetting* possibilita a alteração de valores em propriedade do tipo objeto em indivíduos já existentes na ontologia. A classe *DataPropertySetting* provê a manutenção de valores de propriedades do tipo dado em indivíduos da OIP. Por fim, a classe *OntologyQuerying* provê os recursos necessários para consultas SPARQL na OIP.

Com o objetivo de detalhar a especificação técnica do AIP, são definidos diagramas de sequência para cada um dos comportamentos apresentados no diagrama de classe da Figura 36. A Figura 37 traz o diagrama de sequência para criação de indivíduos na OIP, onde pode ser vista a chamada do método *getOntClass* que retorna a classe da ontologia que receberá o indivíduo. Após, o método *createIndividual* realiza a criação do indivíduo propriamente dito.

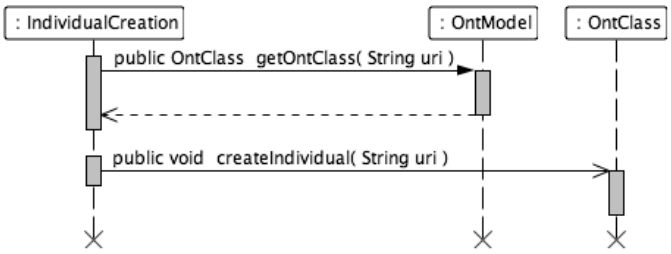

FIGURA 37 - Diagrama de sequência para *IndividualCreation*.

A Figura 38 apresenta o diagrama de sequência da remoção de indivíduos da ontologia. Esta remoção se dá recuperando a referência ao indivíduo através do método *getIndividual* e chamando o método *remove* da classe *Individual* para fazer a exclusão de fato.

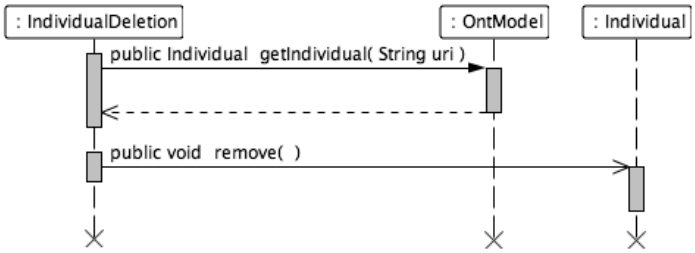

FIGURA 38 - Diagrama de sequência para *IndividualDeletion*.

A Figura 39 apresenta o diagrama de interação para a adição de valores em propriedades do tipo objeto. O objeto da classe *ObjectPropertyAddition* faz as chamadas para os métodos *getIndividual* e *getOntProperty* para recuperar os objetos necessários para a manutenção. Na sequência, é realizada a adição do valor da propriedade em um indivíduo através do método *addProperty,* da classe *Individual*.

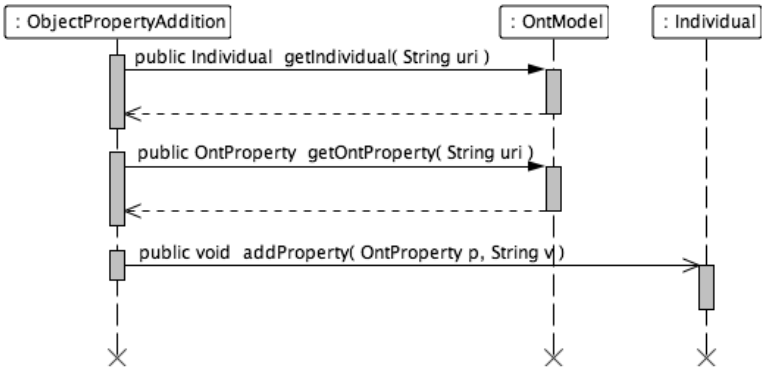

FIGURA 39 - Diagrama de sequência para *ObjectPropertyAddition*.

A Figura 40 apresenta o diagrama de sequência para a alteração em valores de propriedades do tipo objeto já existentes em indivíduos da ontologia. O objeto da classe *ObjectPropertySetting* recupera o indivíduo e propriedades necessárias através da chamada dos métodos *getIndividual* e *getOntProperty*, respectivamente, ambos da classe *OntModel*. Após, é chamado o método *removeProperty* da classe Individual, para apagar algum valor anteriormente definido, e é chamado o método *addProperty* para adicionar o novo valor à propriedade do individuo da OIP.

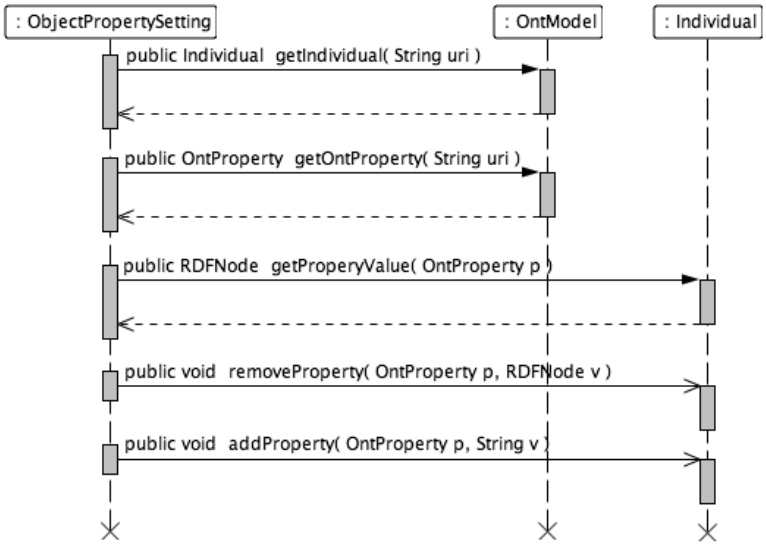

FIGURA 40 - Diagrama de sequência para *ObjectPropertySetting*.

A Figura 41 mostra o diagrama de sequência para alteração em valores de propriedades do tipo data em indivíduos existentes na ontologia. Pode-se observar que a sequência de invocação de métodos é a mesma apresentada na Figura 40, porém, a implementação não é a mesma, pois para OWL propriedades do tipo objeto são diferentes de propriedades do tipo dado. Esta característica é provida pelas estruturas polimórficas do *framework* Jena.

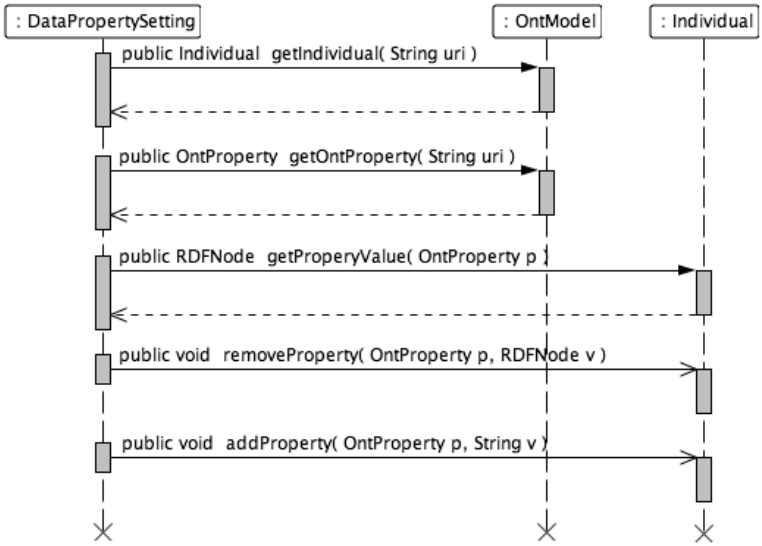

FIGURA 41 - Diagrama de sequência para *DataPropertySetting*.

A Figura 42 apresenta o diagrama de sequência para a realização de consultas SPARQL na OIP. O objeto *OntologyQuerying* usa um conjunto de classes *Factory* providos pelo *framework* Jena para criar as estruturas necessárias para a execução da consulta, sendo invocados os métodos *create* da classe *QueryFactory* e *QueryExectionFactory*. Após, é invocado o método *execSelect* da classe *QueryExecution*, o qual executa de fato a consulta na

ontologia. Este método retorna um *ResultSet*, que pode ser manipulado utilizando o método *nextSolution*.

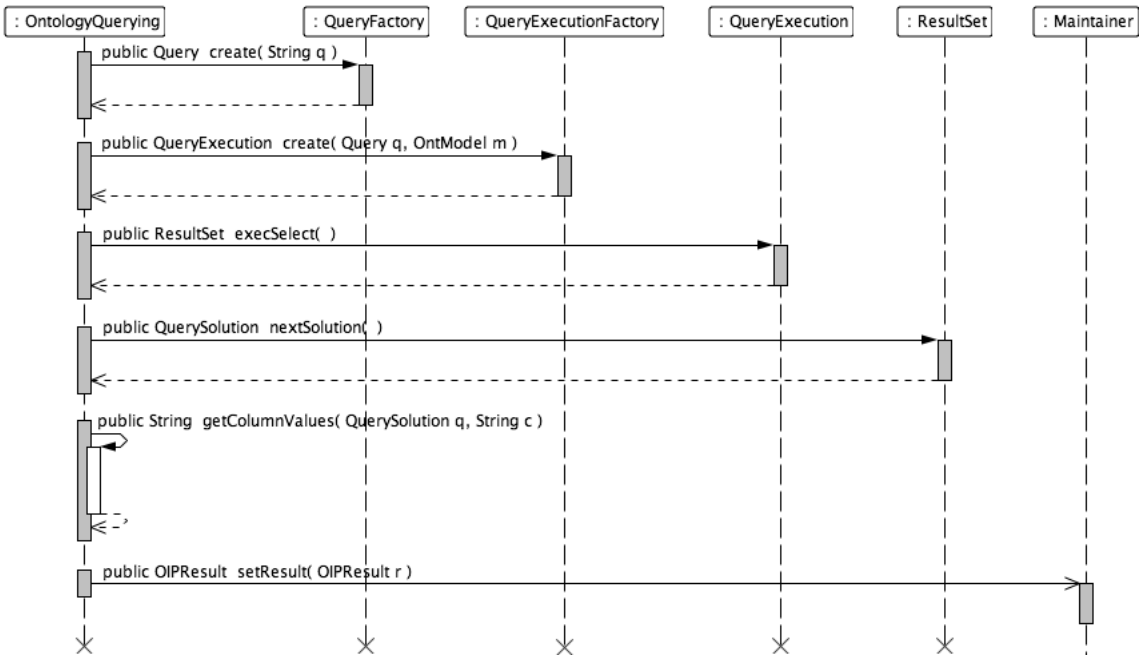

FIGURA 42 - Diagrama de sequência para *OntologyQuerying*.

Ainda pode ser visto na Figura 42, o método *getColumnValues* da classe *OntologyQuerying*. Este método trata e formata cada tupla retornada pelo método *nextSolution* e possibilita a recuperação de um valor específico dentro do *RecordSet*. Ao contrário dos demais comportamentos dos agentes, o *OntologyQuerying* necessita externar seu resultado. Para tal, é feita a chamada do método *setResutl* da classe *Maintainer*, assim o resultado da consulta pode ser manipulado externamente.

5.4 PROTÓTIPO DE INTERFACE DO SISTEMA

Considerando que a OIP e o AIP estão especificados e implementados, faz-se necessário especificar uma camada de aplicação capaz de interagir com as interfaces dos componentes já apresentados. Como alternativa, foi construído um protótipo de aplicação web para apoio às inspeções de garantia da qualidade. Este protótipo não tem o objetivo de traçar padrões para um produto final, mas sim prover uma interface para verificação e validação, tanto do AIP, quanto da OIP. O protótipo definido é caracterizado por:

- rodar em um navegador web;
- ter interface simples e funcional;
- validar campos obrigatórios na entrada de dados;

- não implementar padrões de usabilidade ou ergonomia;
- não implementar padrões de autenticação ou segurança.

O protótipo foi implementado com tecnologia Java EE, mais especificamente, através de JSF e *Facelets*. A Figura 43 apresenta a arquitetura final do sistema de apoio às inspeções de garantia da qualidade, a qual está incrementada com a camada do protótipo de aplicação web, representada pelo nodo JSF. Pode-se observar a existência de três componentes distintos, sendo estes: *Model*, *View* e *Controller*. A *Model* implementa todas as classes de domínio de aplicação, a *View* implementa todas as páginas web necessária para criar interfaces com usuário, e a *Controller*, faz o roteamento entre as requisições da *View* e as respostas da *Model*.

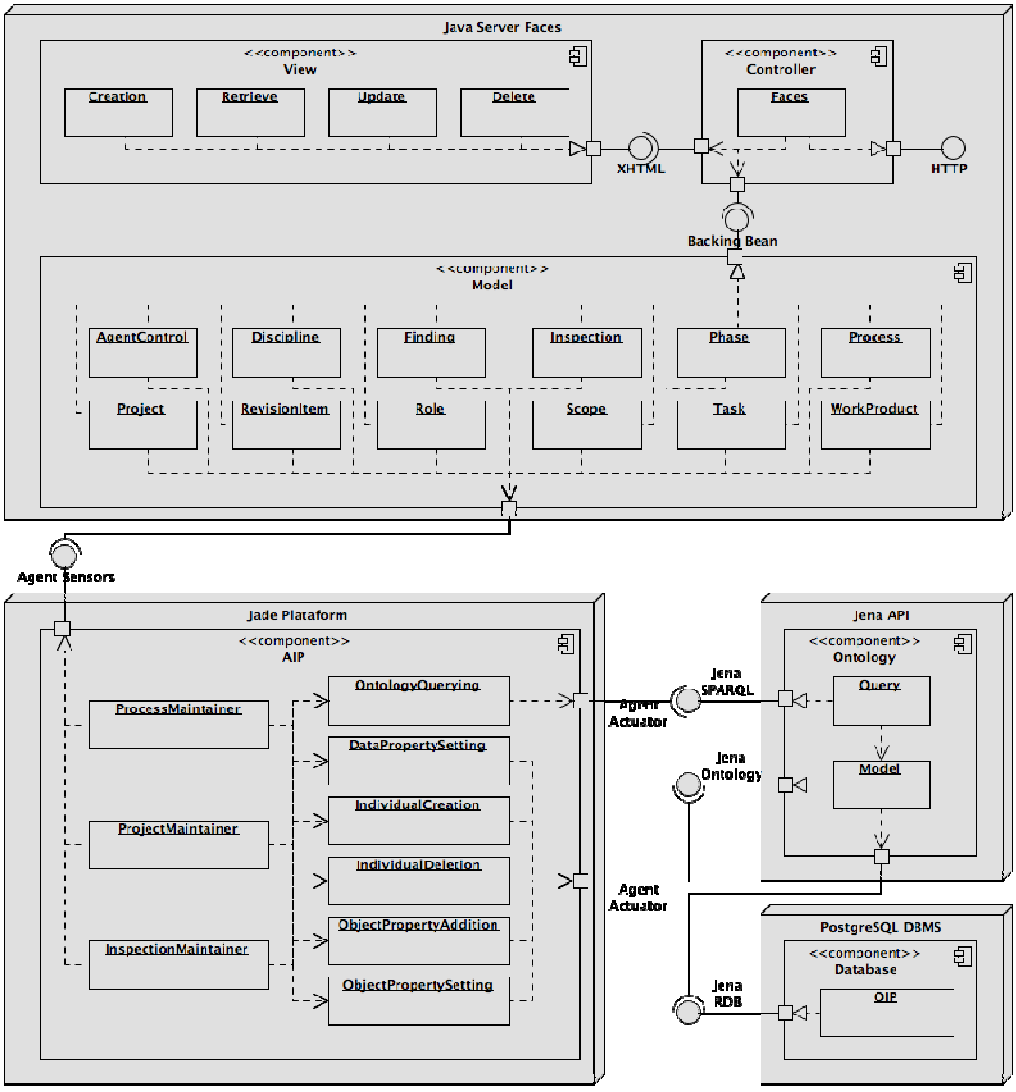

FIGURA 43 - Arquitetura do sistema acrescida da camada de aplicação.

A Figura 44 apresenta três telas básicas do protótipo de aplicação web. Na Figura 44 (A) é vista a tela de entrada do sistema, onde é possível inicializar os três agentes dentro da plataforma JADE. A Figura 44 (B) detalha o primeiro item do menu, *Process*, onde é feita a

manutenção de todos os cadastros necessários para definir o processo de software, insumo básico para as inspeções de garantia da qualidade. A Figura 44 (C) detalha o segundo item do menu, *Inspection*, onde, além de manter cadastros complementares, realizam-se e registram-se os resultados das inspeções de garantia da qualidade em projetos de desenvolvimento de software.

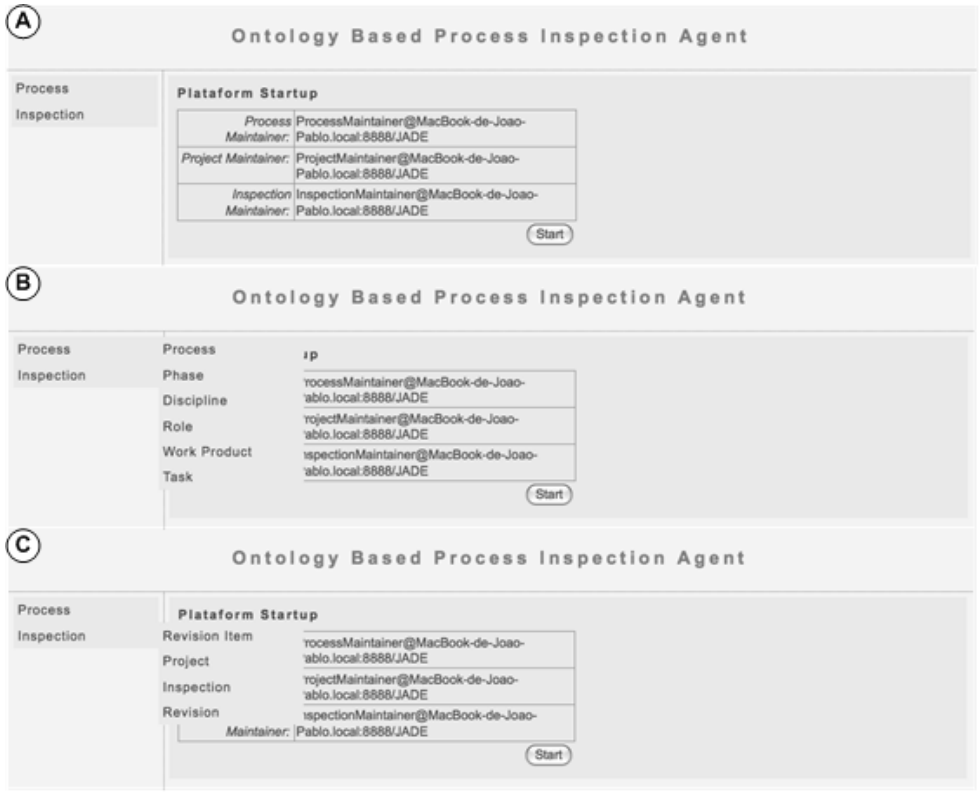

FIGURA 44 - Telas básicas do protótipo de aplicação web.

A Figura 45 apresenta os três principais formulários do protótipo de aplicação web. Pode ser visto na Figura 45 (A) o formulário para criação de uma tarefa, sendo necessário informar um nome (*Name*) e uma descrição complementar para a tarefa (*Description*). Também se faz necessário selecionar uma disciplina para a tarefa (*Discipline*), uma ou mais fases que contém esta tarefa (*Phase*), definir um papel responsável (*Role*) e indicar quais produtos de trabalho são gerados com a tarefa (*WorkProduct*).

Na Figura 45 (B) é apresentado o formulário para criação de inspeções de garantia da qualidade, sendo necessário informar um nome (*Name*), uma descrição complementar, (*Description*), um analista responsável (*Analyst*), e uma data de execução (*Date*). Complementarmente, faz-se necessário selecionar um projeto alvo (*Project*) e um escopo para a inspeção (*Phase*). Como já visto, os demais dados relacionados com uma inspeção de garantia da qualidade são automaticamente definidos pelo próprio AIP.

Por fim, a Figura 45 (C) mostra o formulário de registro de resultados obtidos com a aplicação de um item de revisão a um determinado produto de trabalho. Na parte superior,

pode ser visto um cabeçalho que identifica o projeto, a inspeção, o produto de trabalho verificado e o item de revisão aplicado (*Project*, *Inspection*, *WorkProduct* e *RevisionItem*). Na parte inferior, pode ser visto a identificação da revisão (*Revision*) e três campos onde o analista responsável toma a decisão sobre a aderência do produto de trabalho inspecionado. Caso ele decida que o item não se aplica, deve justificar em *Justification For Not Apply*. Caso ele decida que o item está aderente, deve informar a evidência que corrobora com sua decisão em *Evidence of Adherence*. Caso ele decida que o item não está aderente, ele deve descrever a não conformidade em *Found Issue*.

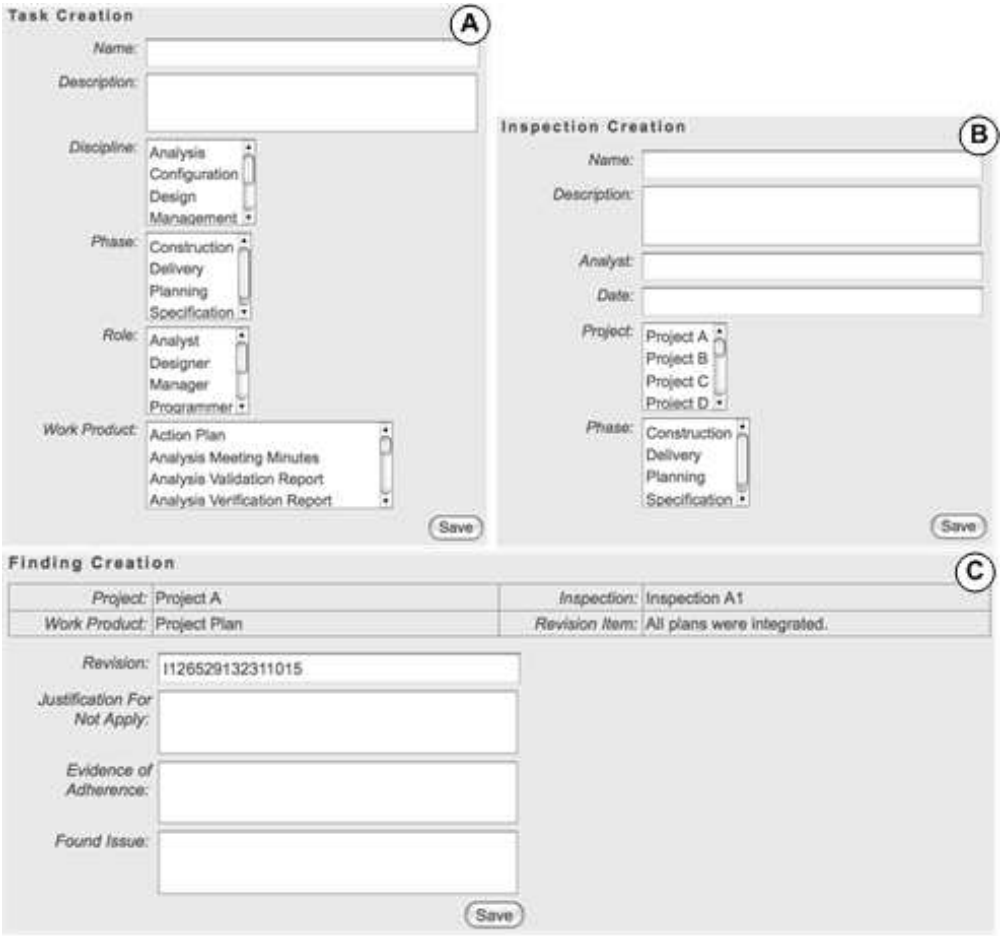

FIGURA 45 - Principais formulários do protótipo de aplicação web.

5.5 COMPARATIVO COM TRABALHOS RELACIONADOS

Com o objetivo de evidenciar os diferencias do sistema de apoio às inspeções de garantia da qualidade apresentado neste trabalho, foi realizada uma análise comparativa desta solução com as soluções apresentadas nos trabalhos relacionados do Capítulo 4. Tal análise teve como critério o mesmo utilizado na Seção 4.2, tanto para comparação com ontologias,

quanto para a comparação com agentes. A Tabela 11 reapresenta o quadro comparativo da Seção 4.2, acrescida da solução apresentada neste Capítulo.

TABELA 11 - Comparativo com trabalhos relacionados.

	Critérios	T1	T2	T3	T4	T5	T6
Ontologias	Responde consultas sobre tarefas, papéis ou produtos de trabalho pertencentes a um processo.	Sim	Sim	Sim	Sim	---	Sim
	Responde consultas sobre itens de revisão aplicáveis a um produto de trabalho.	Não	Não	Não	Não	---	Sim
	Responde consultas sobre inspeções realizadas em projeto de desenvolvimento.	Não	Não	Não	Não	---	Sim
	Responde consultas sobre escopo de uma inspeção de garantia da qualidade.	Sim	Não	Sim	Sim	---	Sim
	Responde consultas sobre resultado de uma inspeção de garantia da qualidade.	Não	Não	Não	Não	---	Sim
	Responde consultas sobre papéis responsáveis por cada não conformidade.	Não	Não	Não	Não	---	Sim
Agentes	Capacidade de prover a manutenção de um cadastro de processos de software.	Não	Não	Não	Sim	Não	Sim
	Capacidade de prover a manutenção de um cadastro de projetos de software.	Não	Não	Não	Não	Sim	Sim
	Capacidade de prover a manutenção de um cadastro de características de qualidade.	Não	Não	Não	Sim	Sim	Sim
	Capacidade de prover a manutenção de um cadastro de inspeções de garantia da qualidade.	Não	Não	Não	Não	Sim	Sim
	Capacidade de definir automaticamente o escopo de inspeções de garantia da qualidade.	Não	Não	Não	Não	Não	Sim
	Capacidade de enquadrar automaticamente os resultados obtidos na verificação de artefatos.	Não	Não	Não	Não	Não	Sim
	Capacidade de designar automaticamente os papéis responsáveis pelas não conformidades.	Não	Não	Não	Não	Não	Sim
	Capacidade de calcular automaticamente o indicador de aderência ao processo.	Não	Não	Não	Sim	Sim	Sim

LEGENDA:
T1 - Ontologia de Processo de Software (FALBO, 1998)
T2 - Ontologia de Modelos de Processo de Software (LIAO; QU; LEUNG, 2005)
T3 - Representação do Conhecimento em Processo de Software (HE et al., 2006)
T4 - Agente Inteligente Baseado em uma Ontologia de PPQA (WANG; LEE, 2007)
T5 - Web Service Inteligente para Avaliações de PPQA (WANG; LEE, 2008)
T6 - Sistema de Apoio às Inspeções de Garantia da Qualidade

5.6 FECHAMENTO DO CAPÍTULO

O presente Capítulo apresentou a visão de solução desenvolvida ao longo deste trabalho para o problema relacionado com a otimização das atividades oriundas de inspeções de garantia da qualidade, deixando claro que o sistema proposto não pretende eliminar a interação humana em inspeções, e sim automatizar atividades específicas aumentando a produtividade dos colaboradores envolvidos. Do ponto de vista tecnológico, uma contribuição

é a integração de ferramentas e tecnologias específicas, como ontologias e agentes, para resolver problemas de engenharia de software.

A OIP atende plenamente ao que se propôs, entretanto, não esgota as possibilidades de representação do domínio de inspeções de garantia da qualidade. Entende-se que para um primeiro mapeamento de um domínio de conhecimento é uma boa prática cobrir o mínimo e necessário para o escopo proposto. Após verificar e validar este domínio se torna plenamente viável e aconselhável o refinamento do modelo para cobrir conceitos complementares ou correlacionados.

O AIP mostrou-se plenamente capaz de automatizar as atividades relacionadas à definição de escopo, enquadramento de resultados durante as inspeções e alocação de papéis responsáveis por não conformidades. Entretanto, entende-se que se os agentes forem dotados de capacidades cognitivas, haverá um aumento potencial na capacidade de automação de atividades, como por exemplo, a antecipação da identificação dos desvios para o mesmo momento de execução do processo. Tal característica reforçaria a autonomia, pró-atividade e sociabilidade dos agentes.

Como fechamento do sistema proposto, este Capítulo apresentou um protótipo de aplicação web capaz de interagir com o AIP. O objetivo deste protótipo é ser uma ferramenta de verificação e validação, tanto da OIP quanto do AIP, e está longe de caracterizar um produto final. Em contra partida, o protótipo aponta como viável a construção de um produto de fato, o qual, além de interagir com o AIP, implemente padrões de segurança, interface e funcionalidades demandadas por qualquer sistema de informação convencional.

Ao analisar os resultados da avaliação comparativa de cada trabalho, nota-se que o sistema de apoio às inspeções de garantia da qualidade atende a todos os critérios. Tal característica se justifica no fato de que o sistema foi projetado para tal, ou seja, o sistema apresentado neste trabalho é mais específico em seu nicho de atuação, o que lhe permite uma maior aderência às necessidades de automação de inspeções de garantia da qualidade. Entretanto, esta especificidade limita a expansão do sistema para outras áreas da engenharia de software não relacionadas à garantia da qualidade de software.

6 VERIFICAÇÃO E VALIDAÇÃO DO SISTEMA

Atividades de verificação e validação, como já visto, garantem que o produto atende às suas especificações e é plenamente funcional quando inserido em um ambiente real de uso. Uma estratégia neste sentido, além de realizar testes funcionais, deve garantir a experimentação em um ambiente que tenha as mesmas condições de um ambiente de produção. Com base nisto, é apresenta a estratégia de verificação e validação do sistema de apoio às inspeções de garantia da qualidade.

O presente Capítulo traz na Seção 6.1 a contextualização do ambiente utilizado como estudo de caso. A Seção 6.2 descreve a situação atual deste ambiente, no que se refere aos resultados obtidos com a implementação de PPQA. A Seção 6.3 descreve as etapas de verificação e validação do sistema, apresentando seus respectivos resultados. A Seção 6.4 faz uma análise comparativa dos dados obtidos sem e com o uso do sistema de apoio às inspeções. A Seção 6.5 faz o fechamento do Capítulo.

6.1 CONTEXTUALIZAÇÃO DO AMBIENTE

A estratégia de verificação e validação elaborada para o sistema de apoio às inspeções de garantia da qualidade teve como cenário de execução um ambiente real de desenvolvimento de software. Este ambiente é parte da estrutura de uma empresa fornecedora de serviços de desenvolvimento, a qual está no mercado nacional e internacional a mais de 18 anos e conta com uma estrutura de aproximadamente 1000 colaboradores. Em 2007, a empresa obteve junto ao *Software Engineering Institute* (SEI) o laudo Nível 2 de Maturidade do CMMI. Para isto, foi necessário atender a algumas PAs do modelo, dentre estas, PPQA. Para atender aos objetivos e práticas definidas pelo CMMI para PPQA, a empresa lançou mão da estratégia apresentada na Seção 3.3 deste trabalho, a qual se encontrava em produção até a escrita desta dissertação.

A partir de 2009, a empresa começou a trabalhar na melhoria de seu processo para obter o laudo Nível 3 de Maturidade, que tem como uma de suas características o monitoramento e melhoria contínua de processos de software. A institucionalização de PAs, como *Organizacional Process Definition* (OPD) e *Organizational Process Focus* (OPF), apontaram a necessidade de se otimizar as atividades de inspeção de garantia da qualidade na

empresa. A partir deste cenário, surge a possibilidade de experimentação do sistema em um ambiente real de aplicação.

6.2 SITUAÇÃO ATUAL

Para viabilizar a comparação de resultados, foi realizado o trabalho de medição da situação atual do processo de inspeção de garantia da qualidade, conforme descrito na Seção 3.3 deste trabalho. Para calcular os indicadores de Cobertura da Inspeção e Aderência ao Processo foi selecionada uma amostra de 10 projetos executados entre 2007 e 2009. Para cada projeto, foram selecionadas 4 inspeções de garantia da qualidade, sendo uma para cada fase definida no processo de desenvolvimento da empresa. Isto totaliza uma amostra de 40 inspeções.

O gráfico da Figura 46 mostra o comportamento do indicador de Cobertura das Inspeções, calculado a partir da amostra anteriormente definida. Pode-se observar que a fase com melhor cobertura é a de Planejamento, com 2,11 itens de revisão para cada produto de trabalho. A fase de Especificação apresenta uma queda considerável, onde a cobertura é de 1,68. A fase de Desenvolvimento melhora o cenário de cobertura, com o valor de 1,92. Já a fase de Entrega novamente derruba o indicador, tendo esta fase a menor cobertura, apenas 1,63.

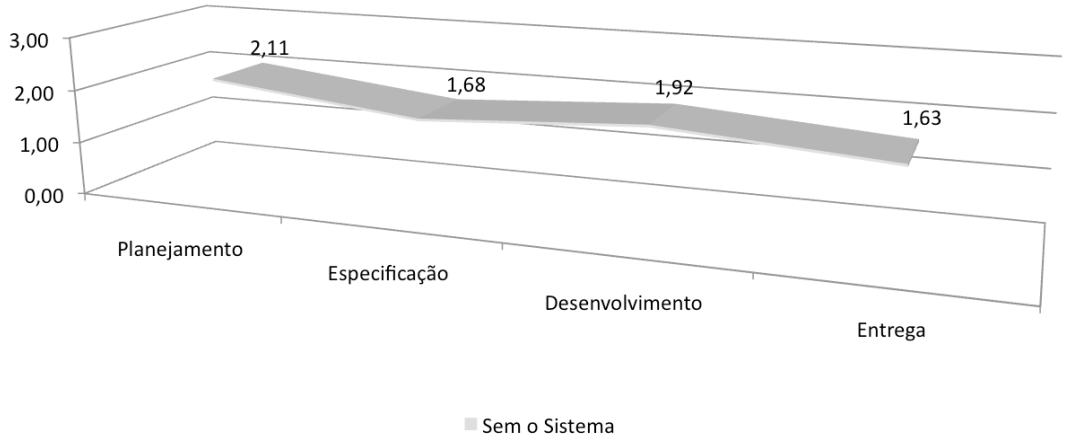

FIGURA 46 - Gráfico para o indicador de Cobertura da Inspeção sem o sistema.

Também pode ser observada no gráfico da Figura 46, a homogeneidade dos dados. Tal observação se confirma através do cálculo do coeficiente de variação da amostra, sendo este de 10,37%. Outro dado relevante de se observar é o valor médio do indicador de Cobertura da Inspeção, o qual demonstra a granularidade das inspeções. Considerando o universo da amostra, 40 inspeções, o indicador possui um valor médio de 1,86 itens de revisão por

produto de trabalho, o que indica uma granularidade muito grossa. Inspeções com granularidade grossa tendem a não ser objetivas como recomenda o CMMI.

A Figura 47 apresenta um gráfico que consolida os valores de Aderência ao Processo da amostra estabelecida. Inicialmente, pode-se observar um valor de Aderência ao Processo de 80,78% na fase de Planejamento. Na sequência, há uma elevação deste valor para 86,27%. Porém, as fases de Desenvolvimento e Entrega quebram a tendência de crescimento, gerando para o indicador os valores 83,91% e 81,32%, respectivamente.

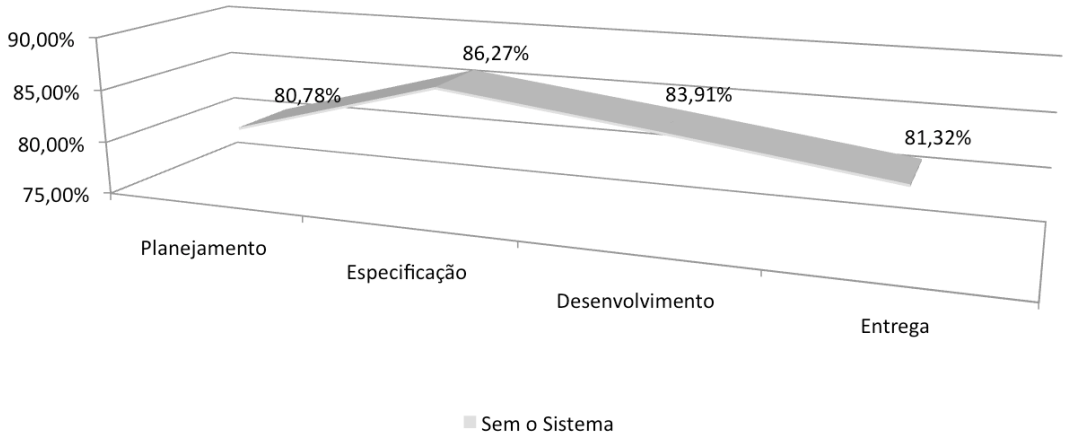

FIGURA 47 - Gráfico para o indicador de Aderência ao Processo sem o sistema.

Ao observar o comportamento do gráfico da Figura 47, nota-se que o universo desta amostragem é menos homogêneo, se comparado com o indicador anteriormente apresentado. Mas, estatisticamente, o coeficiente de variação ainda é considerado baixo, pois apresenta um valor igual a 13,31%. O indicador de Aderência ao Processo tem como média final da amostragem um valor de 83,07%, ou seja, em média, aproximadamente 20% do processo definido pela empresa não é seguido pelas equipes em projetos de desenvolvimento.

6.3 VERIFICAÇÃO E VALIDAÇÃO

O sistema de apoio às inspeções de garantia da qualidade foi verificado inicialmente com testes unitários, onde cada unidade do AIP foi testada em termos de entradas e saídas. Em outras palavras, cada método foi devidamente invocado, passando seus respectivos parâmetros, e o resultado gerado foi avaliado contra o comportamento esperado. Em um segundo momento foram realizados testes integrados, os quais garantem a correta troca de mensagem entre os métodos do AIP. Para tal, ainda em ambiente de desenvolvimento, se executou no sistema um conjunto de cenários aleatórios, que experimentaram todas as funcionalidades do protótipo construído.

A validação do sistema de apoio às inspeções demandou de uma estratégia mais elaborada, onde, o primeiro passo, foi mapear o processo de desenvolvimento da empresa. Para isto, foi realizada uma análise nos documentos que definem todos os processos, procedimentos e padrões aplicáveis ao desenvolvimento de software. Após, o processo foi devidamente inserido no sistema através da interface web apresentada na Seção 5.4. No Apêndice A é apresentado um relatório consolidado de todas as tarefas cadastradas com suas respectivas disciplinas e fases.

Como segundo passo, foi necessário mapear todas as características de qualidade aplicáveis a cada produto de trabalho existente no processo. Desta forma, estabeleceu-se uma coleção de itens de revisão. Cabe salientar que um item de revisão pode ser aplicado a um ou mais produtos de trabalho, e que um produto de trabalho pode possuir vários itens de revisão relacionados. O Apêndice B traz na íntegra a listagem de itens de revisão inseridos na ontologia, apresentando ainda seus respectivos produtos de trabalho relacionados.

O terceiro passo se caracterizou pela execução de inspeções de garantia da qualidade, tendo como ferramenta de apoio o sistema apresentado neste trabalho. Para viabilizar a comparação de resultados, foi utilizada a mesma amostra de inspeções apresentada na Seção 6.2. Tais inspeções foram plenamente inseridas no sistema, como demonstrado pelo Apêndice C. Para mostrar a execução de uma inspeção de garantia da qualidade, o Apêndice D apresenta um relatório de uma inspeção específica, onde podem ser vistos dados gerais da inspeção, o escopo da inspeção e a lista de não conformidades com seus respectivos responsáveis.

Por fim, como quarto passo, foi realizada a coleta de resultados. Os resultados coletados são os indicadores de Cobertura da Inspeção e Aderência ao Processo. Após as devidas coletas, deu-se a realização de algumas estatísticas, tais como: média e coeficiente de variação. Para um melhor entendimento, os dados dos indicadores foram consolidados por fase e apresentados em gráficos. O Apêndice C apresenta uma tabela com todos os dados coletados para fins estatísticos, tanto da situação atual sem o sistema, quanto do novo cenário com o sistema.

6.4 ANÁLISE DOS RESULTADOS

O primeiro indicador a ser analisado é o de Cobertura da Inspeção, lembrando que quanto maior for este valor, mais itens de revisão se têm para um produto de trabalho, logo,

temos uma granularidade mais fina e uma inspeção mais objetiva. O gráfico da Figura 48 apresenta uma comparação entre os valores do indicador sem e com o sistema de apoio às inspeções de garantia da qualidade, consolidados por fases do processo. A fase de Planejamento apresenta uma cobertura de 2,94 itens de revisão para cada produto de trabalho. As fases de Especificação e Desenvolvimento apresentam uma pequena tendência de crescimento, pois seus valores são 3,03 e 3,08 itens de revisão por produto de trabalho, respectivamente. Porém, a fase de Entrega manteve o padrão de queda anteriormente identificado, gerando o valor de 2,59 para o indicador.

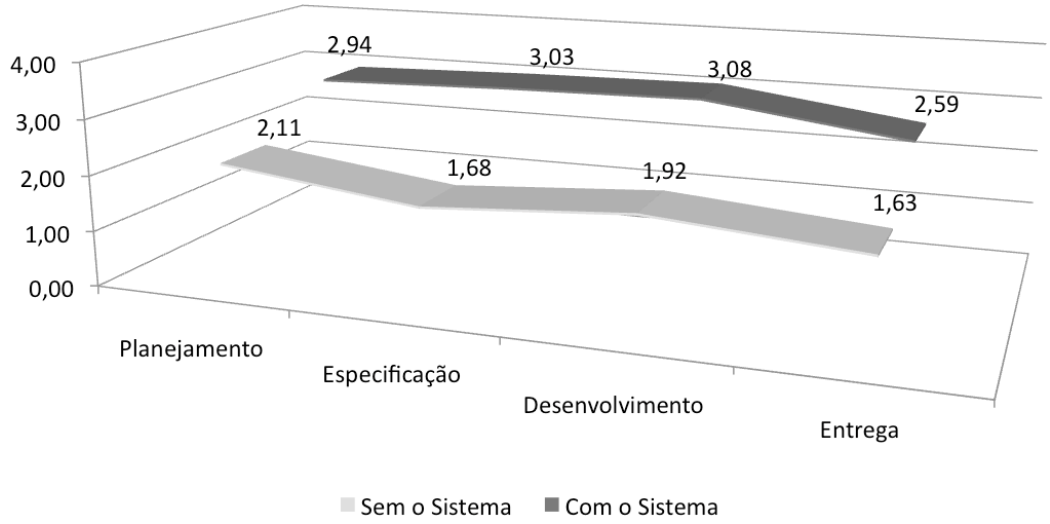

FIGURA 48 - Gráfico para o indicador de Cobertura da Inspeção com o sistema.

Ao analisar a Figura 48, a primeira conclusão que se chega é que a cobertura aumentou. O cálculo da média para o indicador, considerando o uso do sistema, ficou em 2,95 itens de revisão por produto de trabalho, o que indica que a granularidade das inspeções com o uso do sistema é 58,76% mais fina do que antes. Complementarmente, também se observa que a amostra é mais homogênea, se comparada com os valores sem o sistema. O cálculo do coeficiente de variação endossa esta percepção, gerando um valor de 6,34%. Em uma análise final, as duas amostras apresentam queda de cobertura na fase de Entrega. É possível levantar duas hipóteses de causa para este comportamento: não foram identificadas todas as características de qualidade para os produtos de trabalho desta fase ou o processo não está escrito de forma a explicitar tais características de qualidade.

O gráfico da Figura 49 faz a comparação do indicador de Aderência ao Processo sem e com o uso do sistema de apoio às inspeções de garantia da qualidade, onde se tem na fase Planejamento uma aderência de 80,36%, subindo para 87,73% na fase de Especificação. Após, o gráfico apresenta o mesmo padrão de queda para as fases de Desenvolvimento e Entrega, onde a primeira tem a aderência de 85,15% e a segunda de 81,93%. Cabe salientar

que o sistema apresentado não tem a pretensão de aumentar ou diminuir a aderência dos projetos ao processo de desenvolvimento, logo, a análise deste indicador serve para garantir a precisão das inspeções, demonstrada pelo mesmo padrão de resultados obtidos sem e com o sistema.

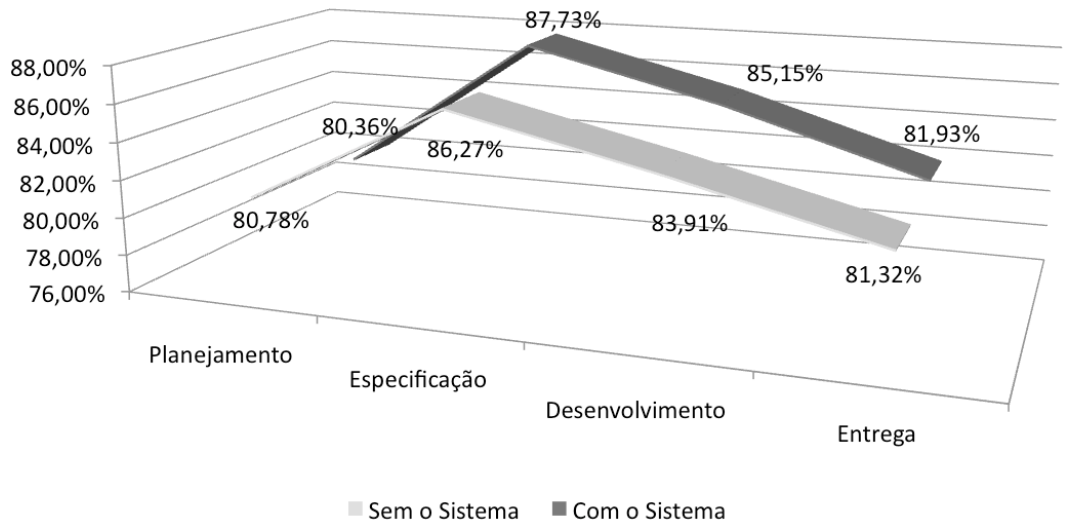

FIGURA 49 - Gráfico para o indicador de Aderência ao Processo com o sistema.

Está explícito no gráfico da Figura 49 o comportamento similar das duas séries. A comparação das médias reforça esta análise, pois com o uso do sistema se obteve uma aderência média de 83,79%, o que dá uma variação de 0,87% da média sem o sistema para a média com o sistema. Outro dado que reforça a homogeneidade da amostra gerada com o uso do sistema é o coeficiente de variação, o qual também é baixo e um pouco menor que o obtido sem o sistema, este valor é de 11,49%. Este contexto estatístico comprova que o aumento da cobertura de uma inspeção através de processos automatizados, quando aplicado a um mesmo cenário, é capaz de gerar resultados similares, onde as diferenças entre percentuais se caracterizam como um ajuste fino e não como um reenquadramento de aderência.

Como análise final foi realizada a comparação do esforço demandado para execução das inspeções de garantia da qualidade. Por questões de confidencialidade das informações relativas ao esforço realizado pela empresa do estudo de caso, não são apresentados os dados reais obtidos durante o experimento, sendo utilizada para análise a variação média do esforço. Este indicador se caracteriza por ser a variação percentual entre a média do esforço gasto em inspeções sem o sistema e a média do esforço gasto em inspeções com o sistema. Após as devidas coletas e medições, apurou-se uma variação de 1,89% para menos, ou seja, o esforço médio de execução das inspeções com o sistema foi menor que o esforço médio da execução das inspeções sem o sistema.

6.5 FECHAMENTO DO CAPÍTULO

O presente Capítulo trouxe como proposta, a verificação e validação do sistema de apoio às inspeções de garantia da qualidade apresentado no Capítulo 5. Para tal, é descrito como a verificação é feita, em termos de testes unitários e integrados, e qual foi a estratégia de validação, envolvendo a caracterização de um cenário real de aplicação, levantamento de amostras para experimentação, coleta de dados e análise de resultados. Além disso, o Capítulo busca dar subsídios estatísticos para a comparação dos cenários sem o sistema e com o sistema apresentado neste trabalho.

Os resultados obtidos são plenamente satisfatórios, pois comprovam que um sistema capaz de automatizar algumas das atividades envolvidas em um processo de PPQA, inspeções de garantia da qualidade mais especificamente, podem de fato agregar valor, aumentando a cobertura de cada inspeção realizada, sem impactar significativamente na aderência, pois os cenários das inspeções eram o mesmo, logo, o resultado deve ser próximo. Também se deve evidenciar que o impacto no custo, horas/homem, das inspeções foi mínimo e para menos, o que reforça o atendimento dos objetivos do trabalho.

7 CONCLUSÕES

O sistema de apoio às inspeções de garantia da qualidade apresentado neste trabalho se mostrou plenamente viável e em conformidade com os objetivos inicialmente traçados. Esta conclusão é fundamentada na análise dos resultados obtidos ao final da atividade de verificação e validação, onde se demonstraram os ganhos gerados com o uso do sistema em inspeções de garantia da qualidade. Na análise dos resultados, destaca-se o indicador de Cobertura da Inspeção, apresentando um aumento de 58,76% em relação ao mesmo indicador medido sem o uso do sistema. Ainda cabe salientar, que as inspeções geraram valores de Aderência ao Processo muito próximo dos valores gerados sem o sistema, isto demonstra que a precisão das inspeções foi mantida. No que se refere ao esforço, pode-se afirmar que o impacto além de pequeno, menos de 2% de variação, foi positivo, pois na média a quantidade de horas demandada foi menor com o uso do sistema.

No que se refere à OIP, caracterizada por mapear os conceitos mínimos e necessários para representar o domínio de conhecimento de inspeções de garantia da qualidade, pode-se concluir que é viável o uso de ontologias em aplicações orientadas a engenharia de software. Entretanto, convém observar a dificuldade em estabelecer modelos que atendam a necessidade da aplicação. Mesmo utilizando uma metodologia para a construção de ontologias, é comum a confusão com modelos de entidades-relacionamentos, o que demanda uma série de refatorações até atingir o modelo final. O ferramental OWL, SPARQL e Jena se mostrou robusto, porém a curva de aprendizado destas ferramentas não é rápida, há poucos exemplos práticos e são necessários bons conhecimentos de orientação a objetos, Java e XML.

Ao analisar o AIP, sendo este um sistema multiagente responsável por perceber ações do usuário e manipular indivíduos na ontologia de acordo com critérios definidos, pode-se concluir que este é plenamente funcional. O encapsulamento da manutenção da ontologia pelo agente viabiliza o uso do sistema por outras aplicações, de forma rápida e transparente. Entretanto, entende-se que não foram esgotadas as possibilidades de autonomia e pró-atividade do agente quanto à manipulação de indivíduos. Acredita-se que a exploração destas capacidades, pode otimizar outras dimensões de uma inspeção de garantia da qualidade não abordadas neste trabalho, como por exemplo, a redução de custos através da realização de inspeções em tempo real de execução do projeto. No âmbito tecnológico, JADE provê as

estruturas necessárias para a construção de sistemas multiagentes. Além disso, JADE tem uma curva de aprendizado mais otimizada, pois a plataforma é bem documenta e exemplificada.

Em última instância, é feita a análise do protótipo de interface para interação com o AIP e a OIP. Esta análise aponta o protótipo como o ponto menos explorado por este trabalho, isto porque, mesmo com o uso de JSF e *Facelets*, as interfaces são pobres do ponto de vista ergonômico e de usabilidade. Porém, como uma interface rebuscada não está dentre os objetivos deste trabalho, não houve impacto nos resultados gerados. Outra questão, também observada com o uso do protótipo, é que a partir de certo número de inspeções criadas, o que reflete em um número considerável de indivíduos na ontologia, o tempo de resposta das consultas começou a degradar. Tal degradação é plenamente tratável, considerando a possibilidade de construção de um produto propriamente dito.

7.1 PRINCIPAIS CONTRIBUIÇÕES

A primeira contribuição deste trabalho diz respeito à estruturação de uma estratégia de implementação de garantia da qualidade de software, visto que não se encontrou na literatura pesquisada uma abordagem capaz de orientar na implementação das práticas CMMI estabelecidas para PA de PPQA. Este trabalho, além de apresentar esta abordagem, comprovou sua eficiência em um ambiente real de desenvolvimento. Tal ambiente foi avaliado pelo SEI para obtenção do laudo de nível 2 de maturidade e foi considerado apto.

A segunda contribuição deste trabalho está relacionada com a modelagem do domínio de conhecimento de garantia da qualidade de software através de uma ontologia. Esta contribuição se reforça pelo seu caráter inovador, já que nenhuma das ontologias estudadas cobriu plenamente o domínio de conhecimento de inspeções de garantia da qualidade de software. Complementarmente, a ontologia mostrou-se robusta ao ser base para a construção de um sistema de apoio às inspeções de garantia da qualidade.

A terceira contribuição deste trabalho remete ao projeto de agentes de software capazes de automatizar atividades específicas de inspeções de garantia da qualidade. O sistema multiagente construído, mesmo com uma arquitetura simples, foi capaz de automatizar as atividades de definições de escopo de inspeções, enquadramento de resultados em inspeções e alocação de papéis responsáveis por não conformidades. Tal capacidade garantiu o aumento da cobertura das inspeções, a manutenção dos percentuais de aderência e a manutenção do esforço despendido.

7.2 TRABALHOS FUTUROS

Com o objetivo de dar continuidade ao objeto de pesquisa apresentado neste trabalho, é proposta uma nova abordagem para implementação de garantia da qualidade. A abordagem atual, a qual guiou todo o desenvolvimento do sistema apresentado, é totalmente reativa, ou seja, tomam-se ações de correção quando há não conformidades. O problema é que, dependendo do momento do projeto de desenvolvimento, não é mais possível retomar ao caminho original. A visão que se tem desta nova abordagem é algo altamente pró-ativo, capaz de interagir com as equipes de desenvolvimento em tempo real, indicando e sinalizando ações que evitem problemas de aderência aos processos.

Tecnicamente, esta proposta pode ser viabilizada também através de agentes e ontologias. Neste caso, não se teria apenas agentes reativos simples, mas um sistema multiagente cognitivo distribuído em toda a infraestrutura de desenvolvimento. Este sistema multiagente, não seria somente responsável por manipular uma ontologia de inspeção de garantia da qualidade, mas sim uma coleção de ontologias específicas que viabilizariam a avaliação da aderência dos artefatos em tempo real de execução. Entende-se que esta abordagem, além da inovação científica e tecnológica, transportaria a disciplina de garantia da qualidade para um novo patamar, onde ela não aponta mais problemas e sim os evita.

REFERÊNCIAS

ABRAN, A.; MOORE, J. W.; BOURQUE, P.; DUPUIS, R. **Guide to the Software Engineering Body of Knowledge**. Los Alamitos: IEEE Computer Society, 2004.

BASTOS, A.; RIOS, E.; CRISTALLI, R.; MOREIRA, T. **Base de Conhecimento em Teste de Software.** São Paulo: Martins Fontes, 2007.

BELLIFEMINE, F.; CAIER, G.; GREENWOOD, D. **Developing Multi-Agent Sytems with JADE.** Chichester: Jonh Wiley & Sons, 2007.

CARROLL, J. J.; DICKINSON, I.; DOLLIN, C.; REYNOLDS, D.; SEABORNE, A.; WILKINSON, K. **Jena: Implementing the Semantic Web Recommendations.** In: HP TECHNICAL REPORTS. 2003. Bristol. Technical Reports... Bristol: HP Labs, 2003.

CHRISSIS, M. B.; KONRAD, M.; SHRUM, S. **CMMI Guidelines for Process Integration and Product Improvement.** Boston: Addison Wesley, 2007.

DIETRICH, J.; ELGAR, C. **A Formal Description of Design Patterns Using OWL.** In: AUSTRALIAN SOFTWARE ENGINEERIN CONFERENCE. 2005. Brisbane. Proceedings... Los Alamitos: IEEE Computer Society, 2005.

DIETRICH, J.; JONES, N. **Using Social Networking and Semantic Web Technology in Software Engineering - Use Cases, Patterns, and a Case Study.** In: AUSTRALIAN SOFTWARE ENGINEERIN CONFERENCE. 2007. Melbourne. Proceedings... Los Alamitos: IEEE Computer Society, 2007.

FALBO, R. **Integração de Conhecimento em um Ambiente de Desenvolvimento de Software.** Rio de Janeiro: Universidade Federal do Rio de Janeiro, 1998.

GHIOTTO, P.; ORTEGA, M.; GRIMÁN, A.; MENDOZA, L.; PÉREZ, M. **Ontology Proposal for Quality Oriented Reuse.** In: INTERNATIONAL CONFERENCE ON INFORMTION REUSE AND INTEGRATION. 2006. Waikoloa Village. Proceedings... Los Alamitos: IEEE Computer Society, 2006.

GIORGINI, P.; KOLP, M.; MYLOPOULOS, J.; PISTORE, M. **Methodologies and Software Engineering for Agent Systems.** Boston: Springer US, 2004.

GRUBER, T. R. **Toward Principles for the Design of Ontologies Used for Knowledge Sharing.** ELSEVIER INTERNATIONAL JOURNAL HUMAN-COMPUTER STUDIES: Maryland Heights, v. 43, p. 907-928, nov/dez 1993.

GRUBER, T. R. **Ontology.** Disponível em: <http://tomgruber.org/writing/ontology-definition-2007.htm>. Acesso em: 01 de fev. 2010.

GUARINO, N. **Formal Ontology, Conceptual Analysis and Knowledge Representation**. ELSEVIER INTERNATIONAL JOURNAL HUMAN-COMPUTER STUDIES: Maryland Heights, v. 43, p. 625-640, nov/dez 1995.

HE, J.; YAN, H.; LIU, C.; JIN, M. **A Framework of Ontology-Supported Knowledge Representation in Software Process.** In: INTERNATIONAL CONFERENCE ON INTELLIGENT SYSTEMS AND KNOWLEDGE ENGINEERING. 2006. Chengdu. Proceedings... Paris: Atlantis Press, 2006.

HOSS, A. M.; CARVER, D. L. **Ontological Approach to Improving Design Quality**. In: AEROESPACE CONFERECE. 2006. Montana. Proceedings... Los Alamitos: IEEE Computer Society, 2006.

LARMAN, G. **Utilizando UML e Padrões: Uma introdução à análise e ao projeto orientado a objetos e ao Processo Unificado**. São Paulo: Bookman, 2004.

LIAO, L.; QU, Y.; LEUNG, H. K. N. **A Software Process Ontology and its Application.** In: WORKSHOP ON SEMANTIC WEB ENABLE SOFTWARE ENGINEERING. 2005. Galway. Proceedings... Paris: Atlantis Press, 2005.

MCBRIDE, B. **Jena: A SemaWeb Toolkit.** IEEE INTERNET COMPUTING: Los Alamitos, v. 6, p. 55-59, 2002.

MCGUINNESS, D. L; HARMELEN, F. V. **OWL Web Ontology Language Overview.** Disponível em: <http://www.w3.org/TR/owl-features/> Acesso em: 01 de fev. 2010.

NOY, N. F.; MCGUINNESS, D. L. **Ontology development 101: A guide to creating your first ontology.** In: SEMANTIC WEB WORKING SYMPOSIUM. 2001. Stanford. Proceedings... Los Alamitos: IEEE Computer Society, 2001.

KAIYA, H.; SAEKI, M. **Ontology Based Requirements Analysis: Lightweight Semantic Processing Approach**. In: INTERNATIONAL CONFERENCE ON QUALITY SOFTWARE. 2005. Melbourne. Proceedings... Los Alamitos: IEEE Computer Society, 2005.

PRESSMAN, R.S. **Software Engineering: A Practitioner's Approach.** Rio de Janeiro: McGraw-Hill, 2001.

PRUD'HOMMEAUX, E.; SEABORNE, A. **SPARQL Query Language for RDF.** Disponível em: < http://www.w3.org/TR/rdf-sparql-query/ >. Acesso em: 01 de fev. 2010.

REENSKAUG, T. **Models - Views - Controllers.** In: XEROX PARC TECHNICAL NOTE. 1979. Palo Alto. Technical Reports... Palo Alto: Xerox, 1979.

RUSSELL, S.; NORVIG, P. **Inteligência Artificial**. Rio de Janeiro: Campus, 2004.

SHADBOLD, N.; HALL, W.; BERNERS-LEE, T. **The Semantic Web Revisited.** IEEE INTELLIGENT SYSTEMS: Los Alamitos, v. 21, p. 96-101, 2006.

SILVA, I. **Projeto e Implementação de Sistemas Multi-agentes: O Caso Tropos.** Recife: Universidade Federal de Pernambuco, 2005.

SOMMERVILLE, I. **Software Engineering.** Boston: Addison Wesley, 2001.

SUN MICROSYSTEMS. **The Java EE 5 Tutorial.** Disponível em: <http://java.sun.com/javaee/5/docs/tutorial/doc/JavaEETutorial.pdf> Acesso em: 01 de fev. 2010.

WANG, M.; LEE, C. **An Intelligent Fuzzy Agent Based on PPQA Ontology for Supporting CMMI Assessment.** In: FUZZY SYSTEMS CONFERENCE. 2007. London. Proceedings... Los Alamitos: IEEE Computer Society, 2007.

WANG, M.; LEE, C. **An Intelligent PPQA Web Services for CMMI Assessment.** In: INTELLIGENT SYSTEMS DESIGN APPLICATIONS. 2008. Kaohsiung. Proceedings... Los Alamitos: IEEE Computer Society, 2008.

WOOLDRIDGE, M. **An Introdution to MultiAgent Systems.** Chichester: Jonh Wiley & Sons, 2002.

WONGTHONGTHAM, P., CHANG, E., DILLON, T. **Enhancing Software Engineering Project Information through Software Engineering Ontology Instantiations**. In: INTERNATIONAL CONFERENCE ON WEB INTELLIGENCE. 2006. Hong Kong. Proceedings... Los Alamitos: IEEE Computer Society, 2006.

WONGTHONGTHAM, P., CHANG, E., DILLON, T. **Ontology Modelling Notations for Software Engineering Knowledge Representation.** In: INTERNATIONAL CONFERENCE ON DIGITAL ECOSYSTEMS AND TECHNOLOGIES. 2007. Cairns. Proceedings... Los Alamitos: IEEE Computer Society, 2007.

APÊNDICE A - PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Process View

Identificator:	I125927142137995	**Name:**	Organizational Process Library
Description:	The organizational process library for software development.	**Version:**	2009

Task View

ID	Name	Discipline	Phase
I125927479098770	Keep Traceability	Analysis	[Construction]
I125927374923947	Apply Change Request	Configuration	[Construction]
I125927378658116	Approve Change Request	Configuration	[Construction]
I125927392726007	Create Baseline	Configuration	[Construction]
I125927396121074	Create Branch	Configuration	[Construction]
I125927403603800	Elaborate Change Request	Configuration	[Construction]
I125927440124755	Keep Configuration Item	Configuration	[Construction]
I125927602175959	Project Class	Design	[Construction]
I125927606181991	Project Database	Design	[Construction]
I125927660473092	Verify Project	Design	[Construction]
I125927475384846	Keep Plans	Management	[Construction]
I125927481700702	Management Project	Management	[Construction]
I125927573980752	Perform Milestone Meeting	Management	[Construction]
I125927578278208	Perform Progress Meeting	Management	[Construction]
I125927611303892	Report Project	Management	[Construction]
I125927382448143	Build System	Programming	[Construction]
I125927386037834	Codify System	Programming	[Construction]
I125927618651749	Test System	Test	[Construction]
I125927668023362	Verify Test Case	Test	[Construction]
I125927680230227	Write Test Case	Test	[Construction]
I125927399854744	Document System	Analysis	[Delivery]
I125927479098770	Keep Traceability	Analysis	[Delivery]
I125927651994648	Verify Documentation	Analysis	[Delivery]
I125927374923947	Apply Change Request	Configuration	[Delivery]
I125927378658116	Approve Change Request	Configuration	[Delivery]
I125927392726007	Create Baseline	Configuration	[Delivery]
I125927396121074	Create Branch	Configuration	[Delivery]
I125927403603800	Elaborate Change Request	Configuration	[Delivery]
I125927440124755	Keep Configuration Item	Configuration	[Delivery]
I125927475384846	Keep Plans	Management	[Delivery]
I125927481700702	Management Project	Management	[Delivery]
I125927493612166	Perform Administrative Closing	Management	[Delivery]
I125927564248790	Perform Closing Meeting	Management	[Delivery]
I125927573980752	Perform Milestone Meeting	Management	[Delivery]
I125927578278208	Perform Progress Meeting	Management	[Delivery]
I125927611303892	Report Project	Management	[Delivery]
I125927382448143	Build System	Programming	[Delivery]

I125927386037834	Codify System	Programming	[Delivery]
I125927431605300	Homologate System	Test	[Delivery]
I125927618651749	Test System	Test	[Delivery]
I125927365748767	Analyze Requirements	Analysis	[Planning]
I125927635900657	Validate Requirements	Analysis	[Planning]
I125927663854791	Verify Requirements	Analysis	[Planning]
I125927374923947	Apply Change Request	Configuration	[Planning]
I125927378658116	Approve Change Request	Configuration	[Planning]
I125927392726007	Create Baseline	Configuration	[Planning]
I125927403603800	Elaborate Change Request	Configuration	[Planning]
I125927440124755	Keep Configuration Item	Configuration	[Planning]
I125927591475476	Plan Configuration	Configuration	[Planning]
I125927416652662	Establish Budget	Management	[Planning]
I125927420619847	Establish Mensuration	Management	[Planning]
I125927424638968	Establish Schedule	Management	[Planning]
I125927435505907	Integrate Plans	Management	[Planning]
I125927567625527	Perform Kick-off Meeting	Management	[Planning]
I125927573980752	Perform Milestone Meeting	Management	[Planning]
I125927596828634	Plan Risk	Management	[Planning]
I125927631754659	Validate Planning	Management	[Planning]
I125927655395437	Verify Planning	Management	[Planning]
I125927389486972	Construct Wireframes	Analysis	[Specification]
I125927479098770	Keep Traceability	Analysis	[Specification]
I125927485429268	Map Business Concept	Analysis	[Specification]
I125927498643564	Perform Analysis Meeting	Analysis	[Specification]
I125927587883613	Perform Wireframe Meeting	Analysis	[Specification]
I125927622248505	Validate Analysis	Analysis	[Specification]
I125927647881976	Verify Analysis	Analysis	[Specification]
I125927675128862	Verify Wireframe	Analysis	[Specification]
I125927688616807	Write Use Cases	Analysis	[Specification]
I125927374923947	Apply Change Request	Configuration	[Specification]
I125927378658116	Approve Change Request	Configuration	[Specification]
I125927392726007	Create Baseline	Configuration	[Specification]
I125927396121074	Create Branch	Configuration	[Specification]
I125927403603800	Elaborate Change Request	Configuration	[Specification]
I125927440124755	Keep Configuration Item	Configuration	[Specification]
I125927371216702	Analyze Technical Solution	Design	[Specification]
I125927614566412	Specify Architecture	Design	[Specification]
I125927644931652	Validate Wireframes	Design	[Specification]
I125927475384846	Keep Plans	Management	[Specification]
I125927481700702	Management Project	Management	[Specification]
I125927573980752	Perform Milestone Meeting	Management	[Specification]
I125927578278208	Perform Progress Meeting	Management	[Specification]
I125927611303892	Report Project	Management	[Specification]
I125927641219120	Validate Test Plan	Test	[Specification]

I125927672418186	Verify Test Plan	Test	[Specification]
I125927685190821	Write Test Plan	Test	[Specification]

APÊNDICE B - ITENS DE REVISÃO DE PRODUTOS DE TRABALHO

ID	Description	Work Product
I125943423456583	All plans were integrated.	[Project Plan]
I125943459984699	All scenarios were coverage.	[Test Case, Use Case Model, Wireframe]
I125943489723800	The agenda, discussions and decisions were recorded.	[Analysis Meeting Minutes, kick-off Meeting Minutes, Lesson Learned, Milestone Meeting Minutes, Progress Meeting Minutes, Wireframe Meeting Minutes]
I125943522580082	The agreement was recorded.	[Analysis Validation Report, Change Request Approved, Planning Validation Report, Product Accepted, Requirements Validation Report, Test Plan Validation Report, Wireframe Validation Report]
I125943553802224	The artifact was continuously updated.	[Action Plan, Analysis Meeting Minutes, Analysis Validation Report, Analysis Verification Report, Architecture Model, Baseline Applied, Branch Applied, Build, Change Applied, Change Request, Change Request Approved, Class Model, Conceptual Model, Configuration Item, Configuration Plan, Database Model, Documentation Verification Report, Homologation Errors Report, Interaction Model, kick-off Meeting Minutes, Lesson Learned, Milestone Meeting Minutes, Planning Validation Report, Planning Verification Report, Product Accepted, Progress Meeting Minutes, Progress Report, Project Budget, Project Measure, Project Plan, Project Schedule, Project Verification Report, Requirements Model, Requirements Validation Report, Requirements Verification Report, Risk Plan, Source Code, System Documentation, Technical Solution Analysis, Test Case, Test Case Verification Report, Test Errors Report, Test Plan, Test Plan Validation Report, Test Plan Verification Report, Traceability Applied, Use Case Model, Wireframe, Wireframe Meeting Minutes, Wireframe Validation Report, Wireframe Verification Report]
I125943630678975	The artifacts used were designed.	[Baseline Applied, Change Applied, Configuration Item]
I125943651892685	The baseline planning was adherent with configuration item planning.	[Configuration Plan]
I12594372158373	The changes were described, analyzed and planned.	[Change Request]
I125943741501611	The classes were designed with high cohesion and low coupling.	[Class Model, Conceptual Model]
I125950947316490	The component's ports and interfaces were defined.	[Architecture Model]
I125950951536778	The component's realization was defined.	[Architecture Model]
I125950959327723	The correct resource was used.	[Action Plan, Analysis Meeting Minutes, Analysis Validation Report, Analysis Verification Report, Architecture Model, Baseline Applied, Branch Applied, Build, Change Applied, Change Request, Change Request Approved, Class Model, Conceptual Model, Configuration Item, Configuration Plan, Database Model, Documentation Verification Report, Homologation Errors Report, Interaction Model, kick-off Meeting Minutes, Lesson Learned, Milestone Meeting Minutes, Planning Validation Report, Planning Verification Report, Product Accepted, Progress Meeting Minutes, Progress Report, Project Budget, Project Measure, Project Plan, Project Schedule, Project Verification Report, Requirements Model, Requirements Validation Report, Requirements Verification Report, Risk Plan, Source Code, System Documentation, Technical Solution Analysis, Test Case, Test Case Verification Report, Test Errors Report, Test Plan, Test Plan Validation Report, Test Plan Verification Report, Traceability Applied, Use Case Model, Wireframe, Wireframe Meeting Minutes, Wireframe Validation Report, Wireframe Verification Report]
I125950967212901	The database model was normalized.	[Database Model]
I125950976715789	The granularity of open defects was average.	[Homologation Errors Report, Test Errors Report]
I125950984971267	The interaction model was implemented all use cases.	[Interaction Model]

I125951005177355	The issue was properly open and assigned.	[Action Plan, Analysis Verification Report, Documentation Verification Report, Homologation Errors Report, Planning Verification Report, Project Verification Report, Requirements Verification Report, Test Case Verification Report, Test Errors Report, Test Plan Verification Report, Wireframe Verification Report]
I125951012390764	The measure was used for project monitoring.	[Project Measure]
I125951018457358	The process tailoring was performed.	[Project Schedule]
I125951026400915	The risk qualification was performed.	[Risk Plan]
I125951041407853	The strategy and technical was recorded.	[Test Plan]
I125951048691451	The technical problem and solution alternative was recorded.	[Technical Solution Analysis]
I125951067762563	The traceability was kept in de model.	[Architecture Model, Class Model, Conceptual Model, Database Model, Interaction Model, Requirements Model, Test Case, Use Case Model, Wireframe]

APÊNDICE C - INSPEÇÕES DE GARANTIA DA QUALIDADE

ID	Name	Description
I126061975194645	Inspection A1	Inspection of planning phase.
I126063115390460	Inspection A2	Inspection of specification phase.
I126468437646076	Inspection A3	Inspection of construction phase.
I126468720651502	Inspection A4	Inspection of delivery phase.
I126468847904559	Inspection B1	Inspection of planning phase.
I126469624514179	Inspection B2	Inspection of specification phase.
I126469945656177	Inspection B3	Inspection of construction phase.
I126470119676673	Inspection B4	Inspection of delivery phase.
I126476741440520	Inspection C1	Inspection of planning phase.
I126476828364671	Inspection C2	Inspection of specification phase.
I126476961233830	Inspection C3	Inspection of construction phase.
I126477175266988	Inspection C4	Inspection of delivery phase.
I126477952653912	Inspection D1	Inspection of planning phase.
I126478066710551	Inspection D2	Inspection of specification phase.
I126478478335768	Inspection D3	Inspection of construction phase.
I126479160700786	Inspection D4	Inspection of delivery phase.
I126503340476511	Inspection E1	Inspection of planning phase.
I126503826126073	Inspection E2	Inspection of specification phase.
I126504117855787	Inspection E3	Inspection of construction phase.
I126504836067729	Inspection E4	Inspection of delivery phase.
I126546666632975	Inspection F1	Inspection of planning phase.
I126546816143624	Inspection F2	Inspection of specification phase.
I126547135641508	Inspection F3	Inspection of construction phase.
I126547286764960	Inspection F4	Inspection of delivery phase.
I126547403702752	Inspection G1	Inspection of planning phase.
I126556175020012	Inspection G2	Inspection of specification phase.
I126556356568577	Inspection G3	Inspection of construction phase.
I126556457788595	Inspection G4	Inspection of delivery phase.
I126556551083998	Inspection H1	Inspection of planning phase.
I126556660948310	Inspection H2	Inspection of specification phase.
I126556842476072	Inspection H3	Inspection of construction phase.
I126556959760969	Inspection H4	Inspection of delivery phase.
I126563233031904	Inspection I1	Inspection of planning phase.
I126564719603104	Inspection I2	Inspection of specification phase.
I126566598712604	Inspection I3	Inspection of construction phase.
I126566840878859	Inspection I4	Inspection of delivery phase.
I126567018759210	Inspection J1	Inspection of planning phase.
I126567164466728	Inspection J2	Inspection of specification phase.
I126567321794586	Inspection J3	Inspection of construction phase.
I126567439906025	Inspection J4	Inspection of delivery phase.

APÊNDICE D - RESULTADO DE UMA INSPEÇÃO

Inspection View			
Identificator:	I126468720651502	**Name:**	Inspection A4
Description:	Inspection of delivery phase.	**Analyst:**	Analyst
Date:	01/10/09	**Project:**	Project A
Phase:	Delivery	**Got Adherence:**	91.07143% (14, 51, 5)

Scope View

ID	Work Product	Revision Item	Status
I126468720906462	Action Plan	The artifact was continuously updated.	Closed
I126468720906468	Action Plan	The correct resource was used.	Closed
I126468720906469	Action Plan	The issue was properly open and assigned.	Closed
I126468720906471	Baseline Applied	The artifact was continuously updated.	Closed
I126468720906472	Baseline Applied	The artifacts used were designed.	Closed
I126468720906473	Baseline Applied	The correct resource was used.	Closed
I126468720906475	Branch Applied	The artifact was continuously updated.	Closed
I126468720906476	Branch Applied	The correct resource was used.	Closed
I126468720906477	Build	The artifact was continuously updated.	Closed
I126468720906478	Build	The correct resource was used.	Closed
I126468720906480	Change Applied	The artifact was continuously updated.	Closed
I126468720906481	Change Applied	The artifacts used were designed.	Closed
I126468720906482	Change Applied	The correct resource was used.	Closed
I126468720906484	Change Request	The artifact was continuously updated.	Closed
I126468720906485	Change Request	The changes were described, analyzed and planned.	Closed
I126468720906486	Change Request	The correct resource was used.	Closed
I126468720906487	Change Request Approved	The agreement was recorded.	Closed
I126468720906489	Change Request Approved	The artifact was continuously updated.	Closed
I126468720906490	Change Request Approved	The correct resource was used.	Closed
I126468720906491	Configuration Item	The artifact was continuously updated.	Closed
I126468720906493	Configuration Item	The artifacts used were designed.	Closed
I126468720906494	Configuration Item	The correct resource was used.	Closed
I126468720906495	Configuration Plan	The artifact was continuously updated.	Closed
I126468720906496	Configuration Plan	The baseline planning was adherent with configuration item planning.	Closed
I126468720906498	Configuration Plan	The correct resource was used.	Closed
I126468720906499	Documentation Verification Report	The artifact was continuously updated.	Closed
I126468720906500	Documentation Verification Report	The correct resource was used.	Closed
I126468720906502	Documentation Verification Report	The issue was properly open and assigned.	Closed
I126468720906503	Homologation Errors Report	The artifact was continuously updated.	Closed
I126468720906504	Homologation Errors Report	The correct resource was used.	Closed
I126468720906505	Homologation Errors Report	The granularity of open defects was average.	Closed
I126468720906507	Homologation Errors Report	The issue was properly open and	Closed

		assigned.	
I126468720906508	Lesson Learned	The agenda, discussions and decisions were recorded.	Closed
I126468720906509	Lesson Learned	The artifact was continuously updated.	Closed
I126468720906510	Lesson Learned	The correct resource was used.	Closed
I126468720906512	Milestone Meeting Minutes	The agenda, discussions and decisions were recorded.	Closed
I126468720906513	Milestone Meeting Minutes	The artifact was continuously updated.	Closed
I126468720906514	Milestone Meeting Minutes	The correct resource was used.	Closed
I126468720906516	Product Accepted	The agreement was recorded.	Closed
I126468720906517	Product Accepted	The artifact was continuously updated.	Closed
I126468720906518	Product Accepted	The correct resource was used.	Closed
I126468720906519	Progress Meeting Minutes	The agenda, discussions and decisions were recorded.	Closed
I126468720906521	Progress Meeting Minutes	The artifact was continuously updated.	Closed
I126468720906522	Progress Meeting Minutes	The correct resource was used.	Closed
I126468720906523	Progress Report	The artifact was continuously updated.	Closed
I126468720906525	Progress Report	The correct resource was used.	Closed
I126468720906526	Project Budget	The artifact was continuously updated.	Closed
I126468720906527	Project Budget	The correct resource was used.	Closed
I126468720906528	Project Measure	The artifact was continuously updated.	Closed
I126468720906530	Project Measure	The correct resource was used.	Closed
I126468720906531	Project Measure	The measure was used for project monitoring.	Closed
I126468720906532	Project Plan	All plans were integrated.	Closed
I126468720906533	Project Plan	The artifact was continuously updated.	Closed
I126468720906535	Project Plan	The correct resource was used.	Closed
I126468720906536	Project Schedule	The artifact was continuously updated.	Closed
I126468720906537	Project Schedule	The correct resource was used.	Closed
I126468720906539	Project Schedule	The process tailoring was performed.	Closed
I126468720906540	Risk Plan	The artifact was continuously updated.	Closed
I126468720906541	Risk Plan	The correct resource was used.	Closed
I126468720906542	Risk Plan	The risk qualification was performed.	Closed
I126468720906544	Source Code	The artifact was continuously updated.	Closed
I126468720906545	Source Code	The correct resource was used.	Closed
I126468720906546	System Documentation	The artifact was continuously updated.	Closed
I126468720906547	System Documentation	The correct resource was used.	Closed
I126468720906549	Test Errors Report	The artifact was continuously updated.	Closed
I126468720906550	Test Errors Report	The correct resource was used.	Closed
I126468720906551	Test Errors Report	The granularity of open defects was average.	Closed
I126468720906553	Test Errors Report	The issue was properly open and assigned.	Closed
I126468720906554	Traceability Applied	The artifact was continuously updated.	Closed
I126468720906555	Traceability Applied	The correct resource was used.	Closed

Issue View

ID	Issue	Responsible
I126468785198751	It was obtained the client accepted.	[Manager]

I126468797028606	The artifact was not created.	[Manager]
I126468802902608	It was used the correct resource.	[Manager]
I126468748987256	The measures of the project are not updated.	[Manager]
I126468758541025	No measures were taken regarding the tests.	[Manager]

APÊNDICE E - PROJETOS, INSPEÇÕES E RESULTADOS

Legend:			
WP	*Work Product*	IC	*Inspection Coverage*
RI	*Revision Item*	PA	*Process Adherence*

Project	**Inspection**	**WP**	**Without OIP/AIP**			**With OIP/AIP**		
			RI	**IC**	**PA**	**RI**	**IC**	**PA**
Project A	Inspection A1	18	38	2,11	100,00%	53	2,94	100,00%
	Inspection A2	31	52	1,68	92,00%	94	3,03	91,30%
	Inspection A3	25	48	1,92	87,80%	77	3,08	90,28%
	Inspection A4	27	44	1,63	96,47%	70	2,59	90,07%
Project B	Inspection B1	18	38	2,11	73,33%	53	2,94	75,00%
	Inspection B2	31	52	1,68	93,75%	94	3,03	95,00%
	Inspection B3	25	48	1,92	68,18%	77	3,08	74,02%
	Inspection B4	27	44	1,63	72,73%	70	2,59	80,85%
Project C	Inspection C1	18	38	2,11	87,50%	53	2,94	87,23%
	Inspection C2	31	52	1,68	78,26%	94	3,03	82,71%
	Inspection C3	22	48	2,18	90,00%	77	3,50	92,21%
	Inspection C4	24	44	1,83	68,42%	70	2,92	76,92%
Project D	Inspection D1	18	38	2,11	63,16%	53	2,94	70,00%
	Inspection D2	31	52	1,68	87,10%	94	3,03	86,57%
	Inspection D3	25	48	1,92	90,24%	77	3,08	89,47%
	Inspection D4	27	44	1,63	86,21%	70	2,59	76,19%
Project E	Inspection E1	18	38	2,11	90,48%	53	2,94	76,92%
	Inspection E2	31	52	1,68	96,00%	94	3,03	94,82%
	Inspection E3	25	48	1,92	93,55%	77	3,08	91,66%
	Inspection E4	23	44	1,91	90,00%	70	3,04	85,48%
Project F	Inspection F1	18	38	2,11	83,33%	53	2,94	83,01%
	Inspection F2	31	52	1,68	72,73%	94	3,03	77,02%
	Inspection F3	25	48	1,92	68,18%	77	3,08	76,00%
	Inspection F4	26	44	1,69	73,68%	70	2,69	79,36%
Project G	Inspection G1	18	38	2,11	86,67%	53	2,94	86,36%
	Inspection G2	31	52	1,68	89,47%	94	3,03	92,95%
	Inspection G3	25	48	1,92	94,74%	77	3,08	96,97%
	Inspection G4	24	44	1,83	100,00%	70	2,92	100,00%
Project H	Inspection H1	18	38	2,11	83,33%	53	2,94	83,01%
	Inspection H2	31	52	1,68	74,07%	94	3,03	77,17%
	Inspection H3	25	48	1,92	81,82%	77	3,08	82,67%
	Inspection H4	27	44	1,63	75,00%	70	2,59	81,82%
Project I	Inspection I1	18	38	2,11	60,00%	53	2,94	62,50%
	Inspection I2	31	52	1,68	91,30%	94	3,03	89,09%
	Inspection I3	25	48	1,92	82,76%	77	3,08	72,88%
	Inspection I4	27	44	1,63	56,52%	70	2,59	55,55%
Project J	Inspection J1	18	38	2,11	80,00%	53	2,94	79,54%

	Inspection J2	31	52	1,68	88,00%	94	3,03	90,67%
	Inspection J3	25	48	1,92	81,82%	77	3,08	85,33%
	Inspection J4	27	44	1,63	94,12%	70	2,59	93,10%