Revista Eletrônica de Sistemas de Informação
ISSN 1677-3071

V. 12, n. 1

jan-abr 2013

doi:10.5329/RESI.2013.1201

Sumário

ISSN: 1677-3071

Revista hospedada em: http://revistas.facecla.com.br/index.php/reinfo
Forma de avaliação: *double blind review*

Esta revista é (e sempre foi) eletrônica para ajudar a proteger o meio ambiente, mas, caso deseje imprimir esse artigo, saiba que ele foi editorado com uma fonte mais ecológica, a *Eco Sans*,que gasta menos tinta.

UM MODELO BASEADO EM ONTOLOGIA E ORIENTADO A RISCOS PARA CERTIFICAÇÃO DE QUALIDADE DE PRODUTOS DE SOFTWARE

AN ONTOLOGY BASED MODEL FOCUSING ON RISKS FOR QUALITY CERTIFICATION OF SOFTWARE PRODUCTS

(artigo submetido em setembro de 2012)

Lizandra Bays dos Santos
Mestre em Computação Aplicada pela Universidade do Vale do Rio dos Sinos (UNISINOS)
lizandrabays@gmail.com

Sergio Crespo Coelho da Silva Pinto
Professor do Depto. de Ciência da Computação na Univ. Federal de Minas Gerais (UFMG)
crespo.sergio@gmail.com

ABSTRACT

This paper presents a model for software product quality certification based on an ontology that uses a risk oriented framework. A literature review about testing and software quality and international standards such as ISO were used as the grounds for the research that also involved a case study. A prototype was developed using ontologies and software agents. The contribution of this work consists on a framework, which is focused on risks for specialization of software quality model, and ontology for the representation of the involved knowledge.

Key-words: software certification; software quality; software testing; ontology.

RESUMO

Este artigo apresenta um modelo para certificação de qualidade de produtos de software baseado em ontologia que faz uso de um *framework* orientado aos riscos. Para desenvolvimento da pesquisa utilizou-se a base teórica relacionada à área de teste e qualidade de software e os modelos internacionais como normas das séries ISO, além da aplicação em um estudo de caso. Um protótipo foi desenvolvido utilizando-se de tecnologias de ontologias e agentes de software. A contribuição do trabalho consiste no *framework* estar focado em riscos para especialização do modelo de qualidade e em apresentar uma ontologia para representar o conhecimento envolvido no processo de certificação.

Palavras-chave: certificação de software; qualidade de software; teste de software; ontologia.

1 INTRODUÇÃO

A demanda cada vez maior por sistemas computacionais impacta diretamente no seu tamanho e complexidade de modo que, garantir que estes sistemas sejam entregues com o menor número de falhas possível visando reduzir os custos de manutenção, torna-se uma busca constante para produtores de software. No contexto do processo de desenvolvimento de software, a avaliação da qualidade e a certificação do produto se apresentam como importantes ferramentas para se identificar a situação de tal software e viabilizar a tomada de decisões sobre ele.

Ampliando o contexto de análise para a indústria de software como um todo, percebe-se, ainda, que existe uma forte demanda, no mundo inteiro, por padrões e melhores práticas de relacionamento, gestão e operação dos "serviços de TI" que são objeto de terceirização (FERRAZ e PROENÇA, 2007). A certificação de software pode se apresentar como um elemento qualificador importante no mercado internacional (ALEM FILHO, 2007).

Avaliar um produto de software é atribuir certo valor a esse produto, com base em requisitos pré-estabelecidos e sob demanda de um patrocinador (COLOMBO e GERRA, 2008). Um modelo de qualidade, por sua vez, deve conter os requisitos da qualidade que o produto sob avaliação deve possuir.

O objetivo deste artigo é apresentar um modelo para certificação de qualidade de produtos de software baseado em ontologia, o qual, a partir de um *framework* orientado a riscos, permite a especialização do modelo de qualidade até a avaliação e classificação do produto segundo critérios pré-estabelecidos.

O *framework* adotado proporciona o foco nos riscos e requisitos do domínio da aplicação na etapa de especialização do modelo de qualidade. Uma ontologia foi desenvolvida para representar o conhecimento envolvido no *framework*.

A seção 2 traz uma breve descrição de conceitos aplicados no trabalho, visando contextualizar a área temática da pesquisa. A seção 3 traz um destaque para o conceito de riscos no contexto de testes de software. Na seção 4 são apresentados, resumidamente, os principais trabalhos relacionados encontrados na literatura, bem como uma análise comparativa entre eles. Na seção 5 é abordada a metodologia adotada na pesquisa enquanto que a seção 6 apresenta o *framework* para certificação de qualidade de produtos que foi desenvolvido nesta pesquisa. A ontologia do certificado de qualidade, por sua vez, é apresentada na seção 7. O estudo de caso e análise dos seus resultados são descritos na seção 8. Por fim, a seção 9 traz as conclusões e limitações da pesquisa, bem como perspectivas de trabalhos futuros.

2 QUALIDADE E CERTIFICAÇÃO DE SOFTWARE

Segundo o IEEE (*Institute of Electrical and Electronic Engineers*) (IEEE, 1990), uma das definições para garantia da qualidade é que esta é o conjunto de atividades planejadas para avaliar os processos pelos quais os produtos são desenvolvidos ou fabricados. Já o controle da qualidade, na definição daquele instituto, é o conjunto de atividades planejadas para avaliar a qualidade dos produtos desenvolvidos ou fabricados. Desta forma, pode-se concluir que a garantia da qualidade está focada na qualidade do processo de desenvolvimento, enquanto que o controle da qualidade está relacionado à qualidade do produto desenvolvido (IEEE, 1990).

Existem diversos esforços na comunidade científica na busca pela qualidade de software por meio de modelos de referência em qualidade do processo (VERMESAN, 1998). A motivação para este trabalho vem da percepção de que há oportunidades de pesquisa sobre certificação de qualidade dos produtos de software para aplicações comerciais em geral. Dentre os trabalhos pesquisados, a maioria trata da certificação de produtos e componentes de software de segurança crítica (WASSYNG *et al.*, 2011; ALVARO *et al.*, 2007; SOKOLSKY *et al.*, 2011; DIXIT, 2009; HECK *et al.*, 2010; BALCI, 2001; KAUR *et al.*, 2008; VERMESAN, 1998).

A qualidade do processo de desenvolvimento é um meio fundamental para se atingir a qualidade desejada do produto final entregue ao usuário (COLOMBO e GERRA, 2008). Porém, seguir processos normatizados e controlados no desenvolvimento, pode não garantir a inexistência de erros ou a conformidade com requisitos, na versão final do software, devido à falta de provas da qualidade do produto (VERMESAN, 1998). Deste modo, para a emissão de um certificado ou selo de qualidade do produto, são necessárias técnicas de controle de qualidade a fim de aferir o grau de aderência de tal produto a requisitos mínimos, segundo um modelo de qualidade previamente definido (COLOMBO e GERRA, 2008).

Alguns modelos de qualidade, como a ISO/IEC 25010 - *Systems and software engineering - Software product Quality Requirements and Evaluation (SQuaRE) – Quality models for software product quality and system quality in use* (ISO/IEC 25010, 2009) que é uma reestruturação da norma ISO/IEC 9126 (NBR ISO/IEC 9126-1, 2003), se apresentam como referências para processos de avaliação da qualidade de produtos de software. Estes *frameworks* trazem um conjunto de atributos que um software deve apresentar, organizados de forma hierárquica, relacionando subcaracterísticas e métricas que permitem verificar a existência ou não de cada característica. Na Figura 1 é representado o modelo da ISO/IEC 25010 (2009).

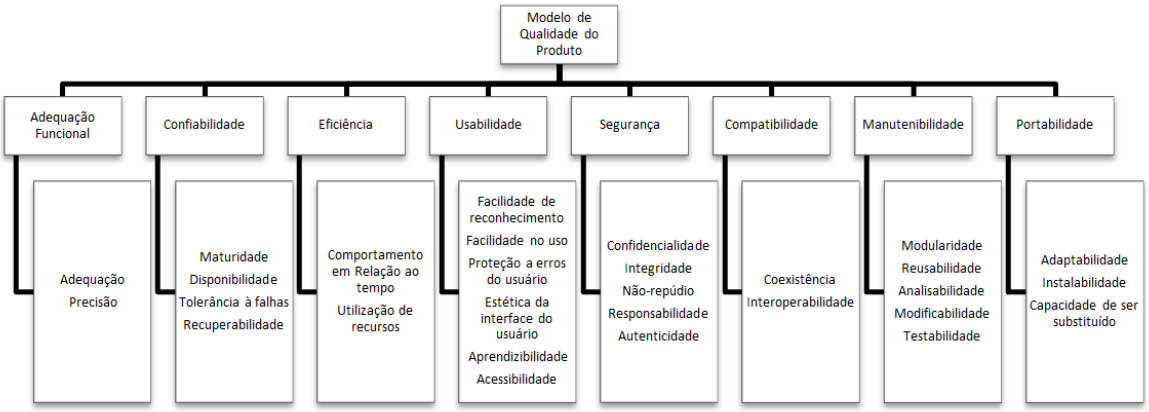

Figura 1 – Modelo de qualidade de produto de software

Fonte: ISO/IEC 25010 (2009)

Esta divisão do conjunto de normas *SQuaRe* aborda atributos de qualidade, mas não trata de como proceder para a avaliação prática de produtos de software a fim de verificar a existência de tais atributos. Um processo de avaliação da qualidade de produtos de software pode ser encontrado na ISO/IEC 25040 - *Evaluation reference model and guide*, a qual é uma releitura da série ISO/IEC 14598 (NBR ISO/IEC 14598-1, 2001), cujo processo é exibido na Figura 2.

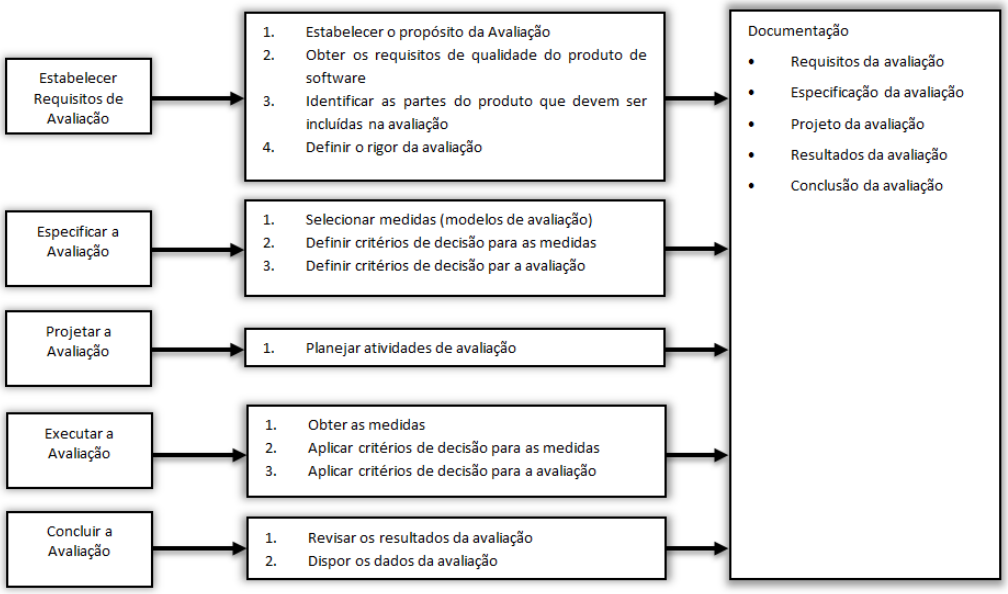

Figura 2 – Modelo de referência de avaliação de qualidade de produto de software

Fonte: ISO/IEC 25040 (2009)

O controle da qualidade é realizado por meio de testes de software. De forma simplificada, pode-se dizer que um software é testado a fim de verificar se sua execução está de acordo com suas especificações. A definição formal mais clássica para teste de software foi introduzida por Myers (1979), no final da década de 70, quando afirmava que teste é o processo de executar um programa ou sistema com a intenção de encon-

trar erros. Passados mais de 30 anos desta definição, observa-se que os objetivos dos testes são bem mais amplos e vão além de encontrar erros. As especificações, que refletem as expectativas do cliente, podem ser tão importantes quanto a existência de erros de codificação. Hetzel (1988), por sua vez, trouxe uma definição que se aproxima dos objetivos relacionados à avaliação de conformidade quando afirmava que teste é uma atividade direcionada para avaliar um atributo ou capacidade de um programa ou sistema e determinar se satisfaz os resultados requeridos.

Os resultados das atividades de testes geram um conjunto de medidas que deve ser utilizado para compor métricas que permitam avaliar se a aplicação sob teste está conforme os requisitos especificados e em qual nível.

A abordagem *Goal Question Metric* (GQM) é amplamente adotada para métricas de software. Definida originalmente por Basili *et al*. (1994) para avaliar defeitos em uma série de projetos do Centro de Voo Espacial da NASA, a técnica baseia-se na suposição de que um processo de medição, para ser efetivo, deve ser focado em objetivos específicos, e sua interpretação deve ser baseada no entendimento desses objetivos (BASILI *et al*., 1994). Trata-se de uma abordagem orientada a objetivos para a medição de produtos e processos de software, que apoia a definição *top-down* do processo de medição e a análise *bottom-up* dos dados resultantes.

O modelo GQM envolve três níveis, conforme a Figura 3. Entre as vantagens observadas no método GQM, destacam-se o apoio na identificação de métricas úteis e relevantes, e a análise e interpretação dos dados coletados.

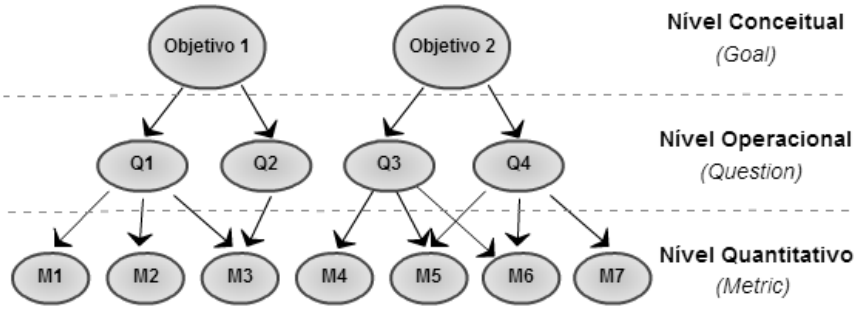

Figura 3 - Estrutura da abordagem GQM

Fonte: adaptado de Basili *et al*. (1994)

No que tange a normatização e certificação na área de Engenharia de Software há a participação da ISO (*International Organization for Standarization*), organização não governamental, cuja missão é promover o desenvolvimento de padrões e atividades relacionadas visando a facilitar a troca internacional de bens e serviços e desenvolver cooperação em atividade intelectual, científica, tecnológica e econômica (SANTOS, 2002). A ABNT (Associação Brasileira de Normas Técnicas) (ABNT, s.d.), por sua vez, é o órgão responsável pela normalização técnica no Brasil, fornecendo a base necessária ao desenvolvimento tecnológico brasileiro. A ABNT é

a única e exclusiva representante no país das entidades internacionais ISO e IEC (*International Electrotechnical Commission*), entre outras.

Certificação de Software é a emissão de um certificado de conformidade de um software a um conjunto de normas ou especificações, comprovada por testes de conformidade e por testes de campo (COLOMBO e GERRA, 2008).

A certificação de qualidade de produto de software tem o foco no produto acabado. Seu objetivo é verificar o grau em que as características de qualidade previamente especificadas estão presentes no produto sob verificação. Com isso, um selo de certificação é emitido ou não. A presença do selo de certificação em um produto de software garante que ele apresenta informação suficiente para um potencial comprador/usuário decidir pela compra ou uso deste produto, além de garantir o funcionamento correto do produto (SANTOS, 2002).

Para se produzir software com qualidade, dentro de prazos e custos controlados, existem alguns modelos de referência que reúnem boas práticas para melhoria do processo de desenvolvimento como o CMMi (*Capability Maturity Model Integration*) e MPS.BR (Melhoria de Processo do Software Brasileiro).

O CMMi é um conjunto de melhores práticas organizadas pelo SEI (*Software Engeneering Institute*), que visam apoiar o desenvolvimento e manutenção de produtos, desde a concepção até a entrega. O modelo baseia-se nas ideias de Watts Humphrey e considera os níveis de capacidade e maturidade da organização (CHRISSIS *et al.*, 2004).

Visando apoiar pequenas e médias empresas brasileiras que não possuem capital para investir no CMMi, surgiu o MPS.BR (SOFTEX, 2007). Este modelo teve como ponto de partida métodos, normas e modelos já definidos e disponíveis no mercado, tais como: a norma ISO/IEC 12207, a norma ISO/IEC 15504 e o modelo CMMi (SOFTEX, 2007). O MPS.BR também é organizado em níveis de maturidade, processos, resultados esperados e atributos dos processos.

3 TESTE DE SOFTWARE SOB UMA PERSPECTIVA DE RISCO

Segundo a base de conhecimento do PMI (*Project Management Institute*) (PMI, 2008), risco é um evento ou condição incerta que, se ocorrer, tem efeito em pelo menos um dos objetivos do projeto.

A base de conhecimento em testes CSTE CBOK (*Certified Software Tester Common Body of Knowledge*) (SOFTWARE CERTIFICATIONS, 2006) define risco como a probabilidade de um evento desfavorável ocorrer resultando em perda. Para esta base de conhecimento, que tem foco em teste de software, o escopo do risco limita-se, então, a eventos negativos, pois preconiza que o resultado implica em perda.

Para Bach (2003), a qualidade do software é considerada boa o suficiente, quando se acredita que os riscos envolvidos na sua utilização são aceitavelmente baixos.

O CSTE CBOK define ainda que controle é tudo o que visa à redução dos riscos, suas ameaças, vulnerabilidades e consequências (SOFTWARE CERTIFICATIONS, 2006).

Neste contexto, o teste de software é considerado um meio de identificar as vulnerabilidades (pontos fracos) de uma aplicação de software, possibilitando gerar controles para reduzir os riscos a níveis aceitáveis (SOFTWARE CERTIFICATIONS, 2006). Deste modo, entende-se que testes de software são atividades de controle de riscos de software.

Pinkster, Burgt e Veenendaal (2006) afirmam que riscos do projeto são aqueles relacionados à gestão dos processos de desenvolvimento e de testes do software. Riscos do produto, por sua vez, são aqueles que afetam diretamente o negócio. Para estes autores a avaliação dos dois tipos de riscos, bem como as medidas de redução desses riscos, deve ser incluída nos devidos planos do projeto.

Vários estudos indicam que testes de software são caros (HUMPHREY, 1999). Além disso, uma das principais limitações do teste é que é impossível testar tudo. Uma alternativa para otimizar os esforços de testes é focá-los nos principais riscos do negócio ao qual a aplicação está associada.

A estratégia RRBT (*Risk & Requirement Based Testing*) é uma técnica de teste que preconiza que seja dado tratamento especial aos testes que cobrem os maiores riscos. Além disso, um inventário de requisitos é considerado. Os requisitos são relacionados aos riscos do produto. Com base nesta informação, são criados os testes que cubram os principais riscos (PINKSTER *et al.*, 2006).

Como os riscos do produto, isoladamente, podem não ser suficientes para uma avaliação baseada em casos de testes, utiliza-se também a especificação de requisitos como base para a realização dos testes. Uma maneira apropriada de relacionar requisitos e riscos é pela análise do funcionamento de determinada funcionalidade e os efeitos de seu não funcionamento (PINKSTER *et al.*, 2006).

Entre as vantagens de testes focados em riscos, além de representarem uma espécie de especialização dos testes para contexto específico da aplicação, está o fato de o esforço acabar sendo aplicado em funcionalidades mais críticas, proporcionando maior valor agregado ao produto.

4 TRABALHOS RELACIONADOS

A pesquisa sobre certificação de software vem sendo praticada aproximadamente desde 1993 (BALCI, 2001). Porém, a maioria dos trabalhos encontrados é focada em certificação do processo de desenvolvimento ou especificamente em certificação de componentes de software de segurança crítica (WASSYNG *et al.*, 2011; ALVARO *et al.*, 2007; SOKOL-

SKY *et al.*, 2011; DIXIT, 2009; HECK *et al.*, 2010; BALCI, 2001; KAUR *et al.*, 2008; VERMESAN, 1998).

Os principais trabalhos relacionados que serviram de inspiração para esta pesquisa são brevemente citados a seguir.

4.1 PROCESSO DE AVALIAÇÃO DE CONFORMIDADE DE PRODUTOS DE SOFTWARE DA RIOSOFT

Dentro do PROIMPE (Programa de Estímulo ao Uso de Tecnologia da Informação em Micro e Pequenas Empresas), a RIOSOFT (2009) lançou um processo de Avaliação de Conformidade de Produtos de Software. Neste modelo a qualidade do produto é avaliada em relação a atributos genéricos de padrões internacionais, como a NBR ISO/IEC 9126, por meio de listas de verificação, além da execução de testes previamente planejados pelo produtor do software. As empresas avaliadas com sucesso têm direito a um certificado de qualidade (RIOSOFT, 2009).

4.2 TESTE OK

Outra instituição brasileira que desenvolveu um selo de qualidade para produtos de software foi a ASSESPRO (Associação das Empresas de Tecnologia da Informação, Software e Internet). Teste OK é um Selo de Certificação de Qualidade de Software com o objetivo de garantir e atestar que o software possui qualidade, sendo aplicado a softwares com desenvolvimento finalizado ou que já estejam em uso no mercado. Neste modelo, vários aspectos genéricos são considerados durante o processo de certificação, entre eles: *performance* ou carga, durabilidade, confiabilidade, recuperação, instalação, acessibilidade, funcionalidade, interface de usuário e usabilidade (ASSESPRO, 2009).

4.3 5CQUALIBR

Uma iniciativa desenvolvida no setor público é o 5CQualiBr. Trata-se de um ambiente dedicado à qualidade de software dento do Portal do Software Público Brasileiro (SPB). O vetor de Qualidade do Produto deste programa apresenta um modelo de qualidade de produto de software para as aplicações disponibilizadas no portal SPB, que são aplicações desenvolvidas na Administração Pública, empresas ou universidades, compartilhadas e distribuídas sob licença de software livre (SPB, 2009).

O modelo é composto por grupos que representam áreas de interesse com objetivo de avaliar a qualidade. Estes grupos são divididos em características (características da qualidade) que são derivadas em atributos e métricas. As métricas, por sua vez, são questões pré-definidas que devem ser respondidas por meio de medições.

4.4 MEDE-PROS®

O MEDE-PROS® é um método de avaliação genérico desenvolvido pelo então CENPRA (Centro de pesquisa Renato Archer), atual CTI/MCT (Centro de Tecnologia da Informação Renato Archer) que visa a avaliar o

software com base no modelo de qualidade explícito nas normas NBR ISO/IEC 9126 e NBR ISO/IEC 12119. Neste método, todos os aspectos do software são avaliados tendo o mesmo valor agregado. O modelo considera um conjunto de elementos que compõem um software e os atributos da qualidade que estão relacionados a estes elementos. Cada atributo é relacionado a um conjunto de questões que compõem as Listas de Verificação (MAITINGUER, 2004).

4.5 MODELO ESPECIALISTA BASEADO EM REQUISITOS ESPECIFICADOS EM EDITAL

Este método é baseado no MEDE-PROS® (COLOMBO e GERRA, 2008), citado anteriormente, e em requisitos especificados em um edital. Um modelo de qualidade foi especializado e aplicado no estudo de caso PNAFM (Programa Nacional de Apoio à Gestão Administrativa e Fiscal dos Municípios Brasileiros), e propõe-se a ser um modelo especialista. Segundo Maintinguer (2004), o modelo seguiu a estrutura de listas de verificação proposta no modelo MEDE-PROS® para a avaliação dos requisitos não funcionais. Para a avaliação dos requisitos funcionais, a proposta se vale de um Guia de Avaliação, o qual é composto, além dos atributos de qualidade relacionados com o requisito da aplicação, de um passo a passo para orientar o avaliador quanto às operações a serem realizadas na utilização do software. A autora destaca ainda, que a sequência de passos deve ser obtida por meio de entrevistas com os possíveis usuários do sistema e que o responsável pela elaboração do guia de avaliação deve estar capacitado na área de domínio da aplicação.

4.6 SPIDER-PQ

Uma ferramenta de apoio à avaliação de qualidade de produtos de software com base no MEDEPROS® (COLOMBO e GUERRA, 2008) foi desenvolvida por Gama e Oliveira (2011). SPIDER-PQ é um sistema de software que auxilia a atividade de preparação, execução e geração de relatório de uma avaliação de qualidade para produtos de software que forem submetidos ao MEDEPROS® (GAMA e OLIVEIRA, 2011). Trata-se de uma aplicação *desktop* que apresenta as funcionalidades de Administração de Usuários, Administração de *Checklist*, Administração de Produto, Administração da Avaliação e Execução da Avaliação, sendo por meio desta última que os avaliadores respondem as questões das listas de verificação (GAMA e OLIVEIRA, 2011).

SPIDER-PQ é integrante do projeto SPIDER (*Software Process Improvement: Development and Research*), um projeto da UFPA (Universidade Federal do Pará) que visa a apresentar um levantamento das ferramentas de software livre com características adequadas para possibilitar a criação de produtos de trabalhos derivados dos resultados esperados descritos nos objetivos dos processos do modelo MPS.BR, e das práticas específicas descritas nos objetivos das áreas de processo do modelo CMMI (SPIDER, 2009).

Até o momento desta pesquisa, a ferramenta SPIDER-PQ ainda estava em fase de homologação em laboratório e não estava publicada no *site* do projeto SPIDER.

4.7 *FRAMEWORK* DE PROCESSO PARA CUSTOMIZAR MODELOS DE QUALIDADE DE SOFTWARE

Pesquisadores sul-africanos (SIBISI e WAVAREN, 2007) propuseram um *framework* de processo que visa à customização de modelos de qualidade por meio do relacionamento dos atributos de modelos tais como NBR ISO/IEC 9126, às necessidades do usuário. A proposta dos autores preconiza a identificação de metas da qualidade. Segundo eles, uma meta de qualidade é a qualidade necessária e suficiente que reflete as necessidades do usuário. Não é, necessariamente, a qualidade perfeita, mas sim, a qualidade que permite que o usuário atinja os seus objetivos. É proposto, ainda, o uso do paradigma GQM (BASILI *et al.*, 1994) para identificação das métricas.

Resumidamente, o *framework* proposto sugere os seguintes passos:

- Passo 1 - Criar um Questionário Genérico de Perfil de Qualidade: consiste em converter as métricas abaixo de cada subcaracterística em questões (perguntas). Se as questões não forem suficientemente claras, utiliza-se o método GQM para refiná-las.
- Passo 2 - Construir um Perfil de Qualidade Específico: consiste em aplicar o questionário para ser respondido por *stakeholders* relevantes, como analistas ou engenheiros de sistemas (para obter a visão do adquirente), ou analistas e engenheiros de software (para obter o ponto de vista do desenvolvedor). Visa a identificar o nível de importância, em percentual, de cada atributo.
- Passo 3 - Construir um Perfil de Qualidade Alvo: consiste em adequar o perfil construído aos objetivos de negócio, por exemplo, superar as expectativas dos clientes mudando a importância de certas características, ou reusar componentes de software, incluindo o reuso como característica relevante.
- Passo 4 - Customizar o Modelo de Qualidade: consiste em utilizar o perfil alvo definido, para selecionar as métricas, desde que cada métrica já possua um nível de importância identificado no modelo.

4.8 MODELO *FUZZY* DE AVALIAÇÃO DE QUALIDADE DE SOFTWARE

Oliveira (2002) implementou uma ferramenta chamada AdeQuaS para avaliação da qualidade de software com base no modelo proposto por Belchior (1997).

O MFAQS (Modelo *Fuzzy* para Avaliação da Qualidade de Software) utiliza a lógica *fuzzy* com a finalidade de facilitar a representação subjetiva da qualidade e a agregação de conceitos. Propõe um processo pelo qual o grau de importância para cada atributo é obtido de especialistas do domí-

nio de aplicação e expresso por meio de termos linguísticos e números *fuzzy* triangulares normais. A combinação dos prognósticos individuais de cada especialista por atributo é feita por meio de interseção e matrizes, e a agregação estabelece o padrão de qualidade, conforme o modelo ISO/IEC 9126, por meio da defuzificação (OLIVEIRA, 2002).

4.9 ANÁLISE DOS TRABALHOS RELACIONADOS

Esta seção apresenta uma análise comparativa entre os trabalhos pesquisados e o modelo que é proposto neste trabalho.

Para a comparação foram elencados critérios os quais julga-se relevantes no contexto e motivação desta proposta, a saber:

- *Critério 1* – Apoia-se em padrões de referência internacionais, como a norma NBR ISO/IEC 9126 ou *SQuaRe*.
- *Critério 2* – O modelo de qualidade é especializado para o domínio da aplicação.
- *Critério 3* – Propõe um processo para especialização do modelo de qualidade.
- *Critério 4* – Prevê a ponderação dos atributos, conforme sua importância para o domínio da aplicação.
- *Critério 5* – A verificação (testes) é orientada pelos riscos de negócio e o domínio da aplicação.
- *Critério 6* – Propõe um sistema para automação do processo de especialização do modelo e avaliação do produto.
- *Critério 7* – Propõe uma arquitetura de sistema baseada em ontologias e agentes de software.
- *Critério 8* – Propõe uma arquitetura de sistema baseada em lógica *fuzzy*.

Os critérios para comparação e análise dos trabalhos relacionados foram definidos a partir de atributos considerados fundamentais, tais como, as referências em que se baseiam, a especialização do modelo de qualidade e características que alguns dos trabalhos pesquisados apresentaram e que os diferenciam dos demais.

O Quadro 1 traz uma consolidação dos critérios que são considerados atendidos para cada um dos trabalhos relacionados.

Observa-se que, apesar de todos apoiarem-se em referências normativas internacionais, para muitos dos trabalhos pesquisados o modelo de qualidade do produto utilizado é genérico. A pesquisa de Maitinguer (2004) apresenta um modelo especializado aos requisitos descritos em um edital de licitação. Apesar de a autora descrever o processo adotado para especializar o modelo, tal processo não se aplica à especialização de modelos cuja base de requisitos não seja um edital. Outro aspecto na pesquisa de Maitinguer (2004) é que os requisitos funcionais são todos tratados dentro do atributo Funcionalidade, e os requisitos não funcionais, que também podem apresentar subcaracterísticas e níveis de importância

diferenciados para diferentes domínios de aplicação, são baseados no método genérico MEDE-PROS®.

Trabalhos relacionados	Critérios de comparação							
	1	2	3	4	5	6	7	8
Modelo da RIOSOFT (2009)	x							
TesteOK! (ASSESPRO, 2009)	x							
5CQualiBr (SPB, 2009)	x							
MEDEPROS® (COLOMBO e GUERRA, 2008)	x							
SPIER-PQ (GAMA e OLIVEIRA, 2011)	x					x		
Modelo de Maitinguer (2004)	x	x	x					
Sistema AdeQuaS (OLIVEIRA, 2002)	x	x	x	x		x		x
Modelo de Sibisi e Waveren (2007)	x	x	x	x				
Sistema de Apoio à Certificação de Qualidade de Produtos de Software	x	x	x	x	x	x	x	

Quadro 1 – Comparativo dos trabalhos relacionados

Fonte: elaborado pelos autores com base na compilação da literatura

A ferramenta SPIDER-PQ, que visa a apoiar avaliações com MEDE-PROS®, se propõe a automatizar o processo de preparação e avaliação, porém se baseia em um modelo de dados relacional e não faz uso de ontologias para representar o conhecimento.

O diferencial do trabalho de Sibisi e Waveren (2007) é apresentar um processo para especialização do modelo de qualidade que considera os níveis de importância de cada um dos atributos para o domínio da aplicação e que é baseado em metas de negócio.

O sistema AdeQuaS (OLIVEIRA, 2002), foi o primeiro trabalho que se propôs a apresentar um modelo especialista. Baseia-se no MFAQS (BELCHIOR, 1997) o qual se utiliza de lógica *fuzzy* para representar o grau de importância dos atributos e de cada avaliador. Esta abordagem se mostra interessante para representar a subjetividade envolvida no processo de avaliação.

Como se pode observar, nenhum dos trabalhos pesquisados tem foco nos riscos do domínio da aplicação, o que se apresenta como um dos diferencias da proposta deste trabalho. Além disso, outro ponto de distinção desta pesquisa é a proposição de um sistema para automação do processo baseado em ontologias e agentes de software.

5 METODOLOGIA

Para o atendimento dos objetivos propostos, a metodologia aplicada iniciou com uma investigação do estado da arte em qualidade e certificação de software, além de trabalhos relacionados, conceitos e tecnologias envolvidas na proposta.

Para os trabalhos relacionados, foram realizadas buscas em bases de trabalhos científicos pelo termo "*software product*", associado às palavras-chave "*certification*", "*framework*" ou "*testing*". Foram avaliados 72 trabalhos, estrangeiros e nacionais.

A etapa seguinte foi o estabelecimento de um modelo que permitisse a rastreabilidade desde o risco do software até a técnica de teste apropriada para aferir os atributos relacionados. Em seguida, foram estabelecidos critérios de julgamento e classificação da qualidade do produto com base na ponderação dos atributos da qualidade. Este modelo culminou no *framework* apresentado na próxima seção.

A validação do modelo deu-se por meio de um estudo de caso.

As etapas seguintes consistiram em desenvolver e testar a ontologia que representa o conhecimento bem o projeto.

Para validação da ontologia foi desenvolvido um protótipo no qual foi aplicada a mesma massa de dados utilizada no estudo de caso, possibilitando a validação do *framework.*

6 *FRAMEWORK* PARA CERTIFICAÇÃO DE QUALIDADE DE PRODUTOS DE SOFTWARE

Esta seção apresenta a proposta de um *framework* para certificação de qualidade de produtos de software, o qual preconiza a seleção dos requisitos com base em riscos para a especialização do modelo de qualidade. O modelo é decomposto em três subprocessos, conforme a Figura 4, a seguir.

Figura 4 - *Framework* para certificação de qualidade de produtos de software

Fonte: elaborada pelos autores

O subprocesso de especialização do modelo de qualidade visa apoiar a definição dos atributos da qualidade específicos para o domínio da aplicação, baseado nos riscos e requisitos do produto. O subprocesso de medição da qualidade contempla as métricas e técnicas para mensuração das evidências dos atributos da qualidade. Já o subprocesso de avaliação da qualidade, por sua vez, deve contemplar uma sistemática de pontuação e critérios de julgamento para as medidas coletadas, de modo que o produto possa ser certificado e comparado com outros.

As etapas dos subprocessos de especialização do modelo, medição e avaliação da qualidade são apresentadas na Figura 5.

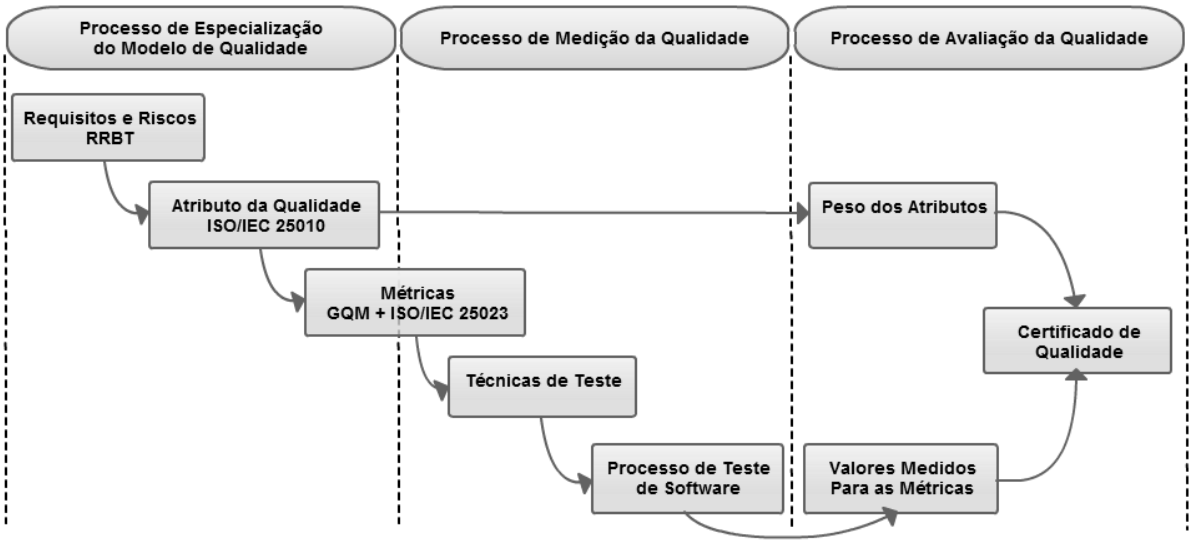

Figura 5 – Subprocessos de especialização do modelo, de medição e de avaliação da qualidade

Fonte: elaborada pelos autores

O subprocesso de especialização do modelo preconiza uma espécie de rastreabilidade desde os requisitos da aplicação até a métrica de software, por meio dos seguintes passos:

- *Passo 1*: Associar riscos de negócio aos requisitos da aplicação;
- *Passo 2*: Associar atributos e subcaracterísticas da qualidade aos riscos;
- *Passo 3*: Estabelecer pesos (grau de importância) para cada atributo da qualidade;
- *Passo 4*: Identificar métricas para cada atributo.

O passo 3 preconiza que a cada atributo de qualidade seja atribuída uma nota ou peso conforme seu grau de importância no contexto do modelo de qualidade que está sendo especializado. Naturalmente, aos maiores riscos de negócio devem ser atribuídos os maiores pesos. Esta ponderação é essencial para o subprocesso de avaliação da qualidade pelo qual será calculado o grau de conformidade ou selo de qualidade do produto.

De posse das métricas inicia-se o subprocesso de medição da qualidade pelo qual os testes devem ser realizados a fim de coletar as medidas. Para esta etapa são previstos dois passos:

- *Passo 5*: Associar técnicas de testes a cada métrica identificada;
- *Passo 6*: Executar testes para obter as medidas.

No passo 5, a cada métrica deve-se relacionar a técnica de testes que permitirá coletar os dados necessários para medição. A medida é o objetivo e a técnica de teste é o meio pelo qual este objetivo será aferido. Neste momento, já se tem subsídios para dar entrada em um processo de teste de software.

No passo 6, as atividades de testes devem ser planejadas e executadas. Vale destacar que o objetivo do teste, neste contexto, não é detectar erros na aplicação, mas sim, coletar as medidas para cada uma das métricas estabelecidas com intuito de avaliar a conformidade do produto aos atributos da qualidade que estas métricas representam.

Para o subprocesso de avaliação da qualidade, é necessário estabelecer critérios de julgamento e classificação das medidas identificadas e coletadas nas etapas anteriores, por meio dos seguintes passos:

- *Passo 7*: Calcular a aderência das medidas obtidas aos valores de referência para todas as métricas;
- *Passo 8*: Calcular o valor do atributo, conforme valor de aderência de suas métricas;
- *Passo 9*: Calcular o grau de conformidade do produto conforme os pesos dos atributos.

No passo 7, as medidas obtidas devem ser comparadas com valores de referência para cada uma das métricas. O coeficiente de aderência ao valor de referência deve ser calculado para cada métrica, de modo que represente o quanto o valor medido é aderente ao valor de referência. Porém, valores de referência podem apresentar diferentes domínios, dependendo do tipo de unidade de medida, grandeza de valores e tipo de métrica. Deste modo, para o coeficiente de aderência, é necessário lançar mão de algumas variações na fórmula (1a, 1b e 1c), conforme abaixo:

$MA = (MV \,/\, RV)$, se RV = 1, quanto maior e mais próximo de 1 melhor (1a)

$MA = (1 - MV)$, se RV = 0, quanto menor e mais próximo de 0 melhor (1b)

$MA = (1 - (MV/RV))$, se RV > 0, quanto menor e mais próximo de RV melhor (1c)

Onde,

- MA - *Metric Adherence Coefficient*: coeficiente de aderência do valor medido para a métrica em relação ao valor de referência;
- MV - *Measured Value*: valor medido para a métrica por meio dos testes de software;
- RV - *Reference Value*: valor de referência para uma métrica, obtido na ISO/IEC 25023.

Com as três variações da fórmula de cálculo de MA, descritas acima, é possível identificar a aderência dos valores medidos de cada métrica, em uma escala de valores que varia de 0 a 1, sendo que 1 representa aderência total. O valor final de cada atributo deve ser totalizado no Passo 8 por meio da soma ponderada dos valores de aderência de todas as métricas que compõem o atributo, conforme fórmula abaixo:

$$\mathrm{AV} = \sum_{i=1}^{n} (\mathrm{MA} * \mathrm{MW})\ \mathrm{MA} = (\mathrm{MV} \,/\, \mathrm{RV}) \quad (2)$$

Onde,

- AV - *Attribute Determined Value*: Valor determinado do atributo, obtido por meio de cálculo do somatório dos valores medidos das métricas multiplicados pelos pesos proporcionais de cada uma;
- MA - *Metric Adherence Coefficient*: coeficiente de aderência da métrica calculado no passo anterior;
- MW - *Metric Proporcional Weight*: peso proporcional da métrica no atributo, considerando o conjunto de métricas que serão utilizadas para avaliar um atributo da qualidade. MW = (1/n);
- n – número de métricas do atributo.

O grau de conformidade do produto é calculado no passo 9 por meio do somatório dos valores calculados para cada atributo multiplicados pelos respectivos pesos. O resultado deve variar entre 0 e 1, sendo que o valor ideal é 1, representando 100% de conformidade. A fórmula de cálculo é a seguinte:

$$QL = \sum_{i=1}^{m}(AV * AW)\ \mathrm{AV} = \sum_{\mathrm{i}=1}^{\mathrm{n}}(\mathrm{MA}^{*}\mathrm{MW})\ \mathrm{MA} = (\mathrm{MV}\ /\ \mathrm{RV}) \qquad (3)$$

Onde,

- QL - *Quality Level*: nível de qualidade do produto;
- AV - *Attribute Determined Value*: Valor determinado do atributo obtido no passo anterior;
- AW - *Attribute Relative Weight*: peso relativo do atributo de qualidade no contexto do modelo de qualidade especializado. O peso deve ser determinado com base nos riscos;
- m - número de atributos selecionados e ponderados.

7 ONTOLOGIA DO CERTIFICADO DE QUALIDADE

A ontologia do certificado de qualidade visa a mapear os conceitos e relacionamentos entre eles, de modo a representar o conhecimento que envolve a especialização do modelo, a medição e a avaliação de qualidade de software.

O termo ontologia tem origem na Filosofia em que trata do conhecimento do ser. Na Ciência da Computação, ontologias visam a descrever os tipos de entidades existentes em um determinado domínio do conhecimento e como elas são relacionadas (GRUBER, 2010 *apud* SILVA, 2010).

A *Web Ontology Language* (OWL) é uma recomendação do W3C (*Word Wide Web Consortium*) que objetiva prover uma representação eficiente para ontologias (W3C, 2004). É uma linguagem utilizada para descrever um determinado domínio de conhecimento por meio de estruturas de representação para classes, propriedades, restrições e indivíduos.

Para o desenvolvimento da ontologia do certificado de qualidade foi adotada a metodologia proposta por McGuinness e Noy (2001) devido a sua simplicidade, por preconizar o reuso de ontologias e um processo iterativo.

Os primeiros conceitos identificados foram os relativos ao Modelo de Qualidade, os quais devem permitir mapear os relacionamentos entre requisitos, riscos, atributos e subcaracterísticas da qualidade. Em seguida foram identificados os conceitos relativos ao processo de medição e avaliação da qualidade, que representam o conhecimento envolvido na certificação do produto de software.

Na busca por reuso de ontologias, algumas ontologias relacionadas com o domínio especificado foram encontradas da literatura. A ontologia de qualidade proposta por Duarte e Falbo (2000) não apresentou termos que se adequassem aos conceitos de riscos e requisitos. A proposta de Gusmão *et al.* (2004) é uma ontologia de riscos para desenvolvimento de software apoiada na taxonomia de riscos do SEI (CARR, 1993 *apud* GUSMÃO *et al.*, 2004), porém é fortemente voltada para a identificação e análise de riscos, atividades que estão fora do escopo deste trabalho.

Uma ontologia para medição de software foi proposta por Ferreira *et al.* (2006), que é apresentada na Figura 6.

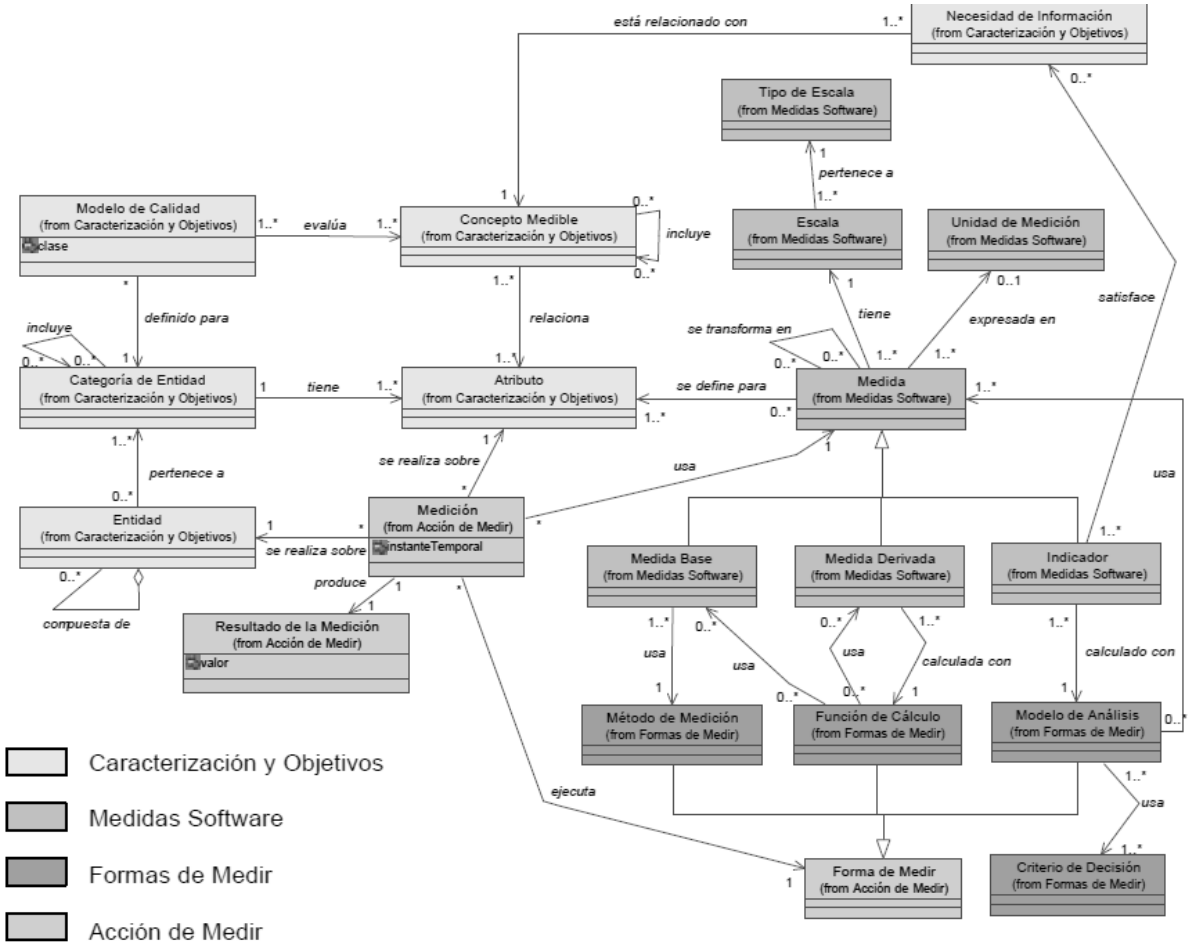

Figura 6 - Ontologia para medição de software

Fonte: Ferreira *et al.*(2006)

A ontologia de medição de software apresentada pelos pesquisadores espanhóis mostrou-se adequada para representar os termos do *framework* para certificação de qualidade de produtos de software, principalmente porque integra as medidas de software com um modelo de qualidade e as necessidades de informação que conduzem o processo de medição. Desta forma, é possível representar o conhecimento envolvido na especialização do modelo de qualidade. Além disso, a proposta de Ferreira *et al*. (2006) trabalha, além dos sinônimos e homônimos, nas lacunas e conflitos encontrados na investigação sobre as principais normas de medição de software, demonstrando um alinhamento com os esforços da ISO/IEC e do IEEE, os quais vêm buscando, desde 2002, harmonizar as terminologias do campo da medição, especialmente o ISO-JTC1-SC7, que trabalha na série de normas SQuaRe (FERREIRA *et al*., 2006).

A ontologia do certificado de qualidade foi, então, elaborada a partir das subontologias de "caracterização e objetivos de medição" e da "forma de medir", propostas por Ferreira *et al*. (2006). Os termos previstos no *framework* para certificação de qualidade de produtos de software foram relacionados aos seus correspondentes na ontologia de medição de software. Porém, como o *framework* preconiza que a especialização do modelo seja apoiada pela abordagem GQM fazendo uso das questões que descrevem as métricas, foi identificada a necessidade de incluir mais duas classes na ontologia, de modo que fossem representados os conceitos de Subcaracterística da qualidade (que compõe um Atributo) e de Questão (que é respondida por uma Medida).

A Figura 7 apresenta a versão final da ontologia do certificado de qualidade, que é uma adaptação da ontologia de medição de software proposta por Ferreira *et al*. (2006).

A ontologia do certificado de qualidade foi modelada com apoio da ferramenta Protegé (2013), pela qual foi possível validar os conceitos e testar diversas consultas. Para manipular o modelo OWL foram implementadas funções de manipulação e consulta por meio do *framework* Jena, fazendo uso das bibliotecas SPARQL (W3C, 2008).

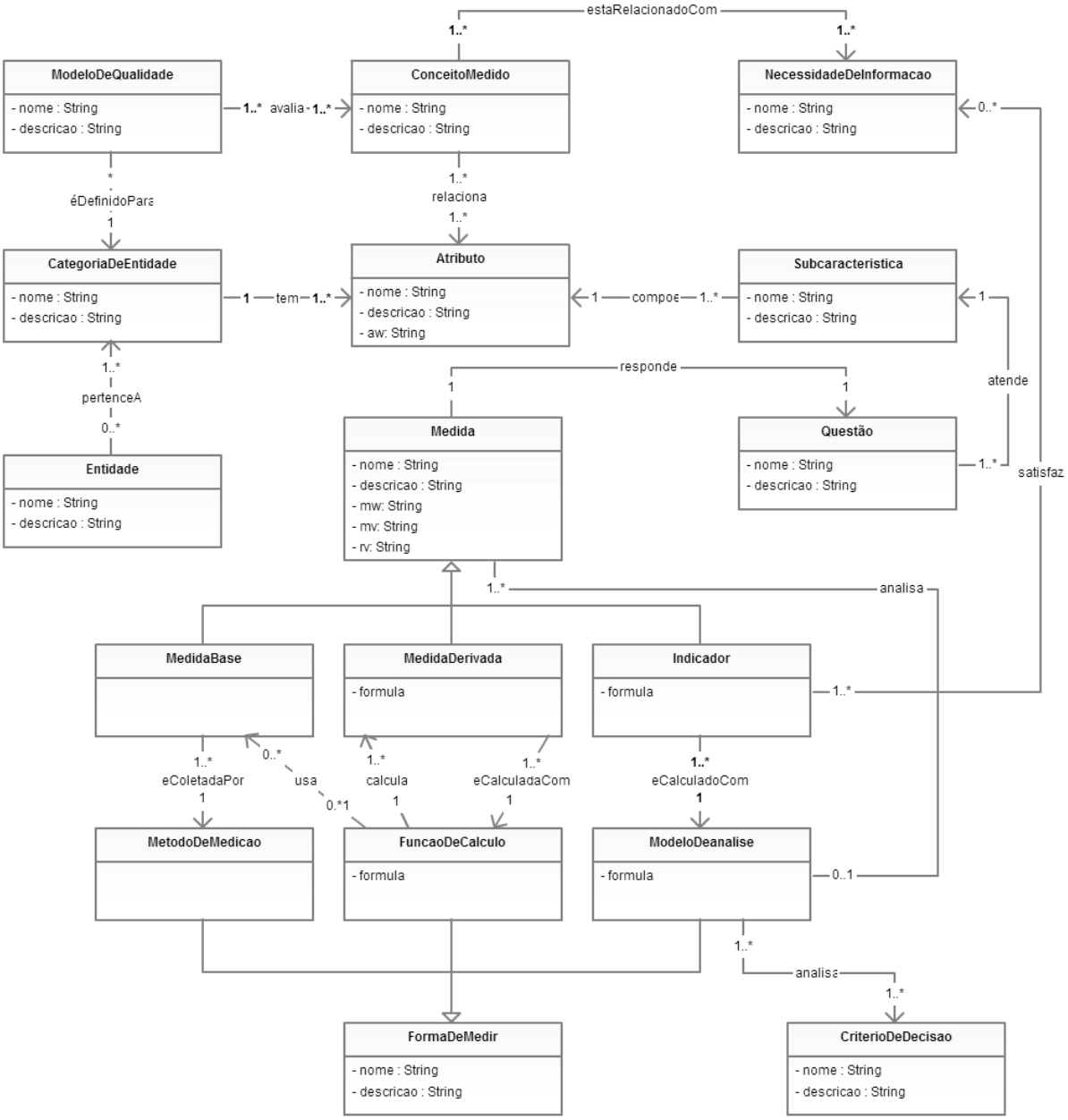

Figura 7 - Ontologia do Certificado de Qualidade

Fonte: adaptada de Ferreira *et al*. (2006)

8 ESTUDO DE CASO

Como este trabalho propõe um *framework* para certificação de qualidade e uma ontologia que representa o conhecimento visando a automatizar o processo proposto, uma validação manual do *framework*, antes do desenvolvimento da ontologia, foi fundamental para a confirmação dos conceitos, das fórmulas de cálculo e para o refinamento da ontologia do certificado de qualidade. Deste modo, a estratégia de validação se dividiu em duas etapas: a validação do *framework* e a validação da ontologia.

8.1 VALIDAÇÃO DO *FRAMEWORK*

Para a validação do *framework* de certificação de qualidade de produtos de software, foi realizado um estudo de caso em um sistema real. Todos os passos dos subprocessos foram realizados manualmente, com apoio de planilhas eletrônicas. A metodologia foi aplicada para um sistema que dá suporte a fiscalizações de tributos federais, o qual foi testado, homologado e implantado em produção. Trata-se de um sistema que foi construído em Java para plataforma *desktop*, que tem uma abrangência de cerca de três mil usuários.

Entre as principais características do sistema estudado, destacam-se a comunicação com outros sistemas e a capacidade de configuração de formas de cálculo e relatórios de saída, o que fornece autonomia ao usuário diante de mudanças na legislação tributária.

No Quadro 2 é apresentada a consolidação dos passos 1 a 3 desta proposta realizados no estudo de caso.

Passo 1: RRBT		*Passo 2*: ISO/IEC 25010		*Passo 3*: Pesos
Requisito	**Risco**	**Característica**	**Subcaracterística**	**Importância**
Enviar e receber nova versão da base centralizada. Enviar e receber informações dos sistemas relacionados.	**R001** - Indisponibilidade do sistema para o usuário.	Confiabilidade	Tolerância a falhas	3
			Maturidade	
			Disponibilidade	
			Recuperabilidade	
	R002 - Insuficiência dos recursos envolvidos com a produção do sistema, causando indisponibilidade.	Eficiência	Utilização de recursos	2
	R003 – Interceptação de informações sigilosas no tráfego de rede utilizado pelo sistema.	Segurança	Integridade	1
	R004 – Acesso liberado, aos usuários dessa aplicação, às informações que ficam inseridas no banco local instalado na estação de trabalho do usuário.		Responsabilidade	
			Confidencialidade	

Quadro 2 - Aplicação dos passos 1 a 3 no estudo de caso

Fonte: elaborado pelos autores

Foram selecionados requisitos relevantes e identificados os riscos associados com apoio da documentação do sistema e informações obtidas com a equipe de desenvolvimento. Os requisitos abordados neste estudo de caso referem-se a funcionalidades de recepção e envio de dados a bases centralizadas e os riscos identificados são relativos à confiabilidade, eficiência e segurança. A identificação dos riscos foi realizada por uma equipe multidisciplinar de especialistas das áreas de infraestrutura, negócios, suporte, segurança e pessoal de desenvolvimento, conforme preconiza o processo de desenvolvimento da empresa produtora do software.

A aplicação do passo 4, no qual são definidos os objetivos e métricas, é apresentada nos Quadros 3 a 7 para cada um dos riscos da aplicação avaliada no estudo de caso, juntamente com a aplicação do passo 5, pelo qual são associadas técnicas de testes às métricas identificadas.

***Passo 4*: GQM**		
R001 - Confiabilidade / Tolerância a Falhas e Maturidade		
Objetivo		
Propósito: avaliar **Questão**: a capacidade de prevenção de falhas **Objeto**: do sistema **Ponto de vista**: conforme usuário		
Questão	**Métrica**	***Passo 5*: Técnica de teste**
Qual o percentual de falhas detectadas que foram corrigidas?	M1: número de falhas corrigidas / número de falhas detectadas	Teste de longa duração Testes de indução de falhas Teste baseados na especificação
Quantas vezes o produto de software causa a queda de todo o ambiente de produção?	M2: 1 - número de quedas do sistema / número de falhas no sistema	Teste de longa duração Testes de indução de falhas
Quantos padrões de faltas são mantidos sob controle para evitar falhas criticas e sérias?	M3: número de ocorrências de falhas sérias e críticas evitadas conforme os casos de testes de indução de falhas / número de casos de testes de indução de falhas executados	Teste de longa duração Testes de indução de falhas
R001 - Confiabilidade / Disponibilidade e Recuperabilidade		
Objetivo		
Propósito: avaliar **Questão**: a disponibilidade **Objeto**: do sistema **Ponto de vista**: conforme usuário		
Questão	**Métrica**	***Passo 5*: Técnica de teste**
Quão disponível é o sistema para uso durante um período de tempo específico?	M4: { tempo de operação / (tempo de operação + tempo de reparo) } M5: total de casos em que o sistema estava disponível e foi utilizado com sucesso pelo usuário / número total de casos em que o usuário tentou usar o software durante um período de tempo	Teste de longa duração Testes de indução de falhas
Qual é o tempo médio em que o sistema fica indisponível quando uma falha ocorre, antes da inicialização?	M6: tempo ocioso total (indisponível) / número de quedas do sistema	Teste de longa duração Testes de indução de falhas
Qual o tempo médio que o sistema leva para completar a recuperação desde o início?	M7: soma de todos os tempos de recuperação do sistema inativo em cada oportunidade / número total de casos em que o sistema entrou em recuperação	Teste de longa duração Testes de indução de falhas

Quadro 3 – Aplicação dos passos 4 e 5 para o Risco R001

Fonte: elaborado pelos autores

Passo 4: GQM		
R002 – Eficiência / Utilização de Recursos		
Objetivo		
Propósito: avaliar **Questão**: a eficiência na utilização de recursos **Objeto**: de produção **Ponto de vista**: conforme usuário		
Questão	**Métrica**	***Passo 5*: Técnica de teste**
Qual é o limite absoluto de transmissões necessárias para cumprir uma função?	M8: número máximo de mensagens de erro e falhas relacionadas à transmissão do primeiro ao último item avaliado / máximo requerido de mensagens de erro e falhas relacionadas à transmissão	Teste de *stress* Valores limites de usuários simultâneos Valores limites de dados trafegados (simular carga máxima)
O sistema é capaz de desempenhar tarefas dentro da capacidade de transmissão esperada?	M9: capacidade de transmissão / capacidade de transmissão específica projetada para ser usada pelo software durante sua execução	Teste de *stress* Valores limites de usuários simultâneos Valores limites de dados trafegados (simular carga máxima)

Quadro 4 - Aplicação dos passos 4 e 5 para o Risco R002

Fonte: elaborado pelos autores

Passo 4: GQM		
R003 – Segurança / Integridade		
Objetivo		
Propósito: avaliar **Questão**: a integridade dos dados **Objeto**: do sistema **Ponto de vista**: conforme usuário		
Questão	**Métrica**	***Passo 5*: Técnica de teste**
Qual é a frequência de eventos de corrupção de dados?	M10: 1 – (número de vezes que o maior evento de corrupção de dados ocorreu / número de casos de testes executados que causaram eventos de corrupção de dados) M11: 1 - (número de vezes que o menor evento de corrupção de dados ocorreu / número de casos de testes executados que causaram eventos de corrupção de dados)	Testes de penetração Simulação de padrões de ataques às vulnerabilidades da conexão

Quadro 5 - Aplicação dos passos 4 e 5 para o Risco R003

Fonte: elaborado pelos autores

Passo 4: GQM		
R004 – Segurança / Responsabilidade e Confidencialidade		
Objetivo		
Propósito: avaliar **Questão**: a segurança de acesso **Objeto**: ao sistema **Ponto de vista**: conforme usuário		
Questão	**Métrica**	*Passo 5*: Técnica de teste
Quão completa é a trilha de auditoria sobre o acesso do usuário ao sistema e dados?	M12: número de acessos do usuário ao sistema e dados gravados no *log* de acesso / número de acessos do usuário ao sistema e dados realizados durante a avaliação	Testes de penetração; Simulação de padrões de ataques às vulnerabilidades da conexão
Quão controlável é o acesso ao sistema?	M13: número (tipos diferentes) de operações ilegais detectadas / número (tipos diferentes) de operações ilegais especificadas	Testes de penetração Simulação de padrões de ataques às vulnerabilidades da conexão

Quadro 6 - Aplicação dos passos 4 e 5 para o Risco R004

Fonte: elaborado pelos autores

O passo 6, que prevê a execução dos testes para coleta das medidas, não foi realizado especificamente para este estudo de caso por questões contratuais estabelecidas pelo cliente do software. Os testes foram realizados em um centro de testes, conforme preconiza o processo de desenvolvimento da empresa produtora do software e os resultados foram cedidos para esta pesquisa.

Para validar o subprocesso de avaliação da qualidade, foram realizados os passos 7 a 9 a partir dos resultados das atividades de testes, conforme mostra o Quadro 7.

Passo 7					*Passo 8*					*Passo 9*
Atributo	**Métrica**	**MV**	**RV**	**MA**	**MW**	**n**	**AW**	**AV**	**m**	**QL**
Confiabilidade	M1	1	1	1,0000	0,1429	7	0,75	0,9082	2	0,9311
	M2	1	1	1,0000						
	M3	1	1	1,0000						
	M4	1	1	1,0000						
	M5	1	1	1,0000						
	M6	120	60	0,5000						
	M7	35	30	0,8571						
Segurança	M10	0	0	1,0000	0,5000	2	0,2500	1,0000		
	M12	1	1	1,0000						

Quadro 7 - Aplicação dos passos 7 a 9 para o estudo de caso

Fonte: elaborado pelos autores

O nível de qualidade (QL) do sistema avaliado no estudo de caso ficou em 0,9311, representando 93,11% do valor ideal que seria 1,000.

Vale destacar, neste momento, que vários experimentos foram realizados para refinar a fórmula de cálculo de aderência das métricas (MA), especialmente para aquela com unidades de medida de tempo. Foi observado que, para diferentes unidades de medida e diferentes grandezas de valores, é necessário tratar de forma diferente a comparação com valor de referência. Para as métricas com unidades de medida de tempo, foram constatadas duas particularidades na recomendação da norma SQuaRe 25023 (ISO/IEC 25023, 2011), da mesma forma que na versão anterior ISO/IEC 9126-2 (2003):

- o valor de referência é “quanto menor e mais próximo de zero melhor”, porém, estes resultados raramente terão valor medido igual a zero.
- não há uma definição quanto à unidade de tempo, se hora, minuto ou segundo, o que pode afetar o valor final calculado.

Deste modo, para as métricas que envolvem medidas de tempo, no estudo de caso M6 e M7, foram definidos valores de referência com base na especificação de requisitos e fixada a unidade de medida em segundos.

Para a validação do sistema desenvolvido foi adotada uma estratégia pela qual foram realizadas mais algumas baterias de testes de sistema em ambiente de desenvolvimento, desta vez, seguindo todos os passos do *framework*, com a mesma amostra de dados do sistema adotado no estudo de caso e pela mesma pesquisadora que executou a validação manual. Foram realizadas todas as associações entre os elementos, conforme os passos de 1 a 5 do *framework*, por meio das interfaces do protótipo. Em seguida, os cálculos foram submetidos ao sistema. Os resultados obtidos foram conforme esperado.

8.2 VALIDAÇÃO DA ONTOLOGIA

Nesta esta etapa foi desenvolvido um protótipo para popular e manipular a ontologia. Para o protótipo desenvolvido foi adotada a arquitetura apresentada na Figura 8.

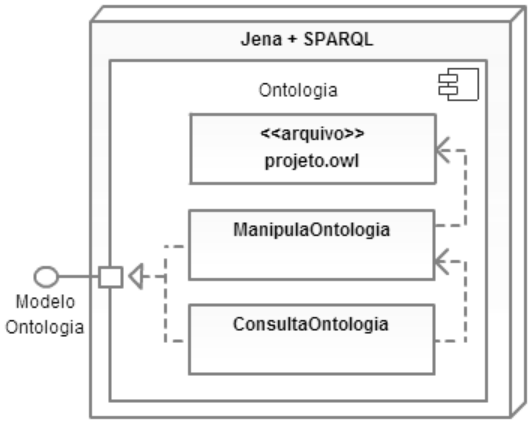

Figura 8 – Arquitetura para o protótipo ontologia do certificado de qualidade

Fonte: elaborada pelos autores

Conforme apresentado na Figura 8, a estrutura Ontologia encapsula as funcionalidades de manipulação e consulta na ontologia. A classe ManipulaOntologia é responsável por criar e manter o modelo da ontologia (arquivo OWL) e apresenta uma interface que permite manipular seus indivíduos. Já a classe ConsultaOntologia traz uma interface que permite realizar as consultas SPARQL.

Para validação da ontologia foram realizados todos os passos do *framework*, para a mesma amostra de dados adotada no estudo de caso, fazendo uso do protótipo.

Os resultados esperados eram realizar todas as associações entre elementos, conforme passos 1 a 5 do *framework* e encontrar os mesmos valores calculados no estudo de caso para todas as fórmulas, conforme passos 7 a 9 do *framework*.

Foram cadastrados o projeto, os riscos e os requisitos pela interface do sistema e as associações foram plenamente realizadas na aplicação.

Como o protótipo não contemplou a emissão de relatórios, para fins de evidências dos testes para esta pesquisa, foram extraídos resultados por meio de consultas SPAQL na ontologia.

A Figura 9 apresenta uma consulta SPARQL que responde a questão “quais são os Atributos relacionados a cada Risco e as Subcaracterísticas que os compõem?”. Neste caso são recuperados os indivíduos das classes NecessidadeDeInformação, ConceitoMedido, Atributo e Subcaracterística onde ConceitoMedido ‘relaciona’ Atributo e ‘estaRelacionadoCom’ NecessidadeDeInformacao e Subcaracterística ‘compoe’ Atributo.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-Schema#>
PREFIX ont: <http://www.semanticweb.org/ontologies/2012/9/Ontology1351443552343.owl#>
SELECT ?Necessidade_nome ?Conceito_nome
       ?Atributo_nome ?Subcaracteristica_nome
 WHERE { ?ConceitoMedido ont:relaciona ?Atributo .
         ?ConceitoMedido ont:estaRelacionadoCom ?NecessidadeDeInformacao .
         ?Subcaracteristica ont:compoe ?Atributo .
         ?ConceitoMedido ont:nome ?Conceito_nome .
         ?Atributo ont:nome ?Atributo_nome .
         ?NecessidadeDeInformacao ont:nome ?Necessidade_nome .
         ?Subcaracteristica ont:nome ?Subcaracteristica_nome }
ORDER BY ?Necessidade_nome ?Conceito_nome
```

Necessidade_nome	Conceito_nome	Atributo_nome	Subcaracteristica_nome
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Disponibilidade"
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Maturidade"
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Recuperabilidade"
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Tolerancia_a_falhas"
"REQUISITO01"	"RISCO02"	"Eficiencia"	"Utilizacao_de_recursos"
"REQUISITO01"	"RISCO03"	"Seguranca"	"Confidencialidade"
"REQUISITO01"	"RISCO03"	"Seguranca"	"Integridade"
"REQUISITO01"	"RISCO03"	"Seguranca"	"Responsabilidade"
"REQUISITO01"	"RISCO04"	"Seguranca"	"Confidencialidade"
"REQUISITO01"	"RISCO04"	"Seguranca"	"Integridade"
"REQUISITO01"	"RISCO04"	"Seguranca"	"Responsabilidade"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Disponibilidade"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Maturidade"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Recuperabilidade"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Tolerancia_a_falhas"
"REQUISITO02"	"RISCO02"	"Eficiencia"	"Utilizacao_de_recursos"
"REQUISITO02"	"RISCO03"	"Seguranca"	"Confidencialidade"
"REQUISITO02"	"RISCO03"	"Seguranca"	"Integridade"
"REQUISITO02"	"RISCO03"	"Seguranca"	"Responsabilidade"
"REQUISITO02"	"RISCO04"	"Seguranca"	"Confidencialidade"
"REQUISITO02"	"RISCO04"	"Seguranca"	"Integridade"
"REQUISITO02"	"RISCO04"	"Seguranca"	"Responsabilidade"

Figura 9 - Consulta que retorna Riscos, Atributos e Subcaracterísticas

Fonte: elaborada pelos autores

A Figura 10 traz o resultado da consulta SPARQL que mostra os valores calculados para as métricas, os atributos e o nível de qualidade do produto.

Atributo_nome	Medida_nome	Medida_MV	Medida_RV	Medida_MA	Atributo_AW	Atributo_AV	Indicador_resultado
"Confiabilidade"	"Remocao de falhas"	"1"	"1"	"1"	"0.75"	"0.7653"	"0,9311"
"Confiabilidade"	"Prevenção de quedas"	"1"	"0"	"0"	"0.75"	"0.7653"	"0,9311"
"Confiabilidade"	"Preveção de falhas"	"1"	"1"	"1"	"0.75"	"0.7653"	"0,9311"
"Confiabilidade"	"Tempo de reparo"	"1"	"1"	"1"	"0.75"	"0.7653"	"0,9311"
"Confiabilidade"	"Disponibilidade"	"1"	"1"	"1"	"0.75"	"0.7653"	"0,9311"
"Confiabilidade"	"Tempo médio indisponível"	"120"	"60"	"0.5"	"0.75"	"0.7653"	"0,9311"
"Confiabilidade"	"Tempo médio de recuperação"	"35"	"30"	"0.85714"	"0.75"	"0.7653"	"0,9311"
"Seguranca"	"Prevenção à corrupção de dados"	"0"	"0"	"1"	"0.25"	"1"	"0,9311"
"Seguranca"	"Auditabilidade do acesso"	"1"	"1"	"1"	"0.25"	"1"	"0,9311"

Figura 10 – Resultado da consulta que retorna os valores calculados

Fonte: elaborada pelos autores

8.3 ANÁLISE DOS RESULTADOS DO ESTUDO DE CASO

A validação do *framework* evidenciou que é plenamente viável a especialização do modelo de qualidade com base na especificação de requisitos e na análise de riscos do software. O resultado QL = 0,9311 mostra-se coerente com os resultados dos testes, visto que a maioria das métricas obteve aderência de 100%.

Como o atributo Confiabilidade teve peso maior que Segurança, os resultados destas métricas tiveram maior impacto no resultado final do que se fosse uma simples média aritmética.

Quanto à ontologia, foi observada agilidade, facilidade e segurança com a automação dos cálculos.

Em síntese, destacam-se os seguintes fatores relevantes observados durante a validação, tanto do *framework* quanto da ontologia:

- a seleção das técnicas de testes mostrou-se uma tarefa não trivial para a qual foi fundamental buscar um parecer de um especialista na aplicação;
- tanto na seleção das técnicas como na análise de riscos, podem ocorrer conflitos de percepção entre diferentes especialistas. O *framework* está preparado para tratar este aspecto;
- a fórmula para cálculo do coeficiente de aderência das métricas (MA) deve ser específica conforme o domínio e a ordem de grandeza do resultado da métrica;
- para métricas cujo resultado seja unidade de tempo, deve-se utilizar como valor de referência um requisito especificado;
- como os testes não foram planejados com base no modelo de qualidade especializado para o sistema alvo do estudo de caso, e tampouco foram executados com objetivo de certificar o produto, nem todas as métricas foram coletadas;
- não foi possível fazer análises comparativas, pois não foi encontrado algum sistema que já tivesse sido certificado por outro processo para o estudo de caso.

9 CONCLUSÃO

Este trabalho apresentou um modelo para certificação de qualidade de produtos de software composto de um *framework* e uma ontologia. Um protótipo baseado em agentes de software foi utilizado para viabilizar um estudo de caso.

Segundo a ISO (ISO/IEC 14598-1, 2001), o processo de avaliação de qualidade do software engloba diversas atividades importantes, basicamente resumidas nas etapas de: identificação de atributos; definição dos procedimentos de avaliação; controle e medição; e análise dos julgamentos realizados. Vários fatores tornam o processo de avaliação complexo e difícil, tais como a escolha das métricas adequadas ou a formação do consenso de especialistas sobre uma mesma questão.

Acredita-se que uma das contribuições desta proposta seja, justamente, a possibilidade de automação do processo de certificação do produto por meio da ontologia, visto que os trabalhos relacionados pesquisados até o momento sugerem modelos conceituais e não apresentam, na sua maioria, propostas de arquitetura ou implementação de um sistema de apoio.

Os resultados demonstraram que o *framework* viabiliza a especialização do modelo de qualidade e a certificação do produto, da mesma forma que a ontologia foi capaz de representar os termos e conceitos envolvidos.

Além disso, apesar de todos os trabalhos relacionados apoiarem-se em referências normativas internacionais, para muitos dos trabalhos pesquisados o modelo de qualidade do produto utilizado é genérico e o *framework* proposto preconiza a especialização do modelo de qualidade.

Outro diferencial da proposta deste trabalho, em relação aos trabalhos relacionados, é a abordagem baseada em riscos, a qual preconiza que todo o esforço de especialização do modelo e a definição dos critérios de classificação sejam focados nos requisitos que são mais importantes devido aos riscos associados a eles.

Acredita-se que, tanto os conceitos do *framework*, quanto a ontologia de certificação de qualidade de produtos de software, podem ser incluídos em processos de certificação, formais ou não, sendo utilizados para certificação independente e voluntária ou, até mesmo, servir como base para certificação regulamentada e compulsória de software.

Na medida em que foram realizados pesquisa e reuso de ontologias já adotadas para o domínio em questão, este trabalho contribui, ainda, para a disseminação do uso de ontologias em aplicações orientadas à engenharia de software, evidenciando as suas potencialidades.

Como oportunidades de continuação da pesquisa e trabalhos futuros vislumbram-se diversas possibilidades, dentre as quais pode-se destacar: (1) Exercitar o modelo em uma aplicação cujos testes sejam planejados conforme o modelo de qualidade, já que o fato dos testes não terem sido conduzidos com o objetivo de certificação representa uma limitação do estudo de caso ora realizado; (2) Realizar um *survey* com usuários sobre as dificuldades ou limitações na etapa de identificação das técnicas de testes; (3) Evoluir o *framework* para computar critérios relacionados ao processo de desenvolvimento, tais como, nível de maturidade e reuso de componentes; (4) Criar selos de qualidade baseados em categorias e níveis de qualidade; (5) Incluir na ontologia um conjunto de conceitos para apoiar a análise de riscos segundo *MoSCoW priority* (PINKSTER *et al.*, 2006); e (6) Representar a importância relativa dos atributos da qualidade como números *fuzzy*, visando a tratar a vagueza e o conflito de percepção entre diferentes especialistas na análise de riscos e ponderação dos atributos.

10 REFERÊNCIAS

ABNT. Conheça a ABNT. Associação Brasileira de Normas Técnicas. s.d. Disponível em: http://www.abnt.org.br/m3.asp?cod_pagina=929. Acesso em: dez 2012.

ALEM FILHO, Pedro. O impacto da certificação de software e serviços na exportação. In: *Painel Setorial Programa Nacional de Certificação de Softwares e Serviços*. Rio de Janeiro: INMETRO, 2007.

ALVARO, Alexandre; ALMEIDA, Eduardo S. de; MEIRA, Silvio R. L. A component quality assurance process. In: SOQUA '07: International Workshop on Software Quality Assurance. 4., Dubrovnik. *Proceedings...* ACM SIGSOFT, September 2007.

ASSESPRO. Teste OK! Associação das Empresas Brasileiras de Tecnologia da Informação, 2009. Disponível em: http://www.testeok.com.br/. Acesso em: dez 2012.

BACH, J. The challenge of "good enough" software. 2003. Disponível em: http://www.satisfice.com/articles/gooden2.pdf. Acesso em: dez 2012.

BALCI, O. A methodology for certification of modeling and simulation applications. *Transactions on Modeling and Computer Simulation* (TOMACS), October 2001.

BASILI, Victor R.; CALDIERA, Gianluigi; ROMBACH, H. Dieter. The goal question metric approach, 1994. Disponível em: ftp://ftp.cs.umd.edu/pub/sel/papers/gqm.pdf. Acesso em: dez 2012.

BELCHIOR, A. D. Um modelo *fuzzy* para avaliação da qualidade de software. 1997. Tese de Doutorado - Departamento de Engenharia de Sistemas e Computação - COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 1997.

CHRISSIS, Mary Beth; KONRAD, Mike; SHRUM, Sandy. *CMMI for development* – Guidelines for process integration and product improvement. Boston: Addison-Wesley, 2004.

COLOMBO, Regina M. T.; GUERRA, Ana C. *Qualidade de produto de software*. PBQP/MCT, 2008. Disponível em: http://www.mct.gov.br/index.php/content/view/2867.html#lista. Acesso em: dez 2012.

DIXIT, Anurag. Intra-component security certification. In: ICAC3 '09: International Conference on Advances in Computing, Communication and Control, Mumbai, *Proceedings...* ACM SIGART, January 2009.

DUARTE, K. C., FALBO, R. A. Uma ontologia de qualidade de software. Workshop de Qualidade de Software. In: Simpósio Brasileiro de Engenharia de Software, 14. João Pessoa. *Anais*... p. 275-285, João Pessoa, 2000.

FERRAZ, Priscila; PROENÇA, Adriano. Serviços e exportação e a certificação de software. In: *Painel Setorial Programa Nacional de Certificação de Softwares e Serviços*. Rio de Janeiro: INMETRO, 2007.

FERREIRA, Mateus; GARCIA, Félix; RUIZ, Francisco; BERTOA, Manuel F.; CALERO, Coral; VALLECILLO, Antonio; PIATTINI, Mario; MORA, Beatriz. Medición del software ontología y metamodelo. Informe Técnico UCLM-TSI-001. Universidad de Castilla-la Mancha, noviembre 2006.

GAMA, Gleyson do Nascimento; OLIVEIRA, Sandro Ronaldo Bezerra. Spider-PQ: uma ferramenta de apoio à avaliação de produtos de software com base no MEDE-PROS. In: Encontro Anual de Computação, 9., Catalão, Goiás. *Anais...* ENACOMP, 2011. Disponível em: http://www.enacomp.com.br/2011/anais/trabalhos-aprovados/pdf/enacomp2011_submission_24.pdf. Acesso em: fev 2013.

GUSMÃO, C. M. G.; GUEDES, M. S.; MONTEIRO, M.; CAMPELLO, A.; AMORIM, L. OntoPRIME: Ontologia de riscos para ambientes de desenvolvimento de software multiprojetos. Universidade Federal de Pernambuco, Recife, Brasil. 2004.

HECK, Petra; KLABBERS, Martijn; EEKELEN, Marko. A software product certification model. *Software Quality Control*, v. 18, n. 1, March 2010. Disponível em: http://alexandria.tue.nl/repository/books/633706.pdf. Acesso em: dez 2012.

HETZEL, William, *The complete guide to software testing*. John Wiley & Sons, 1988.

HUMPHREY, Watts. Bugs or defects. In: *News at SEI*. Carnegie Mellon Univesity - Software Engineering Intitute. April, 1999. Disponível em: http://www.sei.cmu.edu/news-at-sei/columns/watts_new/1999/March/watts-mar99.htm. Acesso em dez 2012.

IEEE. *Standard glossary of software engineering terminology*. Institute of Electrical and Electronics Engineering, Inc, 1990.

ISO/IEC 25010. Systems and software engineering – Software product Quality Requirements and Evaluation (*SQuaRE*) – Quality models for software product quality and system quality in use, 2009.

ISO/IEC 25023. Systems and software engineering – Systems and software Quality Requirements and Evaluation (*SQuaRE*) – Measurement of system and software product quality, 2011.

ISO/IEC 25040. Software engineering - Software product Quality Requirements and Evaluation (*SQuaRE*) – Evaluation reference model and guide, 2009.

KAUR, Parminder; SINGH, Hardeep. Certification process of software components. *SIGSOFT Software Engineering Notes*, v. 33, n. 4, July 2008..

MAITINGUER, Sônia Thereza. Um método de avaliação especialista para produtos de software, desenvolvido a partir dos requisitos de um edital. 2004. Dissertação de Mestrado Profissional - Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica, 2004.

McGUINNESS, D. L.; NOY, N. F. Ontology development 101: a guide to creating your first ontology. Stanford University. 2001. Disponível em: http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html. Acesso em: dez 2012.

MYERS, Glenford J., *The art of software testing*. New York: John Wiley and Sons, 1979.

NBR ISO/IEC 14598-1 (2001). Tecnologia da informação – avaliação de produto de software. Parte 1: Visão geral. Associação Brasileira de Normas Técnicas, 2001.

NBR ISO/IEC 9126-1. Engenharia de software - Qualidade de produto. Parte 1: Modelo de qualidade. Associação Brasileira de Normas Técnicas, 2003.

OLIVEIRA, K. R., AdeQuaS: ferramenta *fuzzy* para avaliação da qualidade de software. 2002. Dissertação de Mestrado - MIA/UNIFOR, Engenharia de Software, Fortaleza, 2002.

PINKSTER, Iris; BURG, Bob van den; JANSSEN, Dennis; VEENENDAAL, Erik van. *Successful test management:* an integral approach. Berlin: Springer, 2006.

PMI. Um guia do conhecimento em gerenciamento de projetos (Guia PMBOK). 4. ed. Pennsylvania: PMI Inc., 2008.

PROTEGÉ. Welcome to Protegé. 2013. Disponível em: http://protege. stanford.edu/. Acesso em: dez 2012.

RIOSOFT. Avaliação de conformidade de produtos de software. Riosoft - agente Softex, 2009. Disponível em: http://www.riosoft.softex.br/cgi/ cgilua.exe/sys/start.htm?sid=53. Acesso em: dez 2012.

SANTOS, Adriana D. Processo de certificação de qualidade de produto de software na Embrapa: apostila de curso. Embrapa Informática Agropecuária, 2002. Disponível em: http://www.cnptia.embrapa.br/files/doc27.pdf. Acesso em: dez 2012.

SIBISI, M.; van WAVEREN, C. C. A process framework for customizing software quality models. In: AFRICON 2007*, Proceedings*..., p. 1-8, 2007.

SILVA, João Pablo S. Um sistema multiagente baseado em ontologias para apoio às inspeções de garantia da qualidade de software. Dissertação (Mestrado em Computação Aplicada) - Programa de Pós-Graduação em Computação Aplicada. Universidade do Vale do Rio dos Sinos. São Leopoldo, RS, 2010.

SOFTEX. MPS.BR - Melhoria de processo do software brasileiro, versão 1.1, 2007. Disponível em: http://www.softex.br/mpsbr/. Acesso em dez 2012.

SOFTWARE CERTIFICATIONS. *Guide to the CSTE common body of knowledge*. Quality Assurance Institute, version 6.2, 2006.

SOKOLSKY, Oleg; LEE, Insup; HEIMDAHL, Mats. Challenges in the regulatory approval of medical cyber-physical systems. EMSOFT '11: ACM International Conference on Embedded Software, 9., Taipei, *Proceedings...* ACM SIGBED, October 2011. Disponível em: http://dl.acm.org/citation.cfm?id=2038677&dl =ACM&coll=DL&CFID=233186484&CFTOKEN=30850098. Acesso em: mai 2012.

SPB. 5CQualiBr. Vetor qualidade de produto SPB". Portal do Software Público Brasileiro, 2009. Disponível em http://www.softwarepublico.gov.br/ 5cqualibr/xowiki/Qualidade. Acesso em dez 2012.

SPIDER, Projeto. sobre o SPIDER. 2009. Disponível em: http://www.spider.ufpa.br/ibndex.php?id=sobre. Acesso em: fev 2013.

VERMESAN, A. I. Software certification for industry - verification and validation issues in expert systems. In: International Workshop on Database and Expert Systems Applications, 9., Vienna, *Proceedings...* IEEE Computer Society, August, 1998. Disponível em: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=707373&tag=1. Acesso em dez 2012.

W3C. OWL web ontology language, overview. 2004. Disponível em: http://www.w3.org/TR/owl-features/. Acesso em: dez 2012.

W3C. SPARQL query language for RDF. 2008. Disponível em: http://www.w3.org/TR/rdf-sparql-query/. Acesso em: dez 2012.

WASSYNG, Alan; LAWFORD, Mark S.; MAIBAUM, Thomas S. E. Software certification experience in the Canadian nuclear industry: lessons for the future. In: ACM International Conference on Embedded software (EMSOFT '11). 9., Taipei, *Proceedings...* ACM SIGBED, October 2011. Disponível em: http://www.cas.mcmaster.ca/~lawford/papers/p219-wassyng.pdf. Acesso em: dez 2012.