UNISINOS

Programa Interdisciplinar de Pós-Graduação em

Computação Aplicada

Mestrado Acadêmico

Lizandra Bays dos Santos

Sistema de Apoio à Certificação de Qualidade de Produtos de Software

São Leopoldo, 2013

Lizandra Bays dos Santos

SISTEMA DE APOIO À CERTIFICAÇÃO DE QUALIDADE DE PRODUTOS DE SOFTWARE

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre, pelo Programa Interdisciplinar de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos – UNISINOS

Orientador: Dr. Sergio Crespo Coelho da Silva Pinto

São Leopoldo
2013

S237s Santos, Lizandra Bays dos
Sistema de apoio à certificação de qualidade de produtos de software / por Lizandra Bays dos Santos. – São Leopoldo, 2013.

94 f. : il. ; 30 cm.

Dissertação (mestrado) – Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, São Leopoldo, RS, 2013.
Orientação: Prof. Dr. Sergio Crespo Coelho da Silva Pinto, Ciências Exatas e Tecnológicas.

1.Agentes inteligentes (software). 2.Qualidade de software. 3.Certificação de qualidade de produtos de software. 4.Ontologias. 5.Framework (Programa de computador). 6.Teste de software. I.Pinto, Sérgio Crespo Coelho da Silva. II.Título.

CDU 004.4
004.05

Catalogação na publicação:
Bibliotecária Carla Maria Goulart de Moraes – CRB 10/1252

Lizandra Bays dos Santos

SISTEMA DE APOIO À CERTIFICAÇÃO DE QUALIDADE DE PRODUTOS DE SOFTWARE

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre, pelo Programa Interdisciplinar de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos – UNISINOS

Aprovado em: ___/____/____.

BANCA EXAMINADORA

_____

Prof. Dr. Sergio Crespo Coelho da Silva Pinto – UNISINOS (orientador)

_____

Prof. Dr. Arthur Torgo Gomez – UNISINOS

_____

Prof. Dr. Credine Menezes - UFRGS

AGRADECIMENTOS

Inicialmente, agradeço à CAPES pelo apoio financeiro.

Agradeço ao Serpro, empresa que, além de proporcionar meu crescimento profissional e pessoal, tem inspirado meus projetos de pesquisa, incluindo o objeto desta dissertação.

Ao meu orientador, Prof. Dr. Sergio Crespo Coelho da Silva Pinto, agradeço por acreditar em minha proposta inicial, quando eram apenas ideias vagas, pela confiança e por todo o apoio. Da mesma forma, aos professores Dr. João Carlos Gluz e Dr. Arthur Tórgo Gómez, agradeço pelo parecer, pelas sugestões e pelo incentivo durante o seminário de andamento.

Faço um agradecimento especial aos amigos, Prof. Ms. Eduardo Pretz e Esp. Karen Cristina Braga. Prof. Eduardo foi quem me ajudou a "plantar a semente" das ideias e conceitos, quando nem tínhamos a pretensão de que se transformassem num trabalho como esse. Karen foi quem me trouxe para o programa de mestrado. Aos dois, obrigada por insistirem.

> *Em tempos em que quase ninguém se olha nos olhos, em que a maioria das pessoas pouco se interessa pelo que não lhe diz respeito, só mesmo agradecendo àqueles que percebem nossas descrenças, indecisões, suspeitas, tudo o que nos paralisa, e gastam um pouco da sua energia conosco, insistindo*. (Martha Medeiros)

RESUMO

A qualidade do processo de desenvolvimento de software, somente, não garante a qualidade do produto que é desenvolvido, sendo necessário combinar técnicas de qualidade do processo e do produto. Existem diversos esforços na comunidade acadêmica e no mercado para de modelos de certificação de qualidade do processo, e observa-se uma lacuna no que tange à certificação de qualidade de produtos de software. Este trabalho o desenvolvimento de um sistema que visa dar suporte para a certificação de qualidade de produtos de software. Um *framework* de processo para certificação de produtos é apresentado, o qual é composto de um subprocesso de especialização do modelo de qualidade focado em riscos, um subprocesso de medição da qualidade e um sobprocesso de avaliação da qualidade. Para apoiar estes subprocessos, a arquitetura do sistema desenvolvido faz uso de ontologias para representar o conhecimento envolvido no modelo de qualidade e agentes de software para manipular os indivíduos nas ontologias.

Palavras-Chave: qualidade de software, certificação de qualidade de software, teste de software, ontologias, agentes.

ABSTRACT

The software development process quality does not assure product quality, being necessary to combine techniques of process quality and product quality. There are several efforts in the academic community and in the market for models of quality certification of the development process, so that there is a gap regarding the quality certification of the product developed. This work presents the development of a system that aims to support the certification of quality software products. A framework of product certification process is showed, which is composed of subprocesses of quality model specialization, quality measurement and quality assessment. To support these subprocesses, the developed system architecture uses ontologies to represent knowledge about the quality model and software agents for handling to individuals in the ontologies.

Keywords: software quality, software certification, software testing, ontologies, agents.

Lista de figuras

Lista de Tabelas

Lista de Siglas

ABIC	Associação Brasileira das Indústrias de Café
ABNT	Associação Brasileira de Normas Técnicas
ASSESPRO	Associação das Empresas Brasileiras de Tecnologia da Informação
BDI	*Belief-Desire-Intention*
CBOK	*CSTE Common Body of Knowledge*
CMMI	*Capability Maturity Model Integration*
CMU	*Carnegie Mellon University*
CSTE	*Certified Software Tester*
CTI	Centro de Tecnologia da Informação Renato Archer
EDS	*Electronic Data Systems*
eSCM	*eSourcing Capability Model*
FINEP	Financiadora de Estudos e Projetos
GQM	*Goal Question Metric*
IBM	*International Business Machines*
IEC	*International Eletrotechnical Comission*
IEEE	*Institute of Electrical and Electronic Engineers*
INMETRO	Instituto Nacional de Metrologia, Qualidade e Tecnologia
ISO	*International Organization for Standardization*
JTC	*Joint Technical Committee*
MCT	Ministério da Ciência e Tecnologia
MFAQS	Modelo *Fuzzy* para Avaliação da Qualidade de Software
MPS.BR	Melhoria de Processo do Software Brasileiro
NASA	*National Aeronautics and Space Administration*
NBR	Norma Brasileira
OWL	*Web Ontology Language*
PA	*Process Area*
PIPCA	Programa interdisciplinar de Pós-Graduação em Computação Aplicada
PMBOK	*Project Management Body of Knowledge*
PMI	*Project Management Institute*
PNAFAM	Programa Nacional de Apoio à Gestão Administrativa e Fiscal dos Municípios Brasileiros
PROIMPE	Programa de Estímulo ao Uso de Tecnologia da Informação em Micro e Pequenas Empresas

RDF	*Resource Description Framework*
RRBT	*Risk & Requirement Based Testing*
SBC	Sociedade Brasileira de Computação
SBQS	Simpósio Brasileiro de Qualidade de Software
SC	*Subcommittee*
SEI	*Software Engeneering Institute*
SMA	Sistemas Multiagentes
SOA	*Service Oriented Architecture*
SOFTEX	Associação para a Promoção da Excelência do Software Brasileiro
SPB	Software Público Brasileiro
SPIDER	*Software Process Improvement: Development and Research*
SQL	*Structured Query Language*
SQuaRE	*Software product Quality Requirements and Evaluation*
TI	Tecnologia da Informação
TIC	Tecnologia da Informação e Comunicação
UFPA	Universidade Federal do Pará
UNISINOS	Universidade do Vale do Rio dos Sinos
W3C	*Word Wide Web Consortium*
XML	*Extensive Markup Language*

Sumário

1. INTRODUÇÃO

A demanda cada vez maior por sistemas computacionais impacta diretamente no tamanho e na complexidade dos mesmos. Mesmo com o envolvimento de diversos profissionais qualificados e a existência de metodologias para a organização do processo, a característica intangível do software faz da construção de sistemas grandes e complexos uma tarefa ainda desafiadora. Por mais organizada e qualificada que seja uma equipe, bem como a metodologia empregada pela mesma, falhas provenientes de diversos pontos do processo podem provocar erros, estes tendo origem tanto na especificação quanto na própria construção do respectivo sistema computacional. Como consequência, podem ocorrer atrasos na execução das tarefas, insatisfação dos usuários ou até mesmo, em casos mais graves, o insucesso total do projeto. Todos estes problemas levam à necessidade de garantir que um dado sistema computacional possua o menor número de falhas possível, reduzindo custos de manutenção e aumentando a qualidade do mesmo.

No contexto do desenvolvimento de software, a avaliação e a certificação se apresentam como importantes ferramentas para se identificar a situação de tal produto e viabilizar a tomada de decisões sobre ele.

Ampliando o contexto de análise para a indústria de software como um todo, percebe-se que existe uma forte demanda, no mundo todo, por padrões e melhores práticas de relacionamento, gestão e operação dos “serviços de TI” que são objeto de terceirização (FERRAZ e PROENÇA, 2007). Para Alem Filho (2007), da Associação Brasileira de Desenvolvimento Industrial, neste cenário, a certificação, tanto de produtos como de serviços de software, justifica-se, do ponto de vista do comprador, pois garante que tal produto ou serviço esteja aderente a requisitos mínimos com um processo de seleção bastante simplificado, pois o custo da verificação, neste caso, é transferido para o fornecedor. Já, do ponto de vista do fornecedor, a certificação pode ser vista como um requisito para competir no mercado e pode trazer ganhos reais em qualidade e produtividade, o que favorece ainda mais a competitividade.

Na indústria em geral existem diversos segmentos de produtos e serviços para os quais existem programas específicos de certificação ou selos de qualidade, tais como, café, brinquedos, feijão, vestuário, energia renovável, usinas de preservação de madeira, agentes de crédito, sistemas eletrônicos de segurança (TOTUM, 2013). A certificação pode agregar valor e ampliar o consumo na medida em que promove a melhoria contínua do produto. Para a

ABIC (Associação Brasileira da Indústria de Café), que mantem um importante programa de qualidade do café, para as indústrias, a vantagem da certificação está na possibilidade de melhor focar seu negócio, lançando ou reposicionando suas marcas de acordo com o perfil de seus consumidores. Para o varejo, é um diferencial importante poder ofertar produtos certificados e uma garantia no quesito segurança. Já, para os consumidores, há um papel também educativo: fazê-los descobrir que os produtos não são todos iguais, que existem diferenças inclusive entre as marcas produzidas por uma mesma empresa e que existem produtos para todos os gostos e bolsos (ABIC, 2013). No contexto da indústria de software entende-se que não é diferente, pois nem todas as aplicações apresentam os mesmos riscos e nem todas precisam o mesmo nível de qualidade.

A certificação de software pode se apresentar como um elemento qualificador importante no do mercado internacional (ALEM FILHO, 2007). Além disso, caso o fornecedor tenha objetivos menos pretenciosos no mercado, um modelo para certificação de qualidade dos produtos pode guiar, não só a definição dos critérios de aceitação pelo cliente na fase de especificação, mas também os testes de aceitação, podendo, ainda, viabilizar a emissão de um selo atestando o nível de qualidade do produto final.

1.1. Contextualização do problema

Existem diversos esforços na comunidade científica na busca pela qualidade de software por meio de modelos de qualidade do processo. Nota-se uma lacuna nestes esforços, pois a qualidade do processo, embora fundamental, não garante a qualidade do produto.

Pesquisadores do CTI (Centro de Tecnologia da Informação Renato Archer) definem que “a qualidade do produto é o que buscamos e a qualidade do processo é o meio para conseguirmos” (COLOMBO e GUERRA, 2008). Desta forma, entende-se que para a emissão de um certificado ou selo de qualidade do produto, são necessárias técnicas de controle da qualidade para aferir o grau de aderência de tal produto a requisitos mínimos, segundo um modelo de qualidade previamente definido.

A questão chave desta proposta é pesquisar como é possível atestar a conformidade com padrões de qualidade de um produto de software e aferir o grau de aderência deste produto aos padrões.

A hipótese trabalhada é a de que deve ser estabelecido um modelo de qualidade especializado, de acordo com os requisitos do domínio da aplicação. Além disso, faz-se

necessário um modelo de verificação de evidências dos requisitos e fatores da qualidade do modelo especializado, que permita a avaliação da conformidade do software a cada um destes fatores. Por fim, o estabelecimento de critérios de priorização e graduação das evidências e fatores de qualidade verificados, pode estabelecer o nível de conformidade do software, o que pode ser chamado de certificado ou selo de qualidade.

Em pesquisa anterior de (SANTOS e PRETZ, 2010) foi proposto um *framework* para especialização de modelos de qualidade que visa apoiar a definição dos atributos da qualidade específicos para o domínio de uma aplicação, baseado nos riscos e requisitos do produto, que contempla, ainda, as métricas e técnicas para mensuração das evidências dos atributos da qualidade. A ideia principal daquela proposta pode ser resumida como uma rastreabilidade desde o requisito do software até a técnica de teste que permitirá a medição da qualidade, considerando os riscos e atributos da qualidade associados.

A especialização de modelos da qualidade de produtos de software pode ser custosa devido à complexidade das relações entre os diferentes elementos na hierarquia do modelo, tais como requisitos da aplicação e atributos da qualidade. Além disso, o processo de avaliação também pode ser dispendioso por contar com resultados de diferentes avaliadores, além dos diferentes pesos dos atributos de qualidade envolvidos. Desta forma, percebe-se uma oportunidade de pesquisa na automação do processo por meio de técnicas de ontologias e agentes para representação e manipulação do conhecimento.

Como não foi encontrado na literatura algum estudo aprofundado acerca da certificação de produtos com base em modelos de qualidade especializados, acredita-se que um diferencial desta proposta é justamente apresentar um processo para a especialização do modelo.

Além disso, o principal diferencial da presente proposta é a aplicação de técnicas de ontologias e agentes para representar e fazer uso do conhecimento, visando automatizar os processos de especialização do modelo de qualidade e de certificação do produto.

1.2. Objetivo da Pesquisa

O objetivo principal deste trabalho é o desenvolvimento de uma ferramenta capaz de apoiar todo o processo de certificação de qualidade de produtos de software, desde a

especialização do modelo de qualidade até a avaliação e classificação do produto segundo critérios pré-estabelecidos.

Para atingir tal objetivo geral, foram definidos os seguintes objetivos específicos de pesquisa:

- Investigar o estado da arte em certificação de software, além dos conceitos e tecnologias envolvidas: qualidade de software, testes de software, sistemas baseados em agentes e ontologias;
- Refinar os subprocessos do *framework* para especialização de modelos de qualidade proposto por (SANTOS e PRETZ, 2010) e criar o *framework* de certificação de qualidade de produtos de software;
- Desenvolver, testar e validar modelos de ontologias para representar o conhecimento envolvido;
- Realizar projeto e desenvolvimento do protótipo da solução proposta, baseado ontologias e em agentes.

De modo resumido, esta proposta visa o desenvolvimento de um sistema que seja capaz de apoiar o relacionamento dos riscos de negócio aos requisitos da aplicação e, a partir daí inferir o atributo da qualidade correspondente. De posse deste relacionamento, o sistema deve ser capaz ainda, de obter as métricas necessárias para que seja conduzido o processo de avaliação da qualidade. A partir dos valores mensurados na avaliação do produto o sistema deve classifica-lo e informar o nível de qualidade do mesmo.

1.3. Metodologia

Esta pesquisa baseia-se em técnicas recentes relacionadas a ontologias, tecnologia de agentes e qualidade de software. Para viabilizar o atendimento dos objetivos específicos propostos, foi adotada a metodologia de trabalho detalhada abaixo:

- Investigação do estado da arte em certificação de software, além dos conceitos e tecnologias envolvidas: implica em atividades de pesquisa na literatura recente não só sobre qualidade de software, testes de software, sistemas baseados em agentes e ontologias, mas também sobre trabalhos relacionados ao tema;
- Desenvolvimento do *framework* de certificação de qualidade de produtos de software: implica em complementar o *framework* para especialização de modelos de qualidade

proposto por (SANTOS e PRETZ, 2010), criando o subprocesso de avaliação da qualidade, estabelecendo critérios de julgamento e classificação do produto conforme a ponderação dos atributos do modelo de qualidade;

- Desenvolvimento, teste e validação de modelos de ontologias para representar o conhecimento envolvido: implica na aplicação de processo interativo de desenvolvimento e no uso de ferramentas de apoio para criar ou adequar as ontologias necessárias e testá-las;
- Projeto e desenvolvimento do protótipo da solução proposta, baseado em agentes: implica na definição dos objetivos e desenvolvimento dos agentes que deverão manipular as ontologias criadas.

1.4. Organização do documento

Este documento apresenta o embasamento teórico e tecnológico utilizado na pesquisa, bem como descreve o trabalho, os objetivos, a metodologia adotada e os resultados obtidos no desenvolvimento do sistema de apoio à certificação de qualidade de produtos de software.

A partir daqui, este documento está organizado na seguinte estrutura de capítulos:

- *Capítulo 2*: traz o resultado da investigação e revisão bibliográfica sobre os conceitos teóricos utilizados no trabalho e das tecnologias envolvidas;
- *Capítulo 3*: apresenta os principais trabalhos relacionados encontrados na literatura e uma análise comparativa entre eles;
- *Capítulo 4*: apresenta o *framework* pra certificação de qualidade de produtos de software que é base para o sistema proposto neste trabalho;
- *Capítulo 5*: descreve a visão geral e os pacotes de ontologia, agente e protótipo do sistema de apoio a certificação de qualidade de produtos de software;
- *Capítulo 6*: apresenta as etapas de verificação e validação do *framework* e do sistema desenvolvido;
- *Capítulo 7*: conclui o trabalho, apresentando suas limitações, contribuições, e perspectivas de continuação da pesquisa em trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA E TECNOLÓGICA

Este capítulo traz um embasamento teórico a respeito dos temas tratados nesta dissertação. Inicialmente, a Seção 2.1 trata do tema mais abrangente que é a qualidade de software e visa contextualizar e conceituar a qualidade do processo de desenvolvimento e a qualidade do produto de software. Em seguida na Seção 2.2, é tratada da delimitação do tema que é a certificação da qualidade de software, trazendo um breve relato da investigação na literatura sobre o estado da arte para este tema. Por fim, a Seção 2.3 conceitua as principais tecnologias utilizadas no desenvolvimento da proposta que são ontologias e sistemas baseados em agentes.

2.1. Qualidade de Software

Um conceito de qualidade do produto amplamente utilizado na manufatura foi definido por (CROSBY, 1979): “Qualidade é conformidade com os requisitos”.

Qualidade, no contexto do INMETRO (Instituto Nacional de Metrologia, Qualidade e Tecnologia), compreende o grau de atendimento (ou conformidade) de um produto, processo, serviço ou ainda um profissional, a requisitos mínimos estabelecidos em normas ou regulamentos técnicos, ao menor custo possível para a sociedade (INMETRO, 2012).

A partir destes conceitos, pode-se concluir que o nível de qualidade de um produto é o quanto o mesmo está de acordo com requisitos ou normas pré-estabelecidas.

Trazendo o conceito para o contexto da Engenharia de Software, o IEEE (*Institute of Electrical and Electronics Engineers*) define qualidade como sendo o grau até o qual um componente, sistema ou processo atende aos requisitos especificados e/ou às necessidades e expectativas do usuário/consumidor (IEEE, 1990).

As características específicas do software como produto, tais como complexidade, invisibilidade e intangibilidade, produção sob medida, não sofrer desgaste com o tempo e o uso, não ter prazo de validade e, praticamente não ter custo de distribuição (CAPOVILLA, 1999), trazem certa dificuldade em aplicar de forma direta os conceitos desenvolvidos na produção de produtos manufaturados. Percebe-se aí, então, um desafio da Engenharia de Software, que é estabelecer processos repetíveis e padronizados para o desenvolvimento de um produto complexo e intangível com qualidade. A qualidade de software é a área da

Engenharia de software que preconiza a normatização do processo de desenvolvimento. Mesmo abordando prioritariamente o processo, o principal objetivo é a garantia da qualidade do produto final.

A disciplina de Engenharia de Software apresenta, então, a qualidade de software sob dois aspectos: garantia da qualidade de controle da qualidade.

Segundo o IEEE (1990), uma das definições para garantia da qualidade de software é que é o conjunto de atividades planejadas para avaliar os processos pelos quais os produtos são desenvolvidos ou fabricados. Já o controle da qualidade 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 controle da qualidade está relacionado à qualidade do produto desenvolvido.

Pode-se definir qualidade de produto de software como a conformidade a requisitos funcionais e de desempenho declarados explicitamente, padrões de desenvolvimento claramente documentados e as características implícitas que são esperadas de todo software desenvolvido profissionalmente. Por necessidades explícitas, pode-se entender requisitos do usuário; necessidades implícitas relacionam-se, por exemplo, com a performance de execução do sistema ou até mesmo com o cumprimento do cronograma e o orçamento do desenvolvimento do produto (COLOMBO e GUERRA, 2008).

A qualidade do processo de desenvolvimento é um meio fundamental e imprescindível para se atingir a tão esperada qualidade do produto final entregue ao usuário. Porém, seguir processos normatizados e controlados para no desenvolvimento, não garante a qualidade do produto de software final, fazendo-se necessária a aplicação de técnicas de controle de qualidade.

2.1.1. Modelos de Qualidade de Software

Alguns modelos de qualidade, como os que serão discutidos nesta Seção, se apresentam como referência para qualidade de produtos de software. Estes *frameworks*, de modo geral, trazem um conjunto de características que um software deve apresentar, organizadas de forma hierárquica, relacionando subcaracterísticas e métricas que permitem verificar a existência ou não de cada uma destas características.

O modelo de McCall (MCCALL, RICHARDS e WALTERS, 1977) é, possivelmente, o primeiro encontrado na literatura cientifica. Trata-se de um modelo que tenta estabelecer uma ligação entre usuários e desenvolvedores, focando nos fatores de qualidade do software que refletem as opiniões dos usuários e as prioridades dos desenvolvedores (COLOMBO e GUERRA, 2008). O modelo de qualidade de (McCALL, RICHARDS e WALTERS, 1977) apresenta, como mostra a Figura 1, três perspectivas principais para definir e identificar a qualidade de um produto de software: revisão do produto (habilidade de se submeter a mudanças), transição do produto (adaptabilidade a novos ambientes) e operação do produto (suas características de operação).

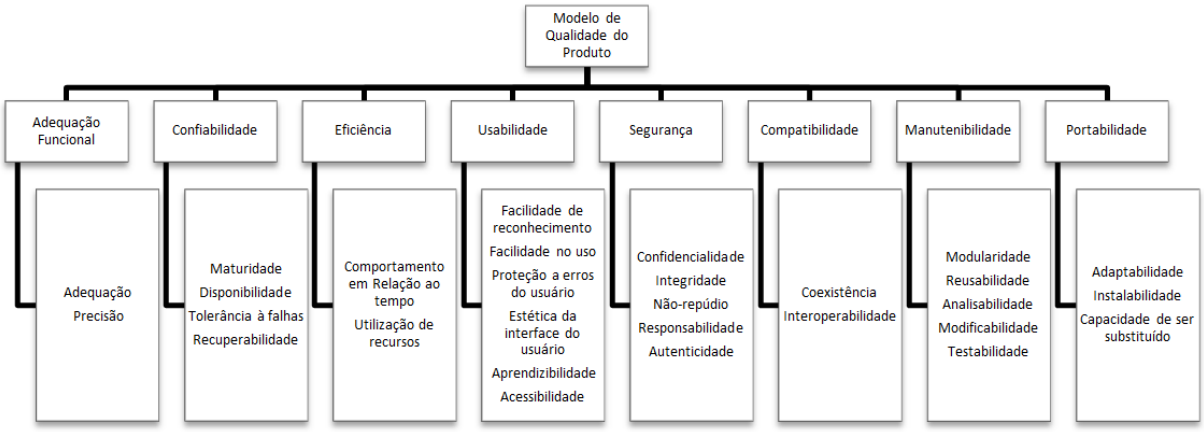
Figura 1 – Modelo de Qualidade de McCall

Fonte: MCCALL, RICHARDS e WALTERS (1977)

McCall define um modelo de qualidade de forma hierárquica, com 11 fatores de qualidade (na Figura 1, fora do triângulo). Cada um dos fatores se subdivide em 23 critérios de qualidade, que ainda podem ser subdivididos em medidas de controle, as quais não estão definidas no modelo.

A série de normas ISO/IEC 25000 - *Software Engineering - Software product Quality Requirements and Evaluation* (*SQuaRE*), na sua divisão ISO/IEC 25010, trata do modelo de qualidade em uso para produtos de software (ISO/IEC 25010, 2009). É uma reestruturação das normas NBR ISO/IEC 9126 e fornece um modelo baseado em um conjunto de características que um produto de software deve apresentar, o qual também é organizado hierarquicamente em subcaracterísticas e atributos, conforme Figura 2.

Figura 2 – Modelo de Qualidade ISO/IEC 25010

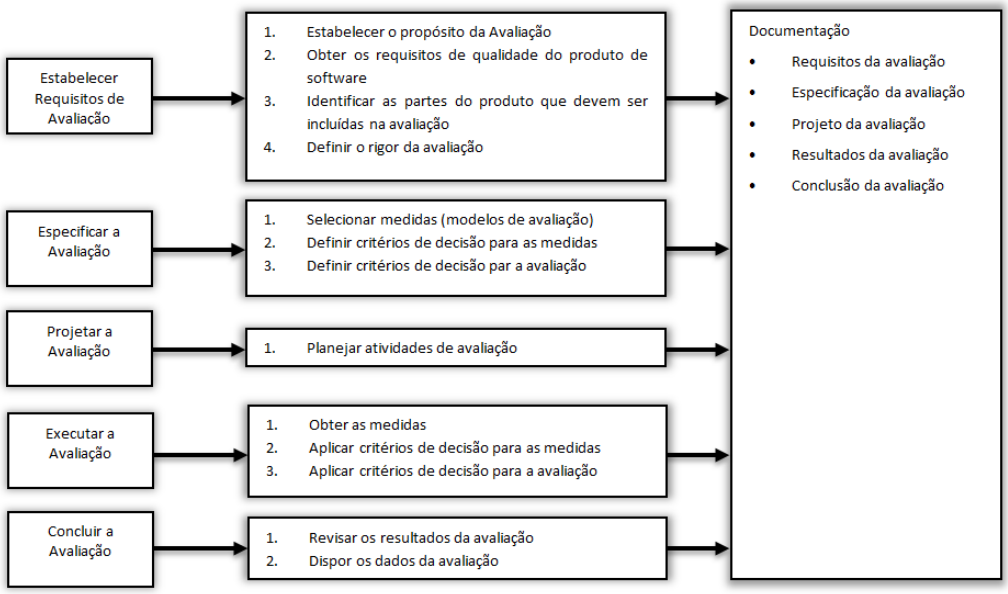


Fonte: ISO/IEC 25010 (2009)

Para Colombo e Guerra (2008), esta norma pode ser aplicada para definição dos requisitos de qualidade de um produto de software, avaliação da especificação de software para verificar se ele irá satisfazer aos requisitos de qualidade durante o desenvolvimento, descrição de particularidades e atributos do software implementado, por exemplo, em manuais de usuário, avaliação do software desenvolvido, antes da entrega, avaliação do software desenvolvido, antes da aceitação.

Esta divisão da *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 divisão ISO/IEC 25040 – *Evaluation reference model and guide*, a qual é uma releitura da série NBR ISO/IEC 14598 (NBR ISO/IEC 14598-1, 2001), e cujo processo é apresentado na Figura 3.

Figura 3 – Modelo de referência de avaliação de qualidade de produto de software conforme a ISO/IEC 25040



Fonte: ISO/IEC 25040 (2009)

A norma ISO/IEC 25040 traz conceitos que tratam de como avaliar a qualidade de software e define um modelo de processo de avaliação genérico, conforme Figura 3. O processo proposto pode ser usado para avaliar produtos já existentes ou produtos intermediários, isto é, em desenvolvimento. Pode ser utilizada por laboratórios de avaliação, fornecedores de software, compradores de software, usuários e entidades certificadoras, cada qual com seu objetivo (ISO/IEC 25040, 2009). Em termos de características de qualidade, pode ser usada a norma ISO/IEC 25010, ou o modelo de McCall, por serem reconhecidos internacionalmente.

2.1.2. Teste de Software

O controle da qualidade, já discutido em seção anterior, 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. Podem ser vistos como uma defesa contra possíveis problemas ocasionados pelo "fator-humano" na Engenharia de Software, tais como, problemas de comunicação entre os diversos perfis do projeto, erros na

transcrição de ideias para especificações, ambiguidades nas especificações textuais, entre outras.

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 encontrar 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 com os 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 o mesmo satisfaz os resultados requeridos.

É importante lembrar, também, que teste é um experimento controlado, ou seja, deve ser planejado e o objetivo a ser alcançado deve ser bem definido.

Desta maneira, as definições apresentadas conduzem à conclusão de que teste é o processo de executar o software de maneira controlada, com objetivo de avaliar seu comportamento conforme requisitos previamente definidos e de detectar erros. Requer domínio de todos os passos de sua execução e um rigoroso planejamento, envolvendo a escolha de técnicas, critérios de seleção de casos de teste e procedimento para suas aplicações.

Técnicas de testes são os meios utilizados pelos testadores para realizarem os objetivos de testes e há vasta bibliografia disponível abordando o assunto. As técnicas são dependentes dos tipos de testes, e muitas vezes se confundem com eles. Tipos de testes são relacionados aos objetivos dos testes, ou seja, "o quê testar", enquanto as técnicas dizem respeito ao "como testar". As técnicas de teste são classificadas de acordo com a origem das informações utilizadas para estabelecer os requisitos de teste (DIAS NETO, 2009).

Os resultados das atividades de testes geram um conjunto de medidas que devem ser utilizadas para gerar 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, Caldiera e Rombach (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, CALDIERA e ROMBACH, 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 Figura 4. O processo de definição de objetivos, no nível conceitual, é uma das etapas mais críticas para o sucesso desta abordagem, na qual um objetivo possui um propósito e três coordenadas, a saber:

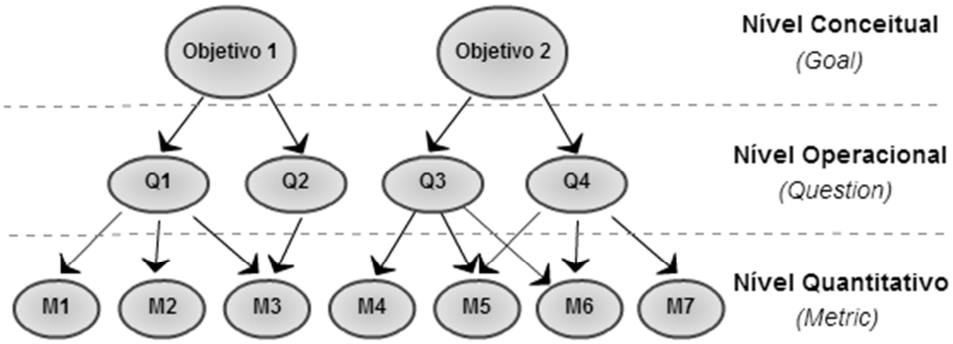
Propósito: um verbo que represente um objetivo, por exemplo "*avaliar*";

Questão: um adjetivo do objeto, por exemplo "*a maturidade*";

Objeto: o objeto em avaliação, por exemplo "*do software*";

Ponto de vista: por exemplo, "*segundo a perspectiva do cliente*".

Figura 4 – Estrutura da abordagem GQM



Fonte: BASILI, CALDIERA e ROMBACH (1994)

Pela abordagem GQM, é no nível conceitual onde são definidos os objetivos (*Goals*) para o objeto a ser medido, considerando modelos de qualidade, pontos de vista e ambientes particulares. No nível operacional é definido um conjunto de questões (*Questions*) que devem auxiliar na caracterização do objeto de estudo dentro do contexto da qualidade. As métricas (*Metrics*) são definidas no nível quantitativo e consistem em um conjunto de dados a serem obtidos, relacionado às questões definidas que visam responde-las de forma quantitativa.

Entre as vantagens observadas no método GQM, destacam-se, apoio na identificação de métricas úteis e relevantes, e a análise e interpretação dos dados coletados.

2.1.3. 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. Para esta base de conhecimento, que tem foco em riscos de projeto, a gestão de riscos tem dois objetivos: aumentar a probabilidade e o impacto dos eventos positivos e reduzir a probabilidade e o impacto dos eventos negativos (PMI, 2008).

No contexto da qualidade de software, as bases da definição de risco não são diferentes: preocupam-se com acontecimentos futuros e há um fator de incerteza. Para James Bach (2003), a qualidade do software é considerada boa o suficiente, quando acredita-se que os riscos do mesmo são aceitavelmente baixos.

Riscos de projeto, segundo Pressmann (1995), seriam aqueles que identificam riscos orçamentários, de cronograma, de pessoal, de recursos, de clientes e de requisitos que impactam no projeto 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 dos mesmos, deve ser incluída nos devidos planos 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 resultado 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 perdas.

O CSTE CBOK define ainda, que controle é tudo o que visa a redução dos riscos, de 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, oportunizando 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.

Uma análise de ricos deve considerar a probabilidade de ocorrência do evento de risco, bem como a perda potencial, ou impacto associado com tal perda, objetivando minimizar a frequência ou o impacto do risco (SOFTWARE CERTIFICATIONS, 2006).

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. James Bach (2003) destaca que, em um contexto particular, alguns fatores podem ser mais importantes do que

outros e que é necessário ser capaz de identificar as diferenças entre importante e não importante, necessário e não necessário.

Uma alternativa para a impossibilidade de testar tudo e para otimizar os esforços de testes é direcioná-los ao que é mais importante no contexto da aplicação, focando e priorizando os 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, aos testes que cobrem os maiores riscos seja dado tratamento especial. Além disso, um inventário de requisitos é considerado. Os requisitos são relacionados aos riscos do produto e, com base nesta informação, são criados os testes que cubram os principais riscos (PINKSTER, BURGT e VEENENDAAL, 2006).

Riscos isolados não são insuficientes para as bases da avaliação do produto de modo que a especificação dos requisitos também é necessária. Uma maneira apropriada de relacionar requisitos e riscos é pela análise do funcionamento de determinada funcionalidade, e os efeitos de seu não funcionamento (PINKSTER, BURGT e VEENENDAAL, 2006).

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

2.2.Certificação de Qualidade de Software

Observam-se movimentos nacionais e internacionais por normatização na área de Engenharia de Software. Neste contexto, há a participação da ISO (*International Organization for Standardization*), organização não governamental, cuja missão é promover o desenvolvimento de padrões e atividades relacionadas de âmbito mundial visando facilitar a troca internacional de bens e serviços e desenvolver cooperação em atividade intelectual, científica, tecnológica e econômica (ISO, 2003). A Associação Brasileira de Normas Técnicas (ABNT), por sua vez, é o órgão responsável pela normalização técnica no Brasil, fornecendo a base necessária ao desenvolvimento tecnológico brasileiro (ABNT, 2003). A ABNT é a única e exclusiva representante no país das entidades internacionais ISO e IEC (*International Electrotechnical Commission*), entre outras.

Para o INMETRO a certificação de produtos ou serviços, sistemas de gestão e pessoas é, por definição, realizada pela terceira parte, isto é, por uma organização independente acreditada para executar essa modalidade de avaliação da conformidade (INMETRO, 2012).

Avaliação de conformidade, segundo a ISO é uma atividade que fornece demonstração de que requisitos especificados relativos a um produto, processo, sistema, pessoa ou organismo são cumpridos (COLOMBO e GUERRA, 2008).

Para Fabrini, Fusani e Lami (2006) avaliação de conformidade é um processo que inclui, mas não se limita a, testar e analisar objetos. Objetos de certificação não são necessariamente produtos, processos ou pessoas, mas algumas de suas propriedades, que sejam submetidas à verificação de conformidade com um ou mais padrões de requisito.

Certificação de Software, por sua vez, 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 GUERRA, 2008).

Avaliar um produto de software é atribuir certo valor a esse produto, com base em requisitos pré-estabelecidos e sob demanda de um patrocinador (SANTOS, 2002). Os requisitos da qualidade que o produto sob avaliação deve possuir devem ser definidos em um modelo de qualidade. Para Colombo e Guerra (2008), um modelo de qualidade deve conter um conjunto de características e atributos e é utilizado para definir requisitos por meio de métricas para os atributos.

Um modelo de qualidade de produto pode ser utilizado em processos de desenvolvimento de software, e aquisição/fornecimento de Software e de avaliação e Certificação de produtos (COLOMBO e GUERRA, 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).

Fabrini, Fusani e Lami (2006) sugerem que um produto certificado por um esquema e organismo confiáveis tem mais valor agregado se comparado a outro não certificado. Valor agregado nem sempre implica em prova ou garantia, mas apenas reforça a confiança no

produto. Para o fornecedor, a maior confiança no serviço é, provavelmente, quando o consumidor prefere o bem ou serviço certificado em relação a um similar não certificado.

2.2.1. Modelos de Referência para Certificação de Software

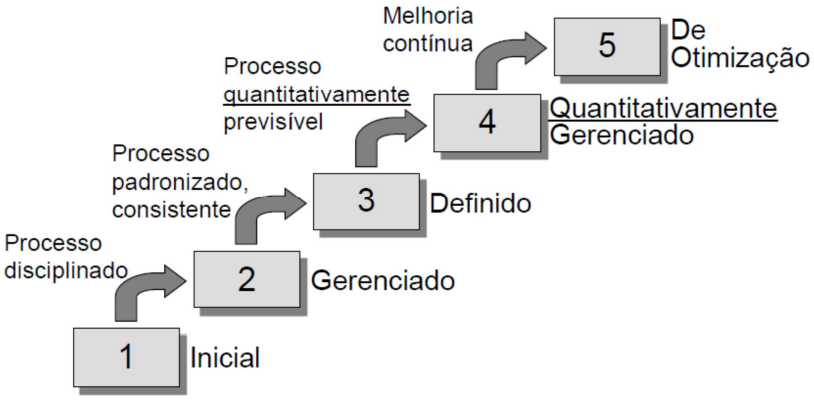
Para se produzir software com qualidade, dentro de prazos e custos controlados, existem alguns modelos de referência que reúnem boas práticas para melhoria do processo de desenvolvimento.

O CMMI (*Capability Maturity Model Integration*) 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 S. Humphrey e considera os níveis de capacidade e maturidade da organização (CHRISSIS, 2004).

O CMMI é definido pelo próprio SEI como sendo uma sistematização das melhores práticas em Engenharia de Software. Este modelo de referência possui dois tipos de representação: estagiada e contínua. Na representação estagiada a organização vai adquirindo níveis de maturidade para um conjunto definido de áreas de processo para cada nível. Já na representação contínua, a organização pode escolher os processos a serem priorizados para melhoria e vai buscando atingir níveis de capacidade em cada um destes processos.

A Figura 5 representa os cinco níveis de maturidade do CMMI.

Figura 5 – Níveis de Maturidade CMMI



Fonte: CHRISSIS (2004)

Os processos da organização, independente do tipo de representação escolhida, são mapeados em termos de áreas de processo. Uma área de processo (PA - *Process Area*) é um agrupamento de práticas relacionadas entre si, em uma determinada área, que, quando implementadas coletivamente, satisfazem um conjunto de objetivos considerados importantes para a promoção de melhorias naquela área (CHRISSIS, 2004).

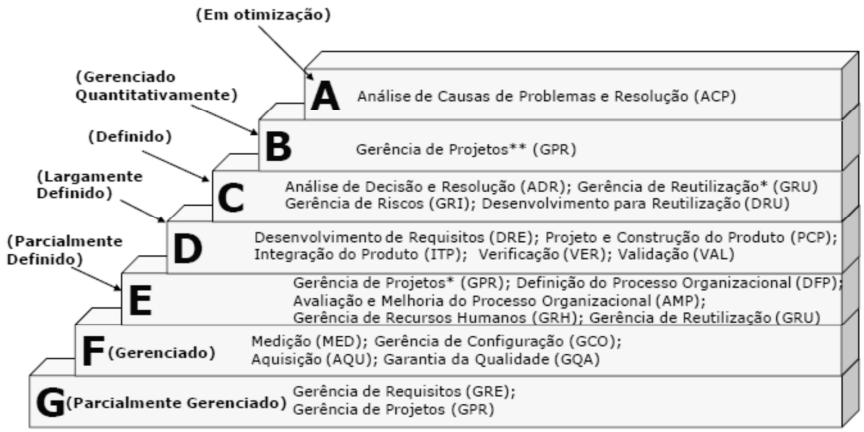
No CMMI as áreas chave de processo se desmembram em objetivos genéricos e específicos, os quais são detalhados em práticas genéricas e específicas, respectivamente. As 22 áreas de processo do CMMI são classificadas em quatro categorias:

- *Process Management* (Gerenciamento de Processos);
- *Project Management* (Gerenciamento de Projetos);
- *Engineering* (Engenharia);
- *Support* (Suporte ou Apoio).

Visando apoiar pequenas e médias empresas brasileiras que não possuem capital alto para investir no CMMI, surgiu o MPS.BR (Melhoria de Processo do Software Brasileiro). 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 (MPS.BR, 2009). Estes modelos de referência, visam certificar a qualidade do processo de desenvolvimento de software.

A Figura 6 representa os níveis de maturidade do MPS.BR.

Figura 6 – Níveis de Maturidade MPS.BR



Fonte: MPS.BR (2009)

De forma análoga ao CMMI, o MPS.BR é organizado em níveis de maturidade, processos, resultados esperados e atributos dos processos.

Motivado pelo crescimento da terceirização de serviços de TI no mundo, um consórcio liderado pela *Carnegie Mellon University* (CMU) e contando com a participação de empresas como: Accenture, EDS, IBM Global Services e Satyam Computer Services, desenvolveu o modelo *eSCM* (*eSourcing Capability Model*). Os esforços de construção do modelo foram divididos em duas frentes de trabalho, focadas nas visões do fornecedor (*eSCM-SP* - eSCM *for Service Providers*) e cliente dos serviços de TI (*eSCM-CL - eSCM for Clients*).

O *eSCM* trata-se é um modelo de referência de capacidade para terceirização em serviços habilitados por TI e possui três propósitos:

- Servir de guia aos provedores de serviço, auxiliando-os na melhoria da capacidade no decorrer do ciclo de vida do fornecimento do serviço;
- Prover aos clientes critérios objetivos de avaliação da capacidade dos provedores de serviço;
- Prover um padrão que pode ser usado pelos provedores de serviço como um diferencial em relação à concorrência.

Nos dois modelos do eSCM, seja para os provedores de serviço (SP) ou para as organizações-cliente (CL), as práticas definidas são distribuídas em três dimensões: o ciclo de vida do fornecimento, áreas de capacidade e níveis de capacidade (HYDER *et al*., 2009), sendo muito semelhante à estrutura do CMMI.

2.3.Tecnologias

Nesta Seção são abordadas as tecnologias de agentes de software e ontologias selecionadas para o desenvolvimento do sistema apresentado neste trabalho.

2.3.1. Sistemas Baseados em Agentes

O crescimento da pesquisa na área de agentes de software é motivado por se constituir em um paradigma apropriado para desenvolver aplicações para ambientes abertos, heterogêneos e distribuídos como a Internet.

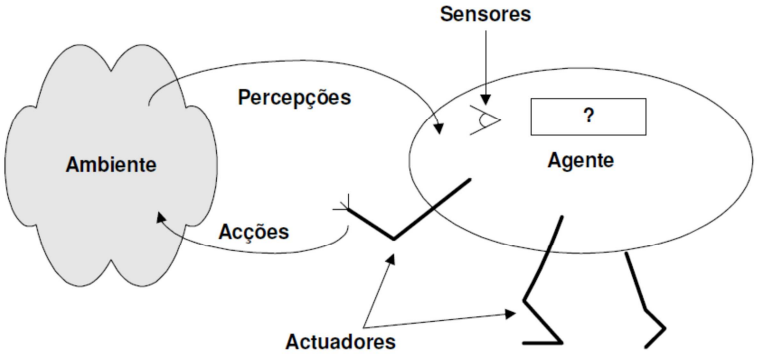
Segundo o AgentLink (LUCK *et al.*, 2005), que é uma comissão europeia que coordena ações baseadas em computação baseada em agentes, esta área de pesquisa é uma das mais vibrantes e importantes que surgiram na tecnologia da informação nos anos 90. No

Roadmap publicado em 2005 (LUCK *et al*., 2005) os pesquisadores destacam que as tecnologias de particular relevância que fornecem infraestrutura para o desenvolvimento de sistemas baseados em agentes são XML (*Extensive Markup Language*), RDF (*Resource Description Framework*) e *Web Services*. Destacam ainda que a Web Semântica, *Web Services* e SOA (*Service Oriented Architecture*), Sistemas complexos, entre outras, são tendências emergentes para aplicação de sistemas baseados em agentes.

Agentes são entidades de software situadas em um determinado ambiente, capazes de perceber eventos e atuar sobre o ambiente, que são projetados para atingir determinados objetivos através de mecanismo de raciocínio prático (WOOLDRIDGE, 2009).

A Figura 7 dá uma visão abstrata de um agente, onde pode-se observar que o mesmo capta informações sensoriais do ambiente e produz, como saída, ações que afetam este ambiente.

Figura 7 – Esquema típico de um agente de software



Fonte: RUSSEL e NORVIG (1995) *apud* REIS (2003)

Para Wooldridge e Jennings (2005, *apud* WOOLDRIDGE, 2009) um agente é um sistema computacional baseado em software que goza das seguintes propriedades:

- Autonomia: opera sem a intervenção direta de humanos ou outros agentes e possui algum tipo de controle sobre as suas ações e estado interno;
- Reatividade: tem a percepção do seu ambiente e responde rapidamente às alterações que nele ocorrem;
- Pró-Atividade: não se limita a agir em resposta ao seu ambiente sendo capaz de tomar iniciativa e exibir comportamento direcionado por objetivos;
- Habilidade Social: é capaz de interagir com outros agentes (e possivelmente com humanos) através de uma dada linguagem de comunicação de agentes.

A autonomia é provavelmente a característica mais consensual na definição do conceito de agente pelos pesquisadores da área (REIS, 2003). Autonomia refere-se ao princípio de que os agentes podem agir baseados nas suas próprias regras de decisão, sem existir a necessidade de serem guiados por humanos (NWANA, 1996 *apud* REIS, 2003). Os agentes possuem estados e metas internas, e agem de forma a atingir estas metas em prol dos seus utilizadores (WOOLDRIDGE e JENNINGS, 1995, *apud* WOOLDRIDGE, 2009).

Sistemas Multiagentes (SMA) são sistemas compostos por múltiplos agentes, que exibem um comportamento autônomo e ao mesmo tempo interagem com os outros agentes presentes no sistema. Estes agentes devem ser capazes de agir de forma autónoma tomando decisões levando à satisfação dos seus objetivos e de interagir com outros agentes utilizando protocolos de interação social inspirados nos humanos. Devem ainda, apresentar pelo menos algumas das seguintes funcionalidades: coordenação, cooperação, competição e negociação (REIS, 2003).

Quando se fala de arquitetura de agentes, há dois aspectos a serem considerados: a arquitetura interna de cada agente e também a arquitetura do sistema multiagente, ou seja, ao modo de organização dos agentes dentro de um sistema e como estão estruturados os seus relacionamentos e interações. Wooldridge e Jennings (1995 *apud* WOOLDRIDGE, 2009) afirmam que a arquitetura de um agente pode ser definida através de uma metodologia específica para defini-los, abrangendo técnicas e algoritmos para suportar essa mesma metodologia.

2.3.2. Ontologias

A Representação do Conhecimento é uma subárea da inteligência artificial que busca formalismos que possam ser usados para representar a informação a respeito do mundo real. Uma forma de representar o conhecimento sobre o mundo ou parte dele é por meio de ontologias.

O termo Ontologia tem origem na Filosofia onde trata do conhecimento do ser. Na Ciência da Computação, ontologias visam descrever os tipos de entidades existentes em um determinado domínio do conhecimento e como elas são relacionadas. Estas representações são formais, pois são manipuláveis por computadores, explicitas, pois são formas de representação, e compartilhadas, pois viabilizam a colaboração .

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 significado 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 *apud* SILVA, 2010).

O maior propósito do uso de ontologias não é apenas servir como vocabulário e taxonomias, mas sim, permitir o compartilhamento e o reuso do conhecimento por aplicações (SIDDIQUI e ALAM, 2010). Uma ontologia deve prover uma descrição de conceitos e relacionamentos existentes em um domínio. Estes elementos devem ser compartilhados e reusados por agentes e aplicações inteligentes.

McGuinness e Noy (2001) argumentam que não existe uma forma correta para modelar uma ontologia, que sempre existirão alternativas viáveis e que o desenvolvimento de ontologias é necessariamente um processo iterativo. As autoras sugerem os seguintes passos para o desenvolvimento de uma ontologia:

- Determinar o domínio e o escopo;
- Considerar reuso de ontologias existentes;
- Enumerar termos importantes;
- Definir classes e hierarquias de classes;
- Definir propriedades das classes;
- Definir as restrições das propriedades;
- Criar instâncias da ontologia.

Siddiqui e Alam (2010), por sua vez, propõe as seguintes atividades para criação de ontologias:

- Coleta: fase de coleta de documentos, normalmente dispostos em linguagem natural em arquivos digitais, para adquirir conhecimento sobre o domínio;
- Extração: fase para explorar o domínio e extrair os componentes da ontologia (aprendizado);
- Organização: fase para gerar a representação formal do conhecimento adquirido para ser usado por agentes;

- Fusão (*merging*): mapeamento de regras visando encontrar características comuns entre bases e derivar novas;
- Refinamento: melhorar, aumentar os níveis de granularidade;
- Recuperação: consiste na recuperação das informações por meio de agentes na Web Semântica.

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 através de estruturas de representação para classes, propriedades, restrições e indivíduos. 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 (SILVA, 2010).

Os principais elementos da OWL são:

- Classes: correspondem às classes raízes da ontologia.
- Subclasses: permitem definir a hierarquia da taxonomia através de generalizações/especializações.
- Indivíduos: são instâncias das classes.
- Propriedades: permitem definir fatos sobre classes e indivíduos. Representam relações entre instâncias de classes.
- Restrições: permitem definir como as propriedades podem ser utilizadas pelas instâncias de uma classe.

Para realizar consultas nas ontologias pode ser utilizada a linguagem SPARQL. Inspirada na *Structured Query Language* (SQL), SPARQL é uma linguagem que viabiliza a busca de informações em grafos RDF. Viabiliza consultas em modelos especificados através de OWL, facilitando a recuperação de indivíduos e propriedades de uma ontologia (W3C, 2008).

Para implementação de ontologias e consultas SPARQL existe o *framework* Jena, que é um *framework* Java para programação de aplicações para Web Semântica. Jena provê ambientes para programação de OWL, entre outras linguagens para representação de ontologias (JENA, 2010).

Uma característica importante do *framework* Jena, é disponibilização de interfaces para motores de inferência para OWL. 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 *OntModel* 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 *apud* SILVA, 2010).

3. 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.

Os principais trabalhos relacionados que serviram de inspiração para esta pesquisa e são brevemente citados a seguir. A Seção 3.1 apresenta o modelo da Riosoft (2009), seguida, na Seção 3.2, do modelo da Assespro-PR (2009), ambos modelos com mais foco comercial. A Seção 3.3 traz uma descrição do MEDEPROS® (COLOMBO e GUERRA, 2008) desenvolvido no CTI/MCT. Também foi pesquisado e é apresentado, na Seção 3.4 o modelo que se propõe a certificação de software público 5CQualiBr (SPB, 2009). Os próximos trabalhos relacionados estudados e apresentados são modelos mais especializados. Na Seção 3.5, um modelo baseando no MEDEPROS® e requisitos de um edital de licitação (MAITINGUER, 2004) e, na Seção 3.6, uma ferramenta que visa apoiar medições realizadas com o MEDEPROS® (GAMA e OLIVEIRA, 2011). A Seção 3.7 apresenta um modelo de avaliação de qualidade que utiliza números *fuzzy* para representar os atributos de qualidade (BELCHIOR, 1997). Na Seção 3.8 é descrito um *framework* para customização de modelos de qualidade baseado em necessidades do usuário (SIBISI e WAVEREN, 2007) sendo, este último, o único trabalho relacionado que não é brasileiro. Por fim, a Seção 3.9 apresenta um comparativo entre os trabalhos pesquisados.

3.1. 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 com o objetivo de fornecer uma credencial de qualidade que sirva de orientação ao mercado de produtos de software e, ao mesmo tempo, com baixo custo de avaliação, beneficiando a pequena empresa e viabilizando sua disseminação pelo país.

Neste modelo a qualidade do produto é avaliada em relação a atributos genéricos baseados em nas seguintes referências normativas:

- NBR ISO/IEC 9126-1: Engenharia de software - Qualidade de produto - Parte 1: Modelo de qualidade;
- NBR ISO/IEC 12119: Tecnologia de informação - Pacotes de software - Teste e requisitos de qualidade;
- NBR ISO/IEC 14598-1: Tecnologia de informação - Avaliação de produto de software - Parte 1: Visão geral;
- NBR ISO/IEC 14598-5 Tecnologia de informação - Avaliação de produto de software - Parte 5: Processo para avaliadores.

A avaliação se dá por meio da execução de testes previamente planejados pelo produtor do software e pela aplicação de listas de verificação. As empresas avaliadas com sucesso têm direito a um certificado de qualidade e ao registro no portal do PROIMPE, como uma forma de reconhecimento público de sua qualidade (RIOSOFT, 2009).

3.2. Teste OK

Outra instituição que desenvolveu um selo de qualidade para produtos de software foi a Assespro-PR (Associação das Empresas de Tecnologia da Informação, Software e Internet do Estado do Paraná), em parceria com a empresa ST Consultory. Teste OK! é um Selo de Certificação de Qualidade de Software com o objetivo de garantir e atestar que o software possui qualidade, sendo o mesmo aplicado em 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).

3.3. 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 derivados em atributos e métricas. As métricas, por sua vez, são questões pré-definidas que devem ser respondidas por meio de medições.

O 5CQualiBr, cujo nome deriva de “conhecimento, comunidade, colaboração, compartilhamento e confiança para qualidade do software público brasileiro”, é coordenado pelo Centro de Tecnologia da Informação Renato Archer (CTI/MCT). Trata-se de um projeto em vigência desde o final de 2008, que conta com recursos da Financiadora de Estudos e Projetos (FINEP) e apoio da Secretaria de Política de Informática do Ministério da Ciência e Tecnologia.

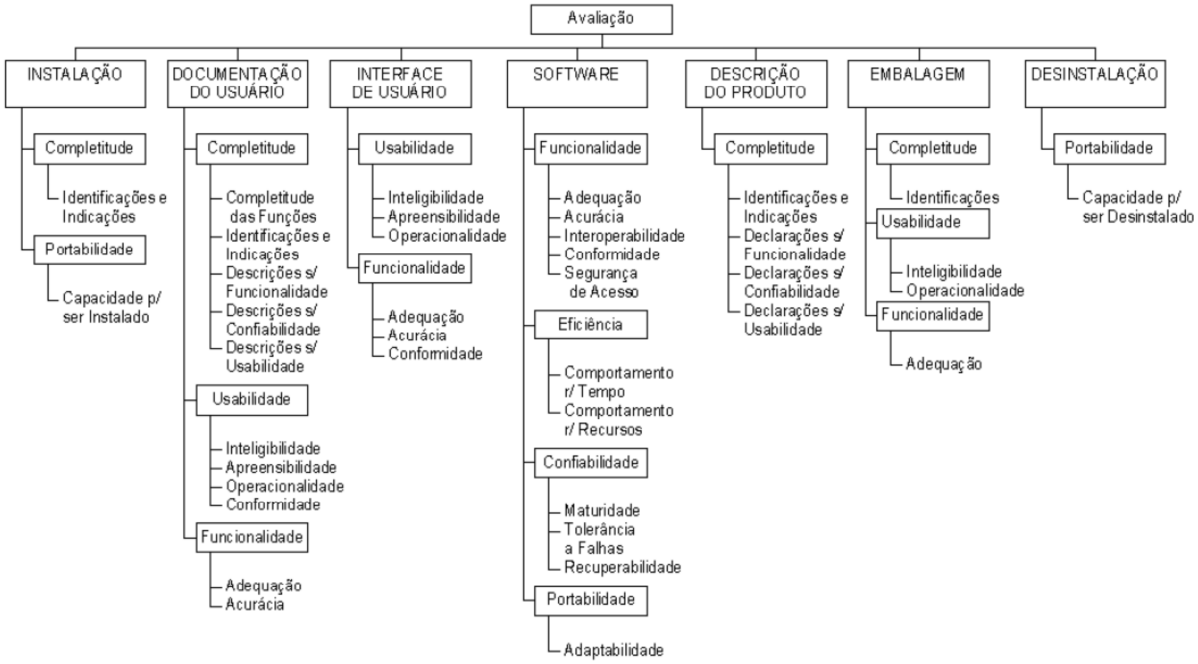
3.4. MEDEPROS®

O MEDEPROS® 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 avaliar o software com base no modelo de qualidade explícito nas normas NBR ISO/IEC 9126-1 (Modelo de Qualidade do Produto) e NBR ISO/IEC 12119 (Teste e requisitos de qualidade). Seu objetivo é Avaliar a qualidade de produto de software, segundo a visão do usuário, verificando o quanto ele está atendendo os padrões das normas internacionais de qualidade.

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, por sua vez, é relacionado a um conjunto de questões que compõem as Listas de Verificação (COLOMBO e GUERRA, 2008).

Na Figura 8 são exibidos os aspectos considerados pelo MEDEPROS® e os atributos da qualidade relacionados a cada um deles.

Figura 8 – Estrutura de atributos da qualidade do MEDEPROS®



Fonte: COLOMBO e GUERRA (2008)

O método MEDEPROS® foi aplicado no Prêmio ASSESPRO, que visava destacar produtos com maior nível de qualidade na indústria de software, na chamada nacional SOFTEX, que visa selecionar projetos de empresas brasileiras exportadoras de produtos de software para conceder linhas de créditos, e na seleção de software para o PNAFAM (Programa Nacional de Apoio à Gestão Administrativa e Fiscal dos Municípios Brasileiros) (COLOMBO e GUERRA, 2008).

3.5. Modelo especialista baseado em requisitos especificados em edital

Este método é baseado no MEDEPROS® (COLOMBO e GUERRA, 2008), citado anteriormente, e em requisitos especificados em um edital. Edital é um tipo de documento conhecido como “instrumento convocatório” utilizado em processos de licitação. Como objeto de pesquisa acadêmica (MAITINGUER, 2004), 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), o qual visava pré-qualificar fornecedores de software para participação no programa.

Segundo Maintinguer (2004), o modelo seguiu a estrutura de listas de verificação proposta no modelo MEDEPROS® para a avaliação dos requisitos não funcionais. Para a avaliação dos requisitos funcionais, a proposta vale-se de um Guia de Avaliação, o qual é composto, além dos atributos de qualidade relacionados com o requisito da aplicação que foram classificados como obrigatórios, desejáveis ou recomendados conforme o edital, 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.

3.6. 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*) que é um projeto da UFPA (Universidade Federal do Pará) que visa 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).

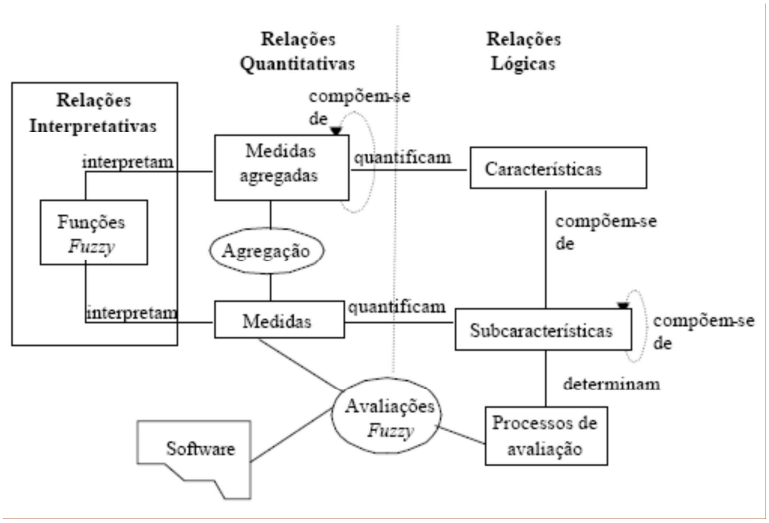
Até o momento desta pesquisa, a ferramenta SPIDER-PQ inda estava em fase de homologação em laboratório e não estava publicada no site do projeto SPIDER.

3.7. Modelo *fuzzy* de avaliação de qualidade de software

Belchior (1997) propôs um modelo de avaliação da qualidade de software baseado em lógica *fuzzy*, o qual foi base para uma ferramenta chamada *AdeQuaS*, implementada em projeto de pesquisa por Oliveira (2002).

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 produto (ou do domínio de aplicação) e expressado 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 9126-1, por meio da defuzificação (BELCHIOR, 1997). A Figura 10 apresenta a estrutura do MFAQS.

Figura 9 - Modelo *Fuzzy* para Avaliação da Qualidade de Software



Fonte: BELCHIOR (1997)

O processo de avaliação do MFAQS é composto de 5 etapas:

- *Etapa 1*: identificação do objeto a ser avaliado, do conjunto de atributos de qualidade de software a ser considerado na avaliação e das instituições pesquisadas;
- *Etapa 2:* escolha dos especialistas, com identificação dos perfis e dos níveis de importância da avaliação de cada um deles (pesos);

- *Etapa 3:* determinação do grau de importância de cada atributo de qualidade que foi identificado na primeira etapa. Nesta fase também são definidos os procedimentos de investigação, como questionários, por exemplo;
- *Etapa 4:* tratamento dos dados coletados dos especialistas, na avaliação de cada atributo de qualidade mensurável considerado;
- *Etapa 5:* agregação dos atributos de qualidade de software, em cada nível hierárquico do modelo de qualidade.

Os valores linguísticos que representam a relevância dos atributos identificados na etapa 1, são os apresentados na Tabela 1.

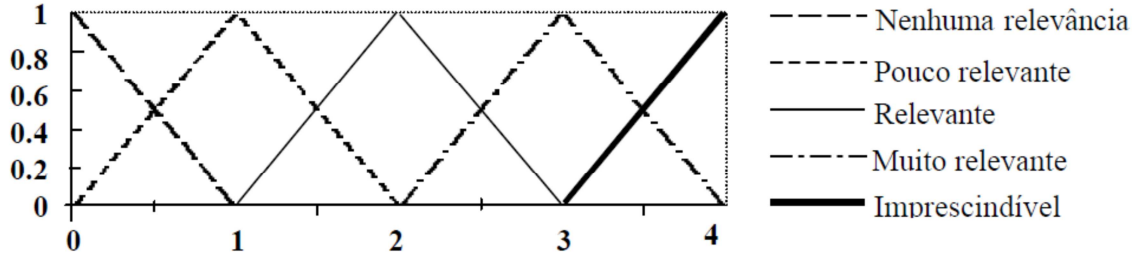
Tabela 1 – Valores linguísticos para a relevância dos atributos da qualidade

Nota	Número *Fuzzy* Normal	Termo Linguístico	Interpretação
0	Ñ1 = (0,0; 0,0; 1,0)	Nenhuma relevância	Indica de maneira absoluta que a existência do atributo de qualidade não tem importância
1	Ñ2 = (0,0; 1,0; 2,0)	Pouco relevante	Indica de maneira absoluta que a existência do atributo de qualidade tem pouca importância
2	Ñ3 = (1,0; 2,0; 3,0)	Relevante	Indica de maneira absoluta que é desejável a existência do atributo de qualidade
3	Ñ4 = (2,0; 3,0; 4,0)	Muito relevante	Indica de maneira absoluta que a existência do atributo de qualidade é muito importante
4	Ñ5 = (3,0; 4,0; 5,0)	Imprescindível	Indica de maneira absoluta que a existência do atributo de qualidade é imprescindível

Fonte: BELCHIOR (1997)

Para identificar os valores linguísticos da Tabela 1, é utilizada a função de pertinência apresentada na Figura 11.

Figura 10 – Função de pertinência para identificar os valores linguísticos



Fonte: BELCHIOR (1997)

O processo de agregação é realizado para cada nível da hierarquia de atributos, resultando, ao final do nível mais alto, em um valor correspondente ao grau de qualidade do objeto avaliado. Se a avaliação envolver a utilização de um padrão de qualidade (PQ) para apoiar a avaliação, o valor obtido do atributo é confrontado com o PQ e calculado o grau de adequação do produto a esse PQ. Isto resulta na determinação do Índice de Qualidade (IQ) de software.

3.8.*Framework* de processo para customizar modelos de qualidade de software

Pesquisadores da Universidade de Pretória na África do Sul (SIBISI e WAVEREN, 2007) propuseram um *framework* de processo que visa a customização de modelos de qualidade por meio do relacionamento dos atributos de modelos tais como NBR ISO/IEC 9126-1, às necessidades do usuário.

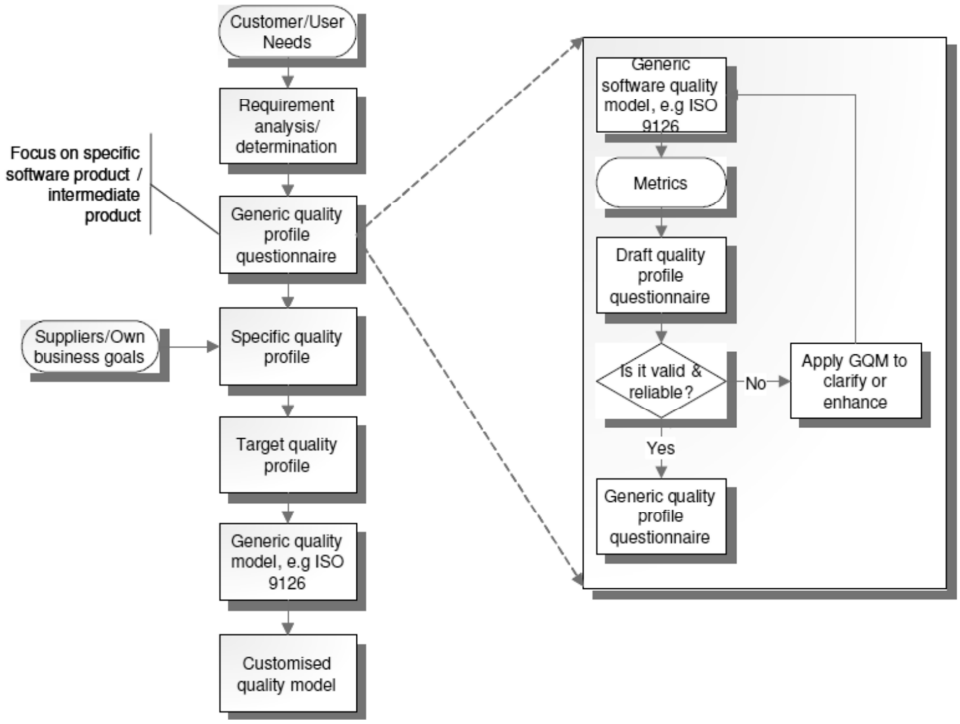
A proposta dos autores preconiza a identificação de metas da qualidade, conforme Figura 9. Segundo eles, uma meta de qualidade é qualidade necessária e suficiente que reflete as necessidades do usuário. Não é, necessariamente, a qualidade perfeita, mas sim, a qualidade que permite que o usuário atinja os seus objetivos. É proposto, ainda, o uso do paradigma GQM (BASILI, CALDIERA e ROMBACH, 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 visão do adquirente), ou analistas e engenheiros de software (para o ponto de vista do desenvolvimento). Visa 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.

Figura 11 – *Framework* para customizar modelos de qualidade de software



Fonte: SIBISI e WAVEREN (2007)

3.9. Análise dos trabalhos relacionados

Esta seção apresenta uma análise comparativa entre os trabalhos relacionados pesquisados e o sistema de apoio à certificação de qualidade de produto de software que é proposto neste trabalho.

Para a comparação foral elencados critérios os quais julga-se relevantes no contexto e motivação desta pesquisa, 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*.

A Tabela 2 traz um consolidado com os critérios que são considerados atendidos para cada um dos trabalhos relacionados citados neste capítulo.

Tabela 2 – Comparativo dos trabalhos relacionados

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	

Fonte: Elaborado pela autora

Observa-se que, apesar de todos os trabalhos pesquisados apoiarem-se em referências normativas internacionais, para maioria dos trabalhos pesquisados o modelo de qualidade do produto utilizado é genérico. A pesquisa de Maitinguer (2004) foi o primeiro trabalho que se propôs a apresentar 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 são todos tratados dentro do atributo Funcionalidade, e os requisitos não funcionais, que também podem apresentar subcaraterísticas e níveis de importância diferenciados para diferentes domínios de aplicação, são baseados no método genérico MEDEPROS®.

A ferramenta SPIDER-PQ, que visa apoiar avaliações com MEDEPROS®, 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 apresentarem um processo para especialização do modelo que 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), 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 tecnologias de ontologias e agentes de software.

4. *FRAMEWORK* PARA CERTIFICAÇÃO DE QUALIDADE DE PRODUTOS DE SOFTWARE

Neste Capítulo é apresentado o *framework* de processo em cujos conceitos se baseia o sistema de suporte à certificação de qualidade de produtos de software desenvolvido neste trabalho e apresentado na sequencia, no Capítulo 5. As Seções 4.1, 4.2 e 4.3 trazem, respectivamente, a descrição dos subprocessos de especialização do modelo de qualidade, de medição da qualidade e de avaliação da qualidade.

Um *framework*, na língua inglesa, é definido como um conjunto de crenças, ideias ou regras que é usado como base para julgamentos, tomada de decisões, etc. (OXFORD, 2000). No contexto deste trabalho, foi concebido um *framework* conceitual que organiza um conjunto de conceitos sobre certificação de software, os relacionamentos e a sequência lógica entre estes conceitos, permitindo um nível de abstração.

O modelo é decomposto em três subprocessos, conforme Figura 12, e sua principal característica é ser focado nos riscos e requisitos do domínio da aplicação .

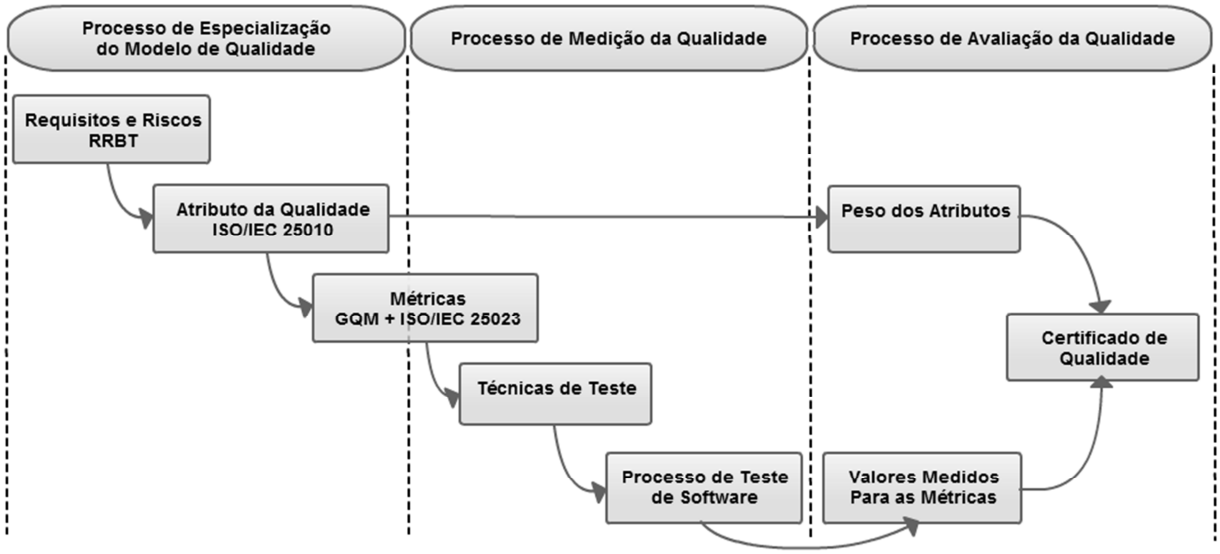
Figura 12 - *Framework* para certificação de qualidade de produtos de software

Fonte: Elaborado pela autora

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

Uma visão geral do *framework* proposto com as atividades de cada subprocesso é apresentada na Figura 13.

Figura 13 - Processos do *framework* para certificação de qualidade de produtos de software



Fonte: Elaborado pela autora

4.1.Subprocesso de especialização do modelo de qualidade

O subprocesso de especialização do modelo de qualidade 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.

Para o *Passo 1*, a proposta é que, a partir de um conjunto de requisitos do sistema, seja aplicada a técnica RRBT. Esta técnica preconiza que os requisitos sejam relacionados aos riscos do produto de modo que o esforço em testes seja focado nos principais riscos (PINKSTER, BURGT e VEENENDAAL, 2006).

É um pré-requisito para esta etapa que exista um conjunto de requisitos especificados para a aplicação e uma análise de riscos de negócio. Vale destacar, neste momento, que não faz parte do escopo do *framework* proposto as atividades de análise de requisitos e de análise de riscos. No contexto deste trabalho, realiza-se a associação entre um conjunto de requisitos e de riscos pré-existentes.

Associados os devidos riscos aos requisitos, no *Passo 2* são associados os atributos da qualidade, segundo a ISO/IEC 25010, a cada risco de negócio, para completar o modelo especialista. Deve ser relacionado a cada risco, o atributo da qualidade que melhor represente as características do risco.

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 atributos de 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.

No *Passo 4*, para apoiar a definição das metas de qualidade e seleção de métricas correspondentes, é proposto o método GQM (*Goal Question Metric*).

Para aplicação do GQM, segundo este *framework*, a definição das metas ou objetivos (*goals*) deve se dar por meio da identificação dos propósitos com base nos riscos já associados aos requisitos. As questões (*questions*) podem ser baseadas na ISO/IEC 25023, pois esta parte da norma apresenta o propósito de medição em forma de questão. Da mesma forma, as métricas (*metrics*) também podem ser baseadas nas próprias métricas apresentadas na ISO/IEC 25023.

O modelo de qualidade especializado é considerado concluído quando foram definidas as métricas que satisfaçam os atributos da qualidade, conforme os riscos e requisitos da aplicação.

4.2.Subprocesso de medição da qualidade

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 o *framework* prevê dois passos:

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

No *Passo* 5, à cada métrica deve-se relacionar a técnica de testes que permitirá coletar os dados necessários para medição. A medida pode ser entendida como o objetivo e a técnica de teste como 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 dos testes, 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.

As atividades de planejamento, projeto e execução dos testes não fazem parte do escopo deste trabalho. Considera-se que o subprocesso de medição da qualidade é concluído quando são obtidos os valores medidos das métricas.

4.3.Subprocesso de avaliação da qualidade

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 o valor e os pesos dos atributos.

No *Passo 7* as medidas obtidas devem comparadas com valores de referência para cada uma das métricas.

Um coeficiente de aderência deve ser calculado para cada métrica, de modo que represente o quanto o valor medido da mesma é aderente ao seu 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 nas fórmulas (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 ou na especificação de requisitos.

A expressão (1a) se aplica para os casos em que RV = 1, ou seja, “quanto maior e mais próximo de 1 melhor”, conforme a interpretação recomendada da norma *SQuaRe* (ISO/IEC 25023). Aplica-se para métricas que avaliam quantidades de eventos evitados ou controlados, como falhas corrigidas ou casos de testes que encontraram erros, por exemplo.

Já a expressão (1b) deve ser utilizada nos casos em que RV = 0. São casos de métricas cuja interpretação recomendada da norma *SQuaRe* é “quanto menor e mais próximo de zero melhor”, e representam quantidades observadas de eventos como quedas no sistema ou corrupção de dados.

A expressão (1c), por sua vez, faz-se necessária para os casos em que o RV > 0. São casos de métricas cuja interpretação recomendada da norma *SQuaRe* é “quanto menor e mais próximo de zero melhor”, porém, não terão valor igual a zero, pois a unidade de medida é tempo. Aplica-se a medidas de comportamento em relação ao tempo e, no contexto deste trabalho, o RV deve ser especificado junto aos requisitos da aplicação.

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 determinado 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 (2) abaixo:

$$AV = \sum_{i=1}^{n} (MA * MW) \quad (2)$$

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 7;
- *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 determinados para cada atributo, multiplicados por seus respectivos pesos. A fórmula de cálculo está representada pela equação (3):

$$QL = \sum_{i=1}^{m} (AV * AW) \quad (3)$$

Onde,

- *QL - Quality Level*: nível de qualidade do produto;
- *AV - Attribute Determined Value*: Valor determinado do atributo obtido no passo 8;
- *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.

5. SISTEMA DE APOIO À CERTIFICAÇÃO DE QUALIDADE DE PRODUTOS DE SOFTWARE

Neste Capítulo é apresentado o protótipo desenvolvido para um sistema baseado em tecnologias de ontologias e de agentes para dar suporte à certificação de qualidade de produtos de software. O sistema segue os subprocessos conforme o *framework* para certificação de qualidade de produtos de software apresentado na seção anterior.

Na Seção 5.1 é apresentada uma visão geral do sistema desenvolvido. A Seção 5.1 descreve o desenvolvimento da ontologia do certificado de qualidade. Na Seção 5.3 é apresentado o desenvolvimento do agente do agente de certificação da qualidade e na Seção 5.4 é descrito o protótipo de interfaces gráficas.

5.1.Visão Geral do Sistema

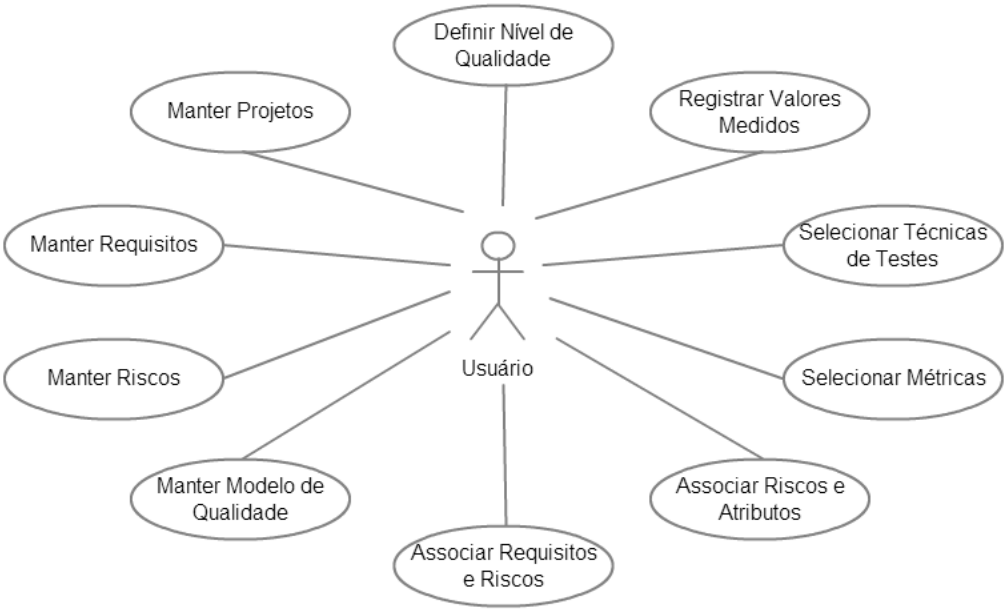
O objetivo central do sistema desenvolvido é apoiar a realização dos passos do *framework* que permitem relacionar os atributos da qualidade e inferir as métricas de modo a especializar o modelo e, a partir de valores mensurados para as métricas, classificar o produto calculando seu nível de qualidade.

Na busca por estes objetivos, utilizou-se de ontologias para mapear os conceitos e relacionamentos de modo a representar o conhecimento que envolve a especialização do modelo, a medição e a avaliação de qualidade de software. A opção por um modelo baseado em ontologias, ao invés de um modelo baseado em dados, neste caso, justifica-se devido às possibilidades de trabalhos futuros alinhados às tendências da web semântica.

Os agentes de software, por sua vez, são utilizados para manipular os indivíduos na ontologia viabilizando inferir as métricas para as quais serão coletadas medidas para posterior avaliação, bem como realizar os cálculos que viabilizam classificar e certificar o produto. No contexto deste trabalho, a opção pela tecnologia de agentes de software justifica-se pelas características já citadas de sistemas orientados a gentes, tais como, autonomia e pró-atividade que viabilizam aplicações de aquisição de conhecimento em trabalhos futuros.

O sistema desenvolvido caracteriza-se pelas funcionalidades apresentadas no diagrama de contexto, representado por meio de um diagrama de casos de uso na Figura 14.

Figura 14 - Diagrama de contexto do sistema



Fonte: Elaborado pela autora

Os casos de uso ‘Manter Projetos’, ‘Manter Requisitos’ e ‘Manter Riscos’, fazem parte da funcionalidade que visa permitir a manutenção dos cadastros necessários para a especialização e certificação do modelo de qualidade. Da mesma forma, o caso de uso ‘Manter Modelo de Qualidade’ permite a manutenção dos cadastros de atributos da qualidade, subscaracterísicas e métricas conforme o modelo de referência escolhido, no caso desta pesquisa a *SQuaRe* ISO/IEC 25010.

Por meio dos casos de uso ‘Associar Requisitos e Riscos’, ‘Associar Riscos e Atributos’ e ‘Selecionar Métricas’, o sistema visa apoiar a especialização do modelo de qualidade, conforme os passos de 1 a 4 do *framework*. Neste caso, a função do agente de software é apoiar a manipulação das ontologias montando as consultas SPARQL necessárias e inferindo as métricas.

A medição da qualidade é realizada pelo sistema por meio dos casos de uso ‘Selecionar Técnicas de Testes’ e ‘Registrar Valores Medidos’ onde são realizados os passos 5 e 6 do *framework*. Neste último, o usuário registra os resultados obtidos com os testes realizados na aplicação que está sendo avaliada.

No caso de uso ‘Definir Nível de Qualidade’ acontecem os passos 7, 8 e 9 do *framework*, onde o sistema deve, com apoio de um agente de software, calcular automaticamente o coeficiente de aderência de cada métrica, o valor de cada atributo da qualidade e, por fim, o nível de qualidade do projeto.

5.2.Ontologia do Certificado de Qualidade

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

Os primeiros conceitos identificados para a Ontologia do Certificado de Qualidade foram os relativos ao Modelo de Qualidade, os quais devem permitir mapear os relacionamentos entre requisitos, riscos, atributos e subcaracterísticas da qualidade. Esta ontologia deve ser populada a partir de uma base de requisitos do sistema sob verificação seus respectivos riscos de negócio, além de atributos, subcaracterísticas e métricas 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.

Algumas ontologias relacionadas com o domínio especificado foram encontradas da literatura, tais como a pesquisa de Duarte e Falbo (2000) que propõe uma ontologia de qualidade, que se relaciona com a ontologia de processo de software (FALBO, 1998 *apud* DUARTE e FALBO, 2000). Porém, não foram encontrados, nesta ontologia, termos que se adequassem aos conceitos de riscos e requisitos que são centrais no *framework* para certificação de qualidade de produtos de software.

Gusmão *et al*. (2004) propõem uma ontologia de riscos para ambientes de desenvolvimento de software apoiada na taxonomia de riscos do SEI (CARR, 1993 *apud* GUSMÃO *et al*., 2004). Sendo esta ontologia fortemente voltada para a identificação e análise de riscos, atividades que estão fora do escopo deste trabalho, a mesma não se mostra adequada para o objetivo do sistema proposto.

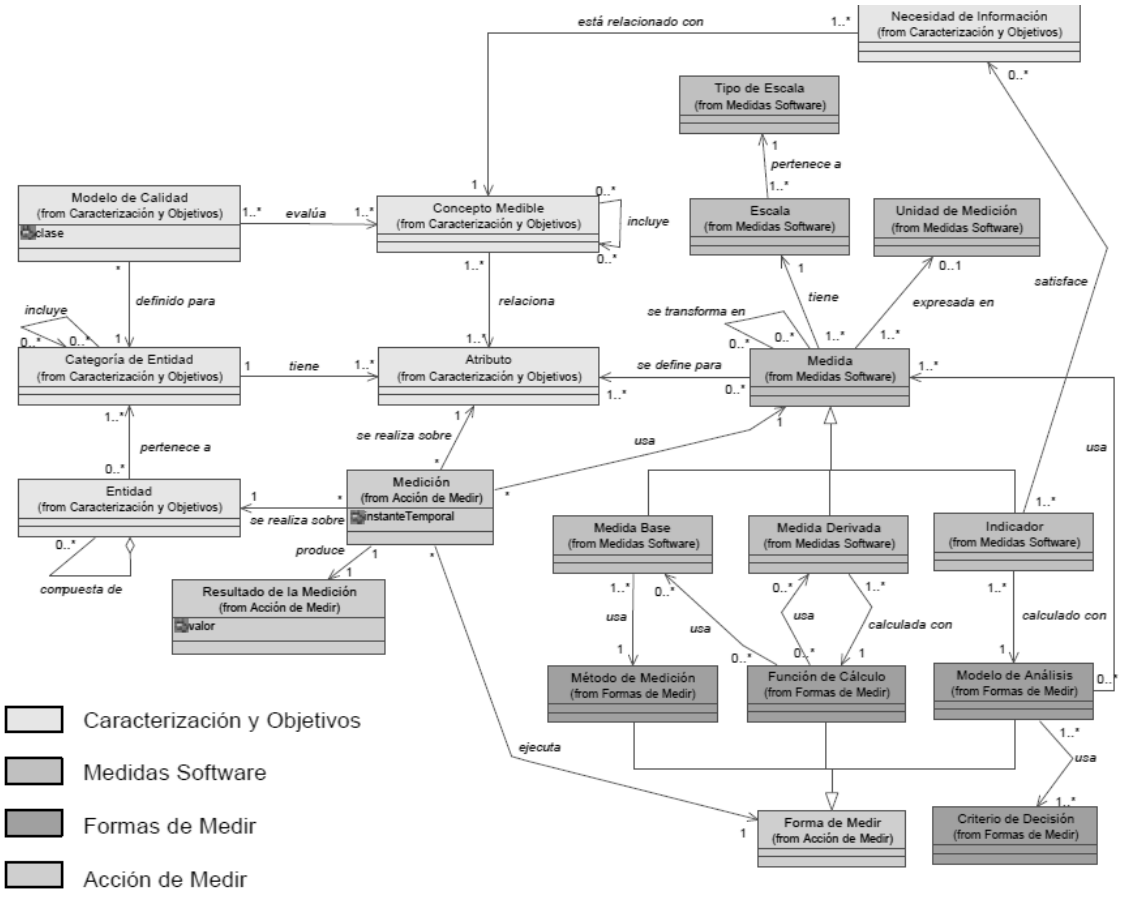
Uma ontologia para medição de software foi proposta por Ferreira *et al*. (2006). Os pesquisadores da Universidade de Castilla-La Mancha (FERREIRA *et al*., 2006), propõem uma Ontologia de Medição de Software inspirada em pesquisa anterior de Garcia *et al*. (2005 *apud* FERREIRA *et al*., 2006) na qual foi realizada uma análise comparativa dos conceitos e termos utilizados nas normas internacionais mais relevantes no campo da medição de software, identificando os pontos comuns e as diferenças, e apresentando uma proposta unificada para uma terminologia comum para medição de software.

A ontologia comum proposta por Ferreira *et al*. (2006) é subdividida em quatro subontologias, a saber:

- **Subontologia da caracterização e objetivos da medição de software**, que inclui os conceitos necessários para estabelecer o escopo e os objetivos do processo de medição;
- **Subontologia de medidas de software**, que estabelece os elementos chave na definição de uma medida de software;
- **Subontologia das formas de medir**, que introduz o conceito de forma de medir para generalizar os diferentes métodos utilizados pelos diferentes tipos de medidas para obter os respectivos resultados de medição;
- **Subontologia de medição**, que estabelece a terminologia relacionada com o ato de medir o software.

Na Figura 15 é apresentada a ontologia de medição de software proposta por Ferreira *et al*. (2006), com a integração das subontologias, exibindo seus termos, atributos e relacionamentos originais em Língua Espanhola.

Figura 15 - Ontologia de Medição de Software



Fonte: FERREIRA *et al*., 2006

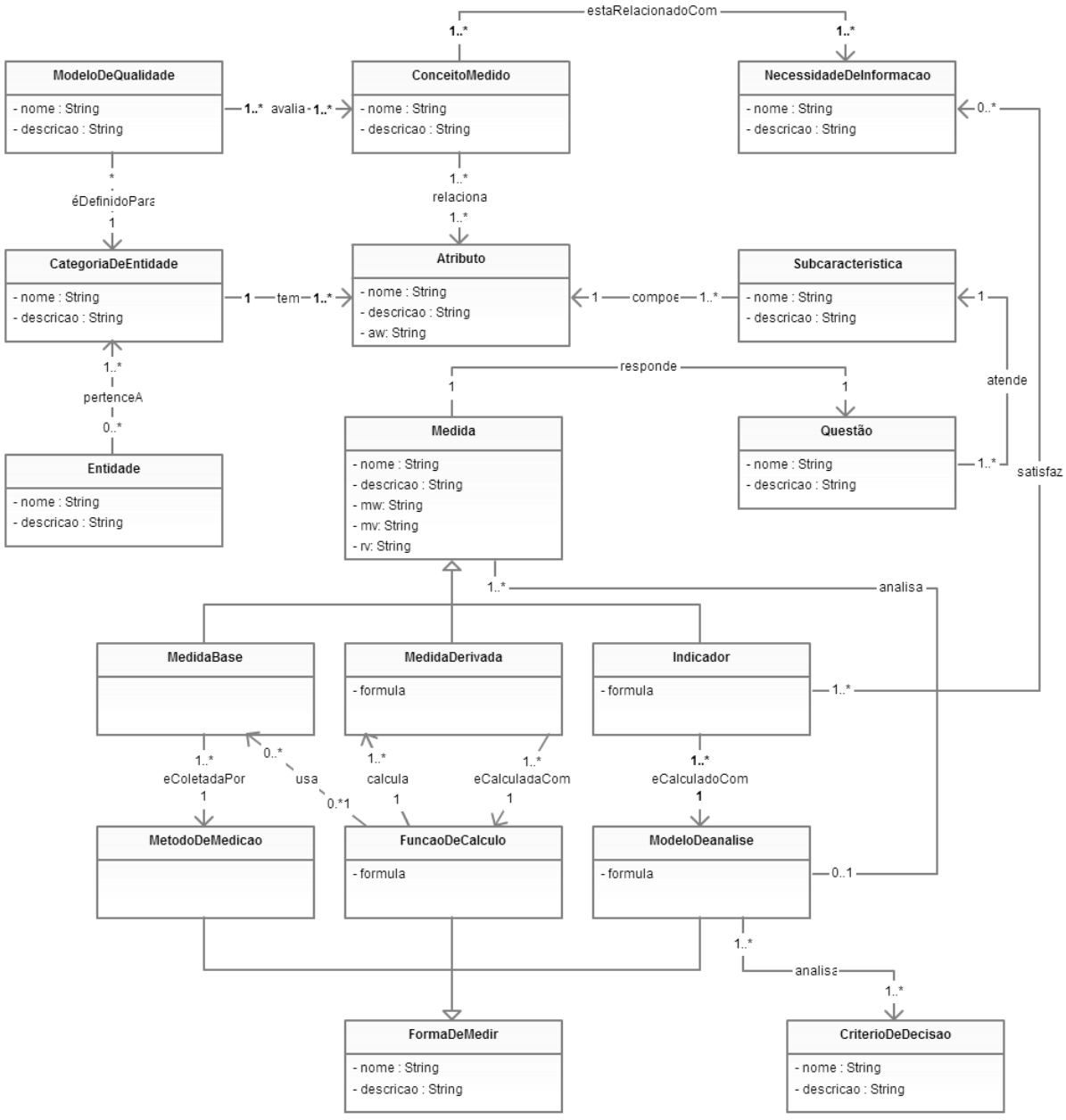
A ontologia de medição de software apresentada pelos pesquisadores espanhóis mostrou-se adequada para representar os termos do *framework* para certificação de qualidade de produtos de software, principalmente porque integra as medidas de software com um modelo de qualidade e as necessidades de informação 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 vem 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). As subontologias de “medida” e da “ação de medir”, foram deixadas de fora do escopo deste trabalho pois verificou-se que seus termos tratam de detalhamentos das medidas e dos resultados de medição, os quais avaliou-se como não sendo fundamentais para a certificação de software, apesar não serem incompatíveis.

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 Subcaracterítica da qualidade (que compõe um Atributo) e de Questão (que é respondida por uma Medida).

A Figura 16, 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).

Figura 16 - Ontologia do Certificado de Qualidade



Fonte: Adaptado de FERREIRA *et al*., 2006

Na Tabela 3, é apresentado um dicionário de termos para as Classes definidas para a ontologia do certificado de qualidade, indicando o termo original correspondente na ontologia de Ferreira *et al*. (2006).

Tabela 3 - Dicionário de termos para as Classes da Ontologia do Certificado de Qualidade

Classe	Termo original (FERREIRA *et al.*, 2006)	Descrição
NecessidadeDeInformacao	*Necessidad de Información*	Informação necessária para especializar o modelo de qualidade. No *framework* representa o **Requisito**.
ConceitoMedido	*Concepto Medible*	Relação entre Atributo e Necessidade de Informação e representa o **Risco**.
Atributo	*Atributo*	Propriedade de uma categoria de entidade que representa o **Atributo da Qualidade**.
Subcaracteristica	-	Representa a **Subcaracterística** da qualidade, que compõe um Atributo.
Questao	-	Representa a **Questão**, a qual é respondida por uma métrica.
ModeloDeQualidade	*Modelo de Calidad*	Conjunto de conceitos mensuráveis que fornecem uma base de atributos da qualidade. No *framework* representada a *SQuaRe* ISO/IEC 25010.
CategoriaDeEntidade	*Categoria de Entidad*	Coleção de entidades com características comuns. No contexto deste trabalho é **produto de software**, pois o mesmo é específico.
Entidade	*Entidad*	No *framework* é o **projeto** da categoria produto de software que vai ser avaliado e certificado.
Medida	*Medida*	Generalização dos diferentes tipos de Métricas.
MedidaBase	*Medida Base*	Especialização de Medida que representa as **Medidas Base** que são usadas nos cálculos das demais medidas.
MedidaDerivada	*Medida Derivada*	Especialização de Medida que representa as **Métricas** que mensuram as subscaracterístcas da qualidade.
Indicador	*Indicador*	Especialização de Medida que representa as **Métricas** que mensuram os valores dos atributos e o nível de qualidade do projeto.
FormaDeMedir	*Forma de Medir*	Generalização de diferentes tipos formas de mensurar ou calcular os resultados das medidas.
MetodoDeMedicao	*Método de Medición*	Especialização de FormaDeMedir que representa a **Técnica de Teste**, que é a forma de medir uma Medida Base.
FuncaoDeCalculo	*Función de Cálculo*	Especialização de FormaDeMedir que representa as **fórmulas** que calculam medidas derivadas fazendo uso de medidas base.
ModeloDeAnalise	*Modelo de Análisis*	Especialização de FormaDeMedir que representa as **fórmulas** que calculam indicadores fazendo uso de critérios de decisão.
CriterioDeDecisao	*Criterio de Decisión*	Representam **valores de referência** para cálculo dos indicadores.

Fonte: Elaborada pela autora

Definidos os conceitos que foram representados como Classes na ontologia do certificado de qualidade, foram identificadas as propriedades de cada conceito. Na Tabela 4, é apresentado um dicionário de termos para as propriedades do tipo objeto que representam os relacionamentos entre os elementos, que foram definidos para a ontologia do certificado de qualidade. Na Tabela 4 é apresentado, ainda, o termo original correspondente na ontologia de Ferreira *et al*. (2006).

Tabela 4 - Dicionário de termos para as propriedades tipo objeto para a Ontologia do Certificado de Qualidade

Propriedade	**Termo original (FERREIRA *et al*., 2006)**	**Domínio**	**Tipo**
estaRelacionadoCom	*está relacionado con*	ConceitoMedido	NecessidadeDeInformacao
relaciona	*relaciona*	ConceitoMedido	Atributo
avalia	*evalúa*	ModeloDeQualidade	ConceitoMedido
éDefinidoPara	*definido para*	ModeloDeQualidade	CategoriaDeEntidade
tem	*tiene*	CategoriaDeEntidade	Atributo
pertenceA	*pertence a*	Entidade	CategoriaDeEntidade
compõe	-	Subcaracteristica	Atributo
éComposto	-	Atributo	Subcaracteristica
atende	-	Questao	Subcaracteristica
éAtendidaPor	-	Subcaracteristica	Questao
responde	-	Medida	Questao
éRespondidaPor	-	Questao	Medida
éColetadaPor	*usa*	MedidaBase	MetodoDeMedicao
mede	-	MetodoDeMedicao	MedidaBase
usa	*usa*	FuncaoDeCalculo	MedidaBase
éCalculadaCom	*calculada com*	MedidaDerivada	FuncaoDeCalculo
calcula	*usa*	FuncaoDeCalculo	MedidaDerivada
analisa	*usa*	ModeloDeAnalise	Medida CriterioDeDecisao
éCalculadoCom	*calculado com*	Indicador	ModeloDeAnalise Indicador
satisfaz	*satisface*	Indicador	NecessidadeDeInformacao

Fonte: Elaborada pela autora

As propriedades tipo dado são aquelas que representam atributos dos indivíduos e complementam a descrição de determinado conceito. A Tabela 5 apresenta um dicionário de termos para as propriedades tipo dados que foram definidas para a ontologia do certificado de qualidade.

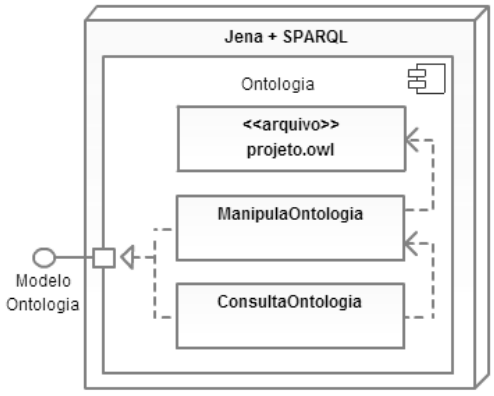
Tabela 5 - Dicionário de termos para as propriedades tipo dado da Ontologia do Certificado de Qualidade

Propriedade	Domínio	Tipo	Descrição
nome	*Thing*	*String*	Nome do indivíduo. Todos os indivíduos na ontologia tem nome.
descrição	*Thing*	*String*	Descrição do indivíduo. Todos os indivíduos na ontologia tem descrição.
aw	Atributo	*String*	Peso do atributo (*Attribute Weight*) no modelo de qualidade.
mw	Medida	*String*	Peso da medida derivada (*Metric Weight*) no atributo.
mv	Medida	*String*	Valor medido (*Measured Value*) para as medidas base e calculado para as medidas derivadas e indicador.
rv	Medida	*String*	Valor de referência (*Reference Value*) para as medidas derivadas.
formula	MedidaDerivada Indicador ModeloDeAnalise FuncaoDeCalculo	*String*	Formula de cálculo para medidas derivadas, indicadores, modelos de análise e funções de cálculo.

Fonte: Elaborada pela autora

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 com por meio do *framework* Jena, fazendo uso das bibliotecas SPARQL. Conforme apresentado na Figura 17, a estrutura Ontologia encapsula estas funcionalidades. 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.

Figura 17 - Arquitetura do componente Ontologia



Fonte: Elaborada pela autora

A recuperação de indivíduos na ontologia se dá por meio de consultas SPARQL, para as quais o Jena possui os recursos necessários. Para validar a capacidade da ontologia de representar o conhecimento do domínio especificado para esta proposta, foram identificadas algumas questões chave e realizadas consultas visando obter as devidas respostas.

A Figura 18 apresenta uma consulta SPARQL que responde a questão “quais são os Atributos relacionados a cada Risco e as Subcaracterísticas que os compõe?”. Neste caso são recuperados os indivíduos das classes NecessidadeDeInformação, ConceitoMedido, Atributo e Subcaracteristica onde ConceitoMedido ‘relaciona’ Atributo e ‘estaRelacionadoCom’ NecessidadeDeInformacao e Subcaracteristica ‘compoe’ Atributo.

Figura 18- Consulta que retorna Riscos, Atributos e Subscaraterísticas

```
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"

Fonte: Elaborada pela autora

A Figura 19 traz a consulta que retorna os indivíduos das classes ConceitoMedido, Subcaracteristica e Questao que respondem a questão “quais as Questões que atendem as Subcaracterísticas relacionadas a cada Risco?”. Onde, ConceitoMedido ‘relaciona’ Atributo, Subcaracteristica ‘compoe’ Atributo e Questao ‘atende’ Subcaracterística.

Figura 19 – Consulta que retorna Questões e Subcaracterísticas de cada Risco

```
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 ?Conceito_nome ?Atributo_nome
       ?Subcaracteristica_nome ?Questao_descricao
 WHERE { ?ConceitoMedido ont:relaciona ?Atributo .
         ?Subcaracteristica ont:compoe ?Atributo .
         ?Questao ont:atende ?Subcaracteristica .
         ?ConceitoMedido ont:nome ?Conceito_nome .
         ?Atributo ont:nome ?Atributo_nome .
         ?Subcaracteristica ont:nome ?Subcaracteristica_nome .
         ?Questao ont:descricao ?Questao_descricao }
ORDER BY ?Conceito_nome ?Atributo_nome
```

Conceito_nome	Atributo_nome	Subcaracteristica_nome	Questao_descricao
"RISCO01"	"Confiabilidade"	"Disponibilidade"	"Quão disponível é o sistema para uso durante um periodo de tempo especifico?"
"RISCO01"	"Confiabilidade"	"Disponibilidade"	"Qual é o tempo médio em que o sistema fica indisponível quando uma falha ocorre, antes da inicialização?"
"RISCO01"	"Confiabilidade"	"Maturidade"	"Qual a proporção de falhas detectadas que foram corrigidas?"
"RISCO01"	"Confiabilidade"	"Recuperabilidade"	"Qual o tempo médio que o sistema leva para completar a recuperação desde o início?"
"RISCO01"	"Confiabilidade"	"Tolerancia_a_falhas"	"Quantas vezes o produto de software causa a queda de todo o ambiente de produção?"
"RISCO01"	"Confiabilidade"	"Tolerancia_a_falhas"	"Quantos padrões de faltas são mantidos sob controle para evitar falhas criticas e sérias?"
"RISCO02"	"Eficiencia"	"Utilizacao_de_recursos"	"Qual é o limite absoluto de transmissões necessárias para cumprir uma função?"
"RISCO02"	"Eficiencia"	"Utilizacao_de_recursos"	"O sistema é capaz de desempenhar tarefas dentro da capacidade de transmissão esperada?"
"RISCO03"	"Seguranca"	"Confidencialidade"	"Quão controlável é o acesso ao sistema?"
"RISCO03"	"Seguranca"	"Integridade"	"Qual é a frequência de eventos de corrupção de dados?"
"RISCO03"	"Seguranca"	"Responsabilidade"	"Quão completa é a trilha de auditoria sobre o acesso do usuário ao sistema e dados?"
"RISCO04"	"Seguranca"	"Confidencialidade"	"Quão controlável é o acesso ao sistema?"
"RISCO04"	"Seguranca"	"Integridade"	"Qual é a frequência de eventos de corrupção de dados?"
"RISCO04"	"Seguranca"	"Responsabilidade"	"Quão completa é a trilha de auditoria sobre o acesso do usuário ao sistema e dados?"

Fonte: Elaborada pela autora

A Figura 20 mostra a consulta que recupera as métricas, respondendo a questão “quais Metricas respondem as Questões que atendem cada Atributo?”. A consulta recupera indivíduos das classes Atributo, Subcaracteristica, Questao e Medida, onde, Subcaracteristica ‘compoe’ Atributo, Questao ‘atende’ Subcaracteristica e ‘eRespondidaPor’ Medida.

Figura 20 - Consulta que recupera as Métricas de cada 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 ?Atributo_nome ?Subcaracteristica_nome
       ?Questao_descricao ?Medida_nome
 WHERE { ?Subcaracteristica ont:compoe ?Atributo
         ?Questao ont:atende ?Subcaracteristica .
         ?Questao ont:eRespondidaPor ?Medida .
         ?Atributo ont:nome ?Atributo_nome .
         ?Subcaracteristica ont:nome ?Subcaracteristica_nome .
         ?Questao ont:descricao ?Questao_descricao .
         ?Medida ont:nome ?Medida_nome }
 ORDER BY ?Atributo_nome ?Subcaracteristica_nome
```

Atributo_nome	Subcaracteristica_nome	Questao_descricao	Medida_nome
"Confiabilidade"	"Disponibilidade"	"Quão disponível é o sistema para uso durante um período de tempo específico?"	"Tempo de reparo"
"Confiabilidade"	"Disponibilidade"	"Quão disponível é o sistema para uso durante um período de tempo específico?"	"Disponibilidade"
"Confiabilidade"	"Disponibilidade"	"Qual é o tempo médio em que o sistema fica indisponível quando uma falha ocorre, antes da inicialização?"	"Tempo médio indisponível"
"Confiabilidade"	"Maturidade"	"Qual a proporção de falhas detectadas que foram corrigidas?"	"Remocao de falhas"
"Confiabilidade"	"Recuperabilidade"	"Qual o tempo médio que o sistema leva para completar a recuperação desde o início?"	"Tempo médio de recuperação"
"Confiabilidade"	"Tolerancia_a_falhas"	"Quantas vezes o produto de software causa a queda de todo o ambiente de produção?"	"Prevenção de quedas"
"Confiabilidade"	"Tolerancia_a_falhas"	"Quantos padrões de faltas são mantidos sob controle para evitar falhas criticas e sérias?"	"Preveção de falhas"
"Eficiencia"	"Utilizacao_de_recursos"	"Qual é o limite absoluto de transmissões necessárias para cumprir uma função?"	"Utilização máxima de transmissão"
"Eficiencia"	"Utilizacao_de_recursos"	"O sistema é capaz de desempenhar tarefas dentro da capacidade de transmissão esperada?"	"Utilização da capacidade de transmissão"
"Seguranca"	"Confidencialidade"	"Quão controlável é o acesso ao sistema?"	"Controlabilidade de acesso"
"Seguranca"	"Integridade"	"Qual é a frequência de eventos de corrupção de dados?"	"Prevenção à corrupção de dados"
"Seguranca"	"Integridade"	"Qual é a frequência de eventos de corrupção de dados?"	"Prevenção à corrupção de dados"
"Seguranca"	"Responsabilidade"	"Quão completa é a trilha de auditoria sobre o acesso do usuário ao sistema e dados?"	"Auditabilidade do acesso"

Fonte: Elaborada pela autora

A Figura 21 exibe a consulta SPARQL que responde a questão “quais as Formas de Medição que medem cada Medida?”. Nesta consulta, foram recuperados os indivíduos das classes MedidaDerivada e MetodoDeMedicao onde FuncaoDeCalculo ‘calcula’ MedidaDerivada e ‘usa’ MedidaBase que ‘eColetadapor’ MetodoDeMedicao.

Figura 21 – Consulta que retorna as Técnicas de Testes para cada Métrica

```
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 DISTINCT ?Medida_nome ?Tecnica_nome
 WHERE { ?MedidaDerivada ont:nome ?Medida_nome .
         ?FuncaoDeCalculo ont:calcula ?MedidaDerivada .
         ?FuncaoDeCalculo ont:usa ?MedidaBase .
         ?MedidaBase ont:eColetadaPor ?MetodoDeMedicao .
         ?MetodoDeMedicao ont:nome ?Tecnica_nome }
 ORDER BY ?Medida_nome
```

Medida_nome	Tecnica_nome
"Auditabilidade do acesso"	"Teste_de_penetração"
"Auditabilidade do acesso"	"Teste_funcional"
"Controlabilidade de acesso"	"Teste_de_penetração"
"Controlabilidade de acesso"	"Teste_funcional"
"Disponibilidade"	"Teste_de_indução_de_falhas"
"Disponibilidade"	"Teste_de_longa_duração"
"Prevenção de quedas"	"Teste_de_indução_de_falhas"
"Prevenção de quedas"	"Teste_funcional"
"Prevenção de quedas"	"Teste_de_longa_duração"
"Prevenção à corrupção de dados"	"Teste_de_penetração"
"Preveção de falhas"	"Teste_de_indução_de_falhas"
"Preveção de falhas"	"Teste_de_longa_duração"
"Remocao de falhas"	"Teste_de_indução_de_falhas"
"Remocao de falhas"	"Teste_funcional"
"Tempo de reparo"	"Teste_de_indução_de_falhas"
"Tempo de reparo"	"Teste_de_longa_duração"
"Tempo médio de recuperação"	"Teste_de_indução_de_falhas"
"Tempo médio de recuperação"	"Teste_de_longa_duração"
"Tempo médio indisponível"	"Teste_de_indução_de_falhas"
"Tempo médio indisponível"	"Teste_de_longa_duração"
"Utilização da capacidade de transmissão"	"Teste_de_penetração"
"Utilização máxima de transmissão"	"Teste_de_penetração"

Fonte: Elaborada pela autora

A consulta que retorna o nível de qualidade (QL) do projeto que está sob avaliação para certificação e apresentada na Figura 22 e responde a questão “qual o Indicador QL do projeto?”. Esta consulta retorna o indivíduo da classe Indicador cujo nome seja ‘QLprojeto’. Retorna ainda a propriedade tipo dado ‘resultado’, que armazena o valor calculado do indicador.

Figura 22 - Consulta que retorna o QL

```
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 ?Indicador_nome ?AV
 WHERE { ?Indicador ont:eCalculadoCom ?ModeloDeAnalise .
         ?Indicador ont:nome ?Indicador_nome .
         ?Indicador ont:resultado ?AV .
 FILTER (?Indicador_nome = 'QLprojeto') }
```

Indicador_nome	AV
"QLprojeto"	"0.8240"

Fonte: Elaborada pela autora

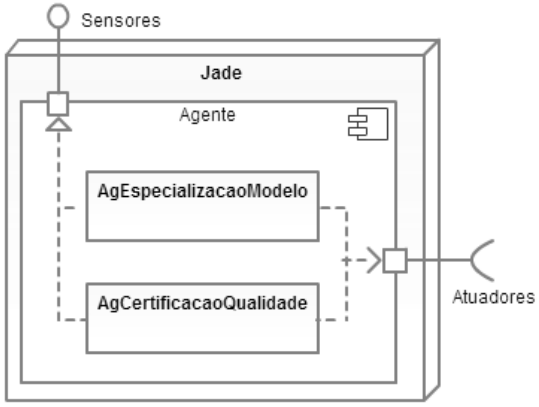
5.3. Agente de Certificação da Qualidade

Conforme descrito na Seção 5.1, no desenvolvimento deste trabalho utilizou-se de agentes para manipular os indivíduos na ontologia, inferir as métricas e realizar os cálculos no sistema de apoio à certificação de qualidade de produtos de software. Para tal foram definidos dois agentes reativos que interagem com o ambiente.

Vale destacar que não é responsabilidade dos agentes manipular a ontologia propriamente dita, mas sim, os indivíduos desta. Seus sensores percebem as ações externas, que são realizadas pelo usuário na interface gráfica da aplicação e, seus atuadores, definem consultas SPARQL, realizando cálculos, e manipulam indivíduos configurando propriedades tipo dado e propriedades tipo objeto, ou criando novos indivíduos.

A arquitetura e implementação dos agentes segue as bases da plataforma Jade, e a estrutura Agentes que encapsula os agentes de especialização do modelo e de certificação da qualidade é apresentado na Figura 23.

Figura 23 - Arquitetura do pacote Agente



Fonte: Elaborada pela autora

AgEspecializacaoModelo é o agente responsável pela especialização do modelo de qualidade. Seus principais comportamentos são DefinirConsulta, que consiste em definir as *query* SPARQL para que sejam realizadas as devidas consultas, e DefinirProriedadeObjeto, que consiste em configurar as propriedades tipo objeto em indivíduos da ontologia.

A funcionalidade de inferir métricas está embutida em DefinirConsultas, pois a mesma se dá por meio de consultas SPARQL nos indivíduos das classes Medida. A funcionalidade de apoio à associação de indivíduos, tais como Riscos e Requisitos, ou Riscos e Atributos, está

embutida em DefinirProriedadeObjeto, pois, tara tal, o agente deve configurar propriedades tipo objeto, que representam as associações.

O agente de certificação da qualidade é representado por AgCertificacaoQualidade. Seu comportamento consiste em CertificarProduto, pelo qual realiza os devidos cálculos de medidas derivadas e indicadores, e DefinirProriedadeDado, pelo qual configura as propriedades tipo dado dos indivíduos da ontologia, propriedades estas que descrevem os resultados dos cálculos.

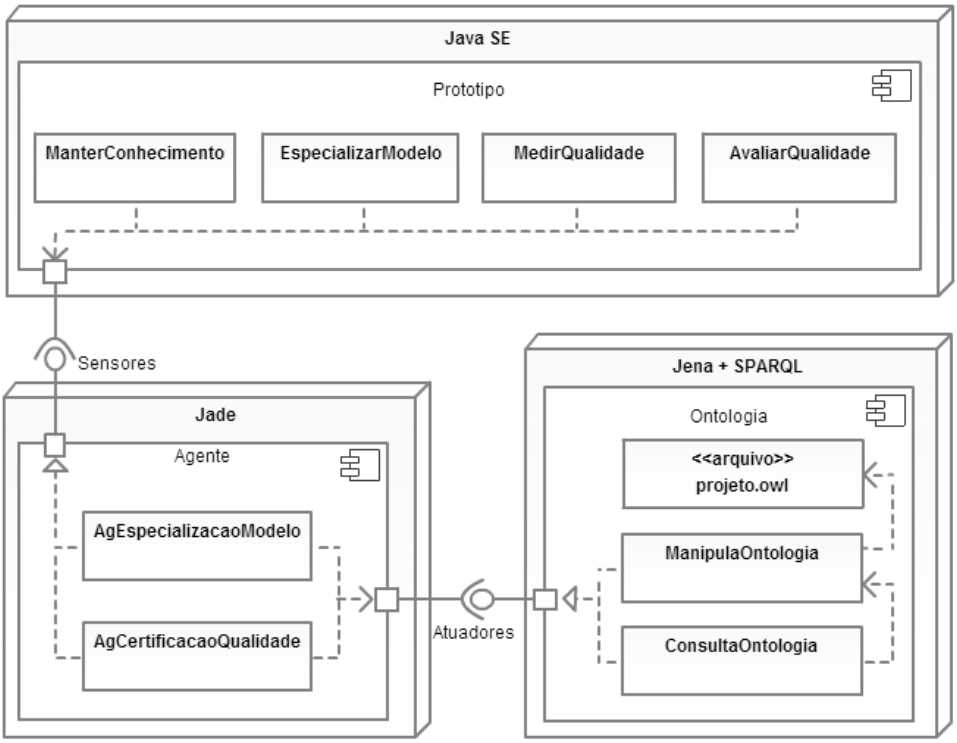
Os agentes atuam sobre o modelo da ontologia (*OntologyModel*) que é instanciado pelo Jena no componente Ontologia, descrito na seção anterior.

5.4. Protótipo de Interfaces

Para viabilizar a interação entre os agentes e a ontologia, foi desenvolvido um protótipo de interfaces, cujo objetivo é prover interface gráfica para as etapas de verificação e validação do sistema. Deste modo, este protótipo não implementa padrões elementares de engenharia de software, tais como, de usabilidade, segurança e validação de campos.

O protótipo foi desenvolvido com tecnologia Java SE (ORACLE, 2013) e roda em plataforma *desktop*. A Figura 24 apresenta a versão completa da arquitetura da aplicação, inclusive o componente Protótipo que encapsula as interfaces gráficas.

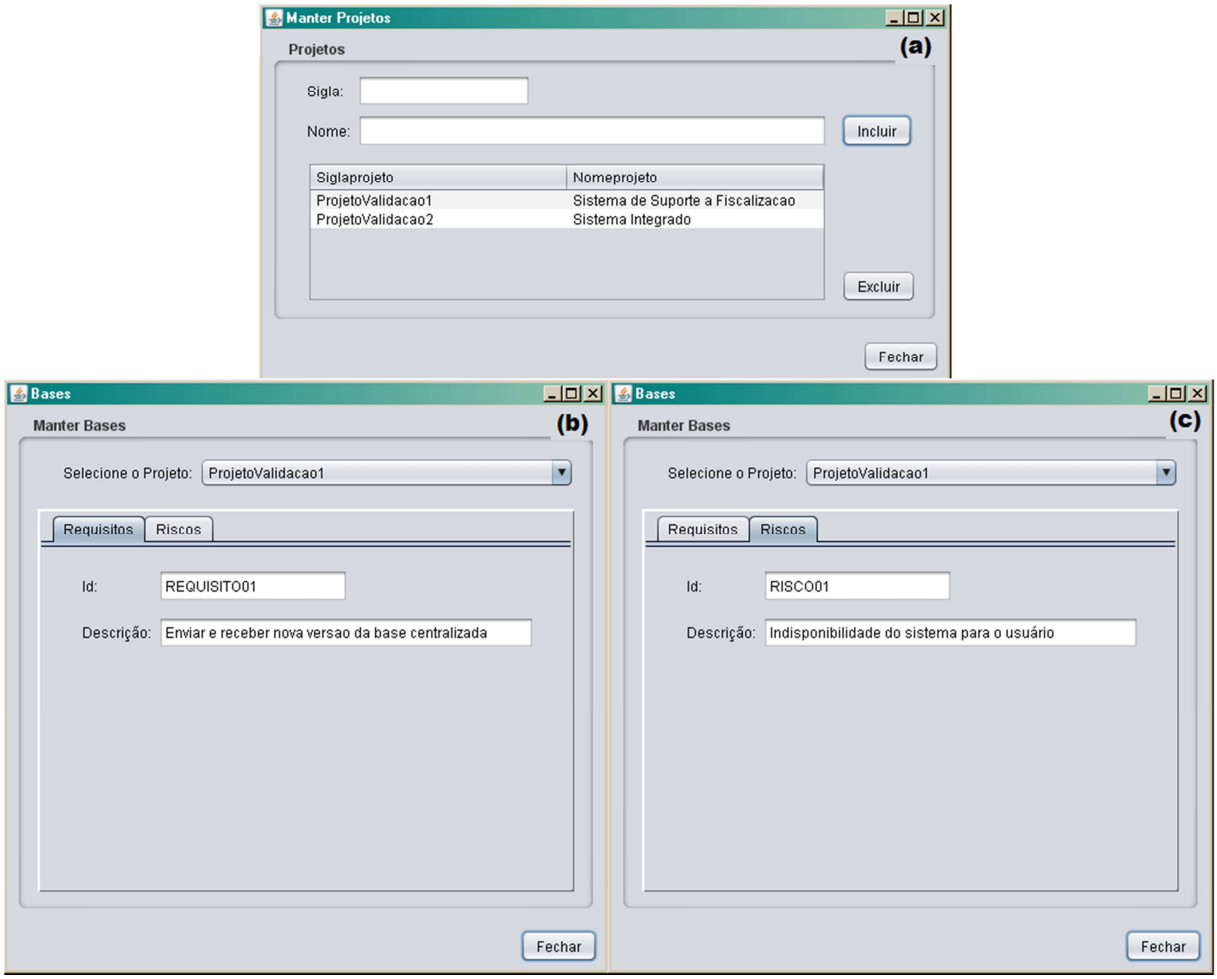
Figura 24 - Arquitetura completa do sistema



Fonte: Elaborado pela autora

Conforme pode ser observado, as interfaces gráficas foram organizadas conforme as funcionalidades do sistema. As Figuras 25 e 26 apresentam as telas da funcionalidade Manter Conhecimento, que visa a manutenção dos cadastros necessários.

Figura 25 - Telas de manutenção de projetos, riscos e requisitos



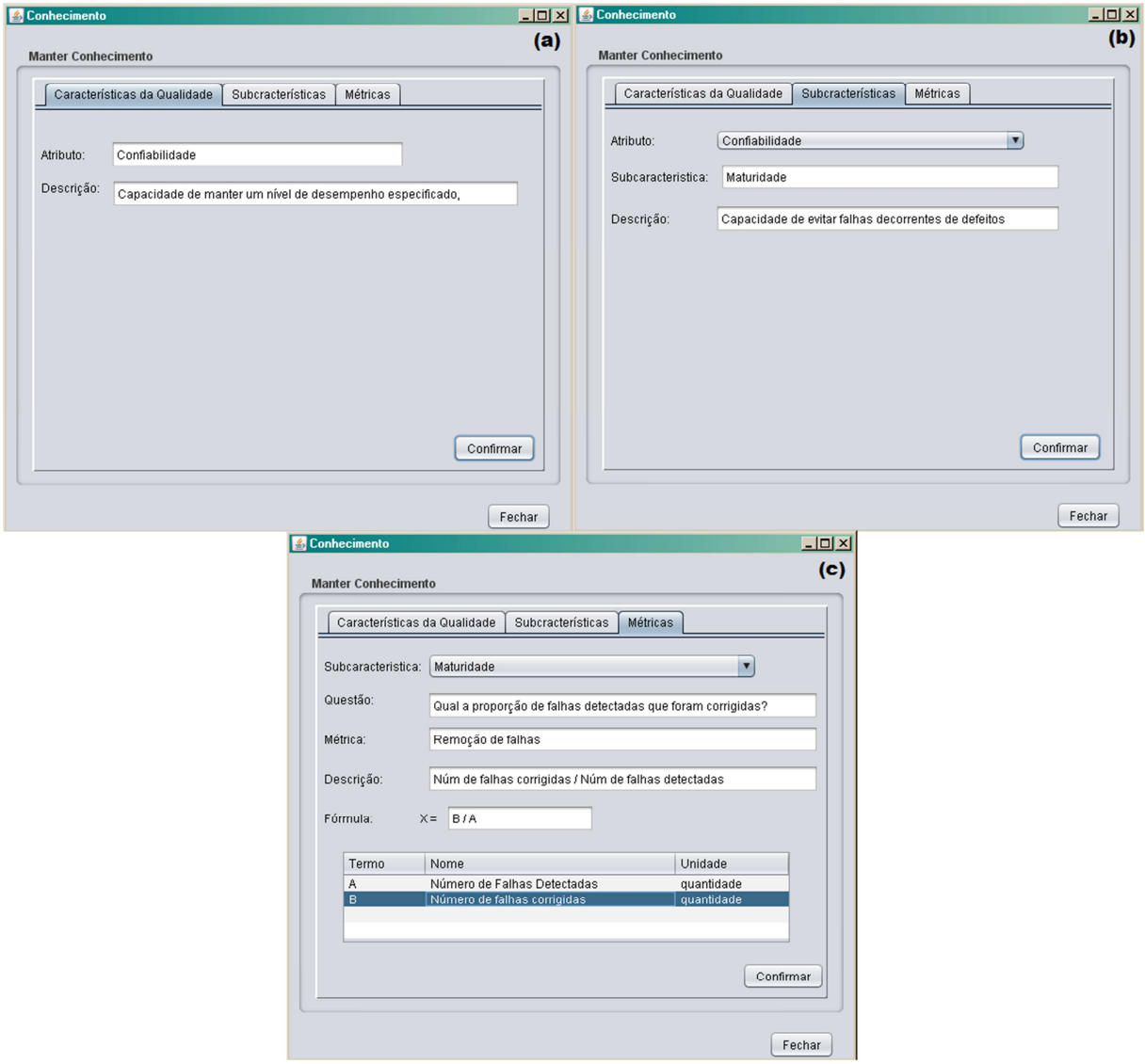
Fonte: Elaborado pela autora

A Figura 25(a) mostra tela pela qual o usuário pode cadastrar os projetos que serão submetidos à certificação. Já as Figuras 25(b) e 25(c), mostram as telas onde são realizados os cadastros dos requisitos e dos riscos do projeto, respectivamente.

Cabe reforçar que estas telas são apenas de cadastro e não de associação. As atividades de associação se darão nas interfaces que serão exibidas nas próximas figuras.

O cadastro de atributos da qualidade, suas subcaracterísticas e métricas correspondentes é realizado nas telas exibidas nas Figuras 26(a), 26(b) e 26(c), respectivamente.

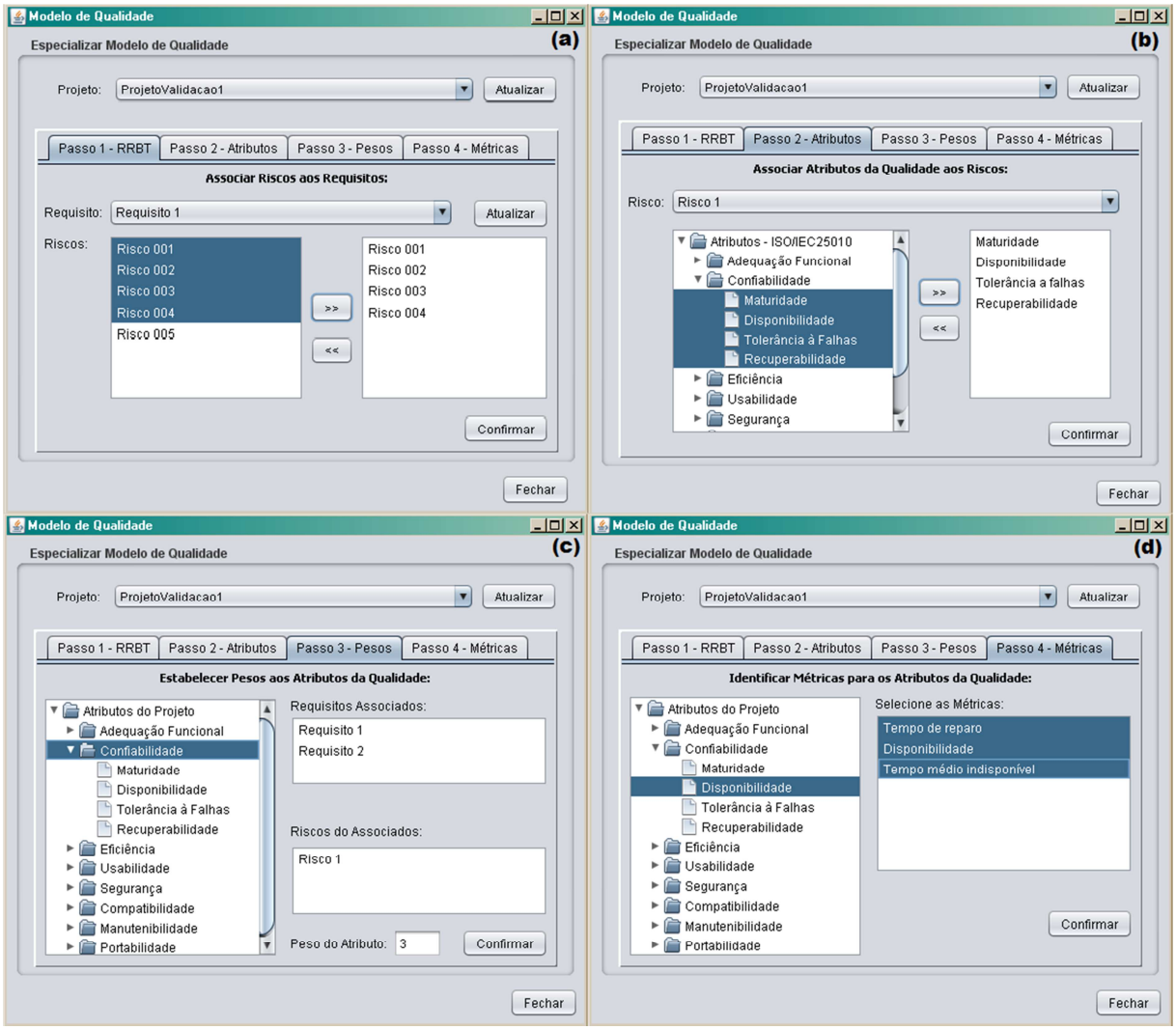
Figura 26 - Telas de manutenção de cadastros de atributos, subcaracterísticas e métricas



Fonte: Elaborado pela autora

A Figura 27 mostra as telas para realizar os passos de 1 a 4 do *framework*, que correspondem à funcionalidade de Especializar Modelo. O passo 1 que corresponde à associação de Riscos e Requisitos é realizado na tela exibida na Figura 27(a); em 27(b) são associados atributos da qualidade a cada risco, conforme passo 2; em 27(c) é atribuído o peso de cada atributo, conforme passo 3 sendo que, esta tela exibe, ainda, os requisitos e os riscos que já foram associados ao atributo, a fim de facilitar a ponderação; na tela exibida em 27(c) é realizado o passo 4 pelo qual são selecionadas as métricas para medição de cada atributo.

Figura 27 - Telas de especialização do modelo de qualidade

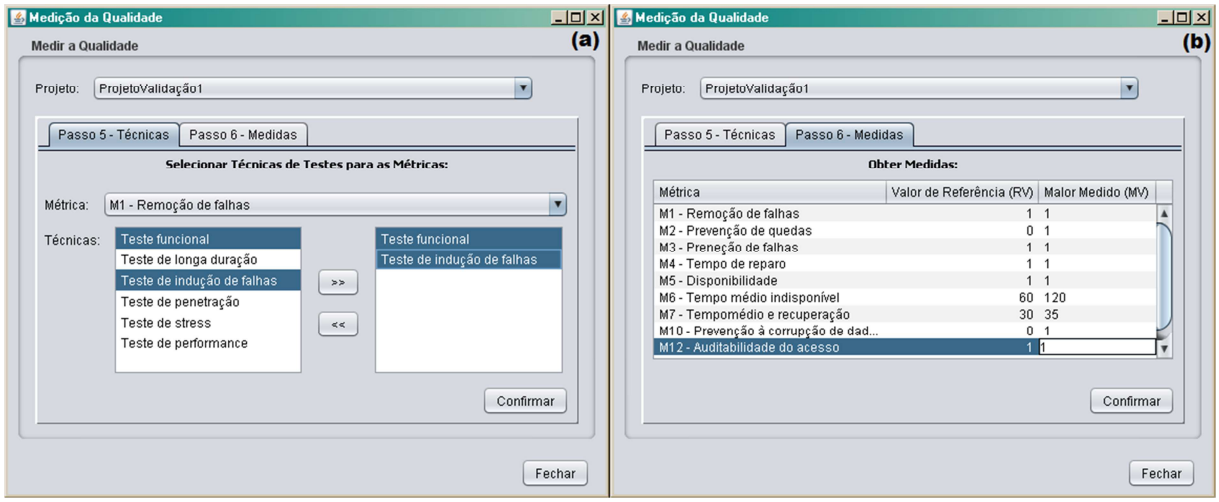


Fonte: Elaborado pela autora

A funcionalidade que se relaciona com o processo de Medir a Qualidade é realizada pelas telas que são exibidas na Figura 28. Em 28(a), antes da realização dos testes, o usuário seleciona as técnicas de testes, conforme preconiza passo 5 do *framework* de certificação de qualidade do produto de software. Depois de realizados os testes, em 28(b), o usuário registra os resultados obtidos em cada medida.

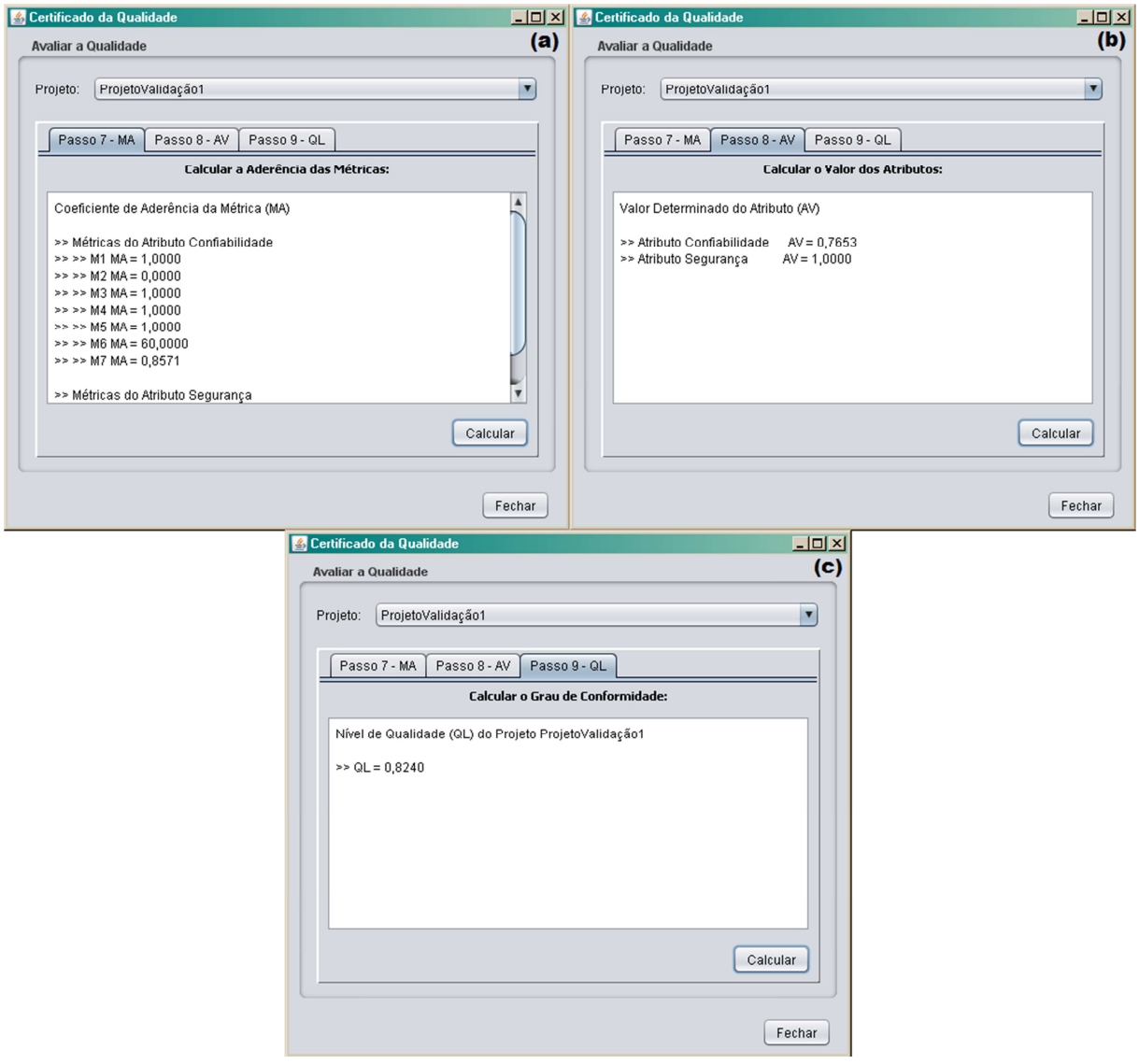
A Figura 29 traz as telas pelas quais se realiza a funcionalidade de Avaliar a Qualidade, correspondente aos passos 7 a 9 do *framework*. Estas telas basicamente exibem o resultado dos cálculos de aderência das métricas em 29(a), valor determinado dos atributos em 29(b) e o resultado final do nível de qualidade do projeto em 29(c).

Figura 28 - Telas de medição da qualidade



Fonte: Elaborado pela autora

Figura 29 - Telas de certificação da qualidade



Fonte: Elaborado pela autora

6. VERIFICAÇÃO E VALIDAÇÃO

No desenvolvimento de software, o processo de verificação garante que os produtos de trabalho selecionados estão de acordo com os requisitos especificados. Geralmente é um processo incremental, iniciando com a verificação de componentes do produto e terminando com a verificação do produto final (CMU/SEI, 2006). O processo de validação valia o produto contra as necessidades do usuário e deve ser realizado no ambiente operacional ou em um ambiente simulado (CMU/SEI, 2006).

Este Capítulo apresenta a estratégia de verificação e validação do sistema de apoio à certificação de qualidade de produtos de software. Na Seção 6.1 é descrita a etapa de verificação do sistema enquanto que a validação, tanto do *framework* quanto do sistema, é descrita na Seção 6.2. Uma síntese com avaliação dos resultados é apresentada na Seção 6.3.

6.1.Verificação

A verificação do sistema foi por meio de testes em nível unitário, de integração e de sistema, sendo todos os níveis realizados em ambiente de desenvolvimento. No nível unitário, foram verificadas as entradas e saídas de cada método, passando-se os devidos parâmetros e observando os resultados apresentados, na medida em que os mesmos foram desenvolvidos. Os testes de integração foram realizados para verificar a troca de mensagens entre os métodos. Os testes em nível de sistema foram realizados por meio da interface gráfica, pela qual foram realizados todos os cenários do caminho crítico da aplicação, basicamente os que permitem a realização dos passos do *framework* de certificação de qualidade de produtos de software.

Vale destacar que os testes foram realizados em laboratório, pois o sistema não foi implantando em ambiente real. Além disso, por tratar-se de um protótipo, não foram observados aspectos de usabilidade ou segurança.

6.2.Validação

Como este trabalho propõe um *framework* para certificação de qualidade e um sistema de apoio que visa automatizar o processo proposto, uma validação manual do *framework*, antes do desenvolvimento do protótipo do sistema, 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 do sistema, conforme descrito nas seções seguintes.

6.2.1. 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, e todos os passos dos subprocessos foram realizados manualmente, com apoio de planilhas eletrônicas.

O estudo de caso baseou-se em uma entrega parcial de um sistema de suporte à emissão de auto de infração, na fiscalização de tributos federais. O sistema é desenvolvido por uma empresa de TIC (Tecnologia da Informação e Comunicação) de grande porte e é a terceira versão de um sistema que, desde a primeira, encontra-se em produção há vinte anos. Entre suas principais características, destacam-se a capacidade de agregar toda a atividade de fiscalização em uma única aplicação, comunicando e recebendo dados de outros sistemas, e a capacidade de configuração de formas de cálculo e de padrões de relatórios de saída, o que fornece autonomia ao usuário para realizar determinadas alterações na aplicação, conforme as mudanças na legislação tributária.

O sistema em questão foi construído em Java para plataforma *desktop*, e tem uma abrangência de cerca de três mil usuários.

Para a validação do *framework*, a primeira etapa foi buscar a especificação de requisitos do sistema. Foram selecionados requisitos relevantes e identificados os riscos associados com apoio de informações obtidas com a equipe de desenvolvimento.

Os requisitos selecionados para este estudo de caso referem-se a funcionalidades de recepção e envio de dados a bases centralizadas. Os riscos foram obtidos de um documento de gestão de riscos que é gerado por um comitê composto pelas áreas de negócio, de infraestrutura, de suporte e de segurança da empresa, juntamente com a equipe de desenvolvimento, conforme preconiza o processo de desenvolvimento daquela empresa.

Deste modo, o passo 1 - Associar riscos de negócio aos requisitos da aplicação, foi realizado com apoio desta documentação e foram identificados quatro riscos para os dois requisitos selecionados.

A partir daí, a etapa seguinte foi a realização do passo 2 - Associar atributos e subcaracterísticas da qualidade aos riscos. Nesta etapa foram associados três atributos aos ricos, e selecionadas as subcaracterísticas mais relevantes.

Em seguida, foi considerada a ponderação dos riscos no documento de gestão de ricos já citado no passo 1, e realizado o passo 3 - Estabelecer pesos (grau de importância) para cada atributo da qualidade.

A Tabela 6 exibe o resultado da realização dos passos 1 a 3 do *framework*, para o estudo de caso.

Tabela 6 - Resultados dos passos 1 a 3 do *framework* para o 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, das informações que ficam inseridas no banco local instalado na estação de trabalho do usuário.		Responsabilidade	
			Confidencialidade	

Fonte: Elaborado pela autora

Por tratar-se de uma aplicação *desktop* com acessos pontuais às bases centralizadas ou de outros sistemas e, principalmente pelo tipo de informação que é trafegada na rede, os atributos Confiabilidade e Eficiência foram considerados mais importantes do que Segurança.

A etapa seguinte foi a realização do GQM para completar o passo 4 - Identificar métricas para cada atributo. As questões e as métricas, foram facilmente identificadas na norma *SQuaRe* ISO/IEC 25023, visto que a norma traz uma descrição para as métricas em forma de questão, para cada subcaracterística.

O passo 5 - Associar técnicas de testes a cada métrica identificada, foi realizado com apoio do que sugere a norma NBR ISO/IEC 9126-1 (2003) como método de aplicação para

verificação de cada métrica e, principalmente, com a opinião da equipe de desenvolvimento, especialmente o arquiteto e os analistas de requisitos.

A Tabela 7 traz os resultados da realização dos passos 4 e 5 para o risco 001 do estudo de caso.

Tabela 7 - Resultados dos passos 4 e 5 do *framework* para o Risco 001 do estudo de caso

***Passo 4*: GQM**		
R001 - Confiabilidade / Maturidade e Tolerância a Falhas		
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 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 / numero 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

Fonte: Elaborado pela autora

Para avaliação da Confiabilidade, faz-se necessário que o sistema seja submetido a um longo período de tempo em operação e que sejam simuladas situações que causem falhas e quedas no sistema.

Para o risco 002, que tem relação com a eficiência na utilização de recursos de modo que não resulte em indisponibilidade do sistema, os resultados dos passos 4 e 5 são apresentados na Tabela 8.

Tabela 8 - Resultados dos passos 4 e 5 do *framework* para o Risco 002 do estudo de caso

***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 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 stress Valores limites de usuários simultâneos Valores limites de dados trafegados (simular carga máxima)

Fonte: Elaborado pela autora

Como pode ser observado, para Eficiência aparecem as técnicas de teste de stress e carga, e as medidas são com base nas mensagens de erro apresentadas pela aplicação durante as situações de simulação de carga máxima e valores limites. Para este tipo de teste é fundamental a utilização de ferramentas automatizadas para as simulações necessárias. A capacidade de transmissão deve ser medida como uma relação da quantidade de bits por uma unidade de tempo ou por usuário, funcionalidade, etc.

Tanto para Confiabilidade como para Eficiência, é necessário conhecer os valores mínimos necessários, desejáveis e aceitáveis para os requisitos de modo que as medidas possam ser comparadas e a devida avaliação seja realizada. Para o sistema que é alvo deste estudo de caso, o modelo de requisitos especifica que são estimados 900 usuários simultâneos com carga aproximada de 40 MB trafegados por usuário, na funcionalidade de receber uma nova versão da aplicação da base centralizada. Para a funcionalidade de integração com outros sistemas, a estimativa é igualmente de 900 usuários simultâneos.

Os riscos 003 e 004 tratam de Segurança. A recategorização deste atributo é uma das principais diferenças na revisão das normas na série *SQuaRe*. Na versão anterior, NBR ISO/IEC 9126-1 (2003), Segurança era uma subcaracterística de Funcionalidade e, agora, na versão revisada ISO/IEC 25010 (2009), Segurança é apresentada como um atributo com suas devidas subcaraterísticas.

A Tabela 9 apresenta os resultados dos passos 4 e 5 para o risco 003, que tem relação com a integridade dos dados.

Tabela 9 - Resultados dos passos 4 e 5 do *framework* para o Risco 003 do estudo de caso

***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

Fonte: Elaborado pela autora

Para avaliações da integridade dos dados e segurança de acesso, são sugeridos testes de penetração. São testes que simulam ataques às vulnerabilidades do sistema. Estes tipos de testes são tipicamente executados por especialistas em segurança e baseados em padrões de ataques e de vulnerabilidades.

Para o risco 004 foi avaliado que as subcaraterísticas mais adequadas seriam Responsabilidade e Confidencialidade. Os resultados dos passos 4 e 5 para o risco 004 estão na Tabela 10.

Tabela 10 - Resultados dos passos 4 e 5 do *framework* para o risco 004 do estudo de caso

***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

Fonte: Elaborado pela autora

Para completar o subprocesso de medição da qualidade a próxima etapa é realizar o passo 6 - Executar testes para obter as medidas.

Cabe ressaltar, neste momento, que os testes na aplicação alvo não foram realizados especificamente para esta pesquisa, por questões contratuais estabelecidas pelo cliente. Os testes foram realizados em um centro de testes com o objetivo de identificar e remover defeitos e não conformidades, conforme preconiza o processo de desenvolvimento da empresa desenvolvedora do software. Após a conclusão dos testes, os resultados foram cedidos para esta pesquisa.

De posse dos resultados dos testes, foi possível a realização da etapa de avaliação da qualidade, que visa a certificação do produto, através da realização dos passos 7, 8 e 9 do *framework* de certificação de qualidade de produtos de software.

O passo 7 - Calcular a aderência das medidas obtidas aos valores de referência para todas as métricas foi realizado aplicando-se a fórmula do MA (*Metric Adherence Coefficient*) para as métricas cujas medidas foram coletadas.

Como os testes não foram realizados com o objetivo de certificar o produto, mas sim de encontrar e remover falhas, nem todas as métricas identificadas para o modelo de qualidade do sistema foram medidas. Deste modo, para a avaliação final do produto que visa validar o *framework* proposto nesta pesquisa foi retirado do escopo do modelo de qualidade o atributo Eficiência. O passo 8 - Calcular o valor do atributo conforme valor de aderência de suas métricas, foi calculado aplicando-se a fórmula do AV (*Attribute Determined Value*) para Confiabilidade e Segurança.

O nível de qualidade do produto foi calculado no passo 9 - Calcular o grau de conformidade do produto conforme os pesos dos atributos, aplicando-se a fórmula do QL (*Quality Level*) que consiste no somatório dos valores determinados dos atributos considerando seus pesos. A Tabela 11 traz a sínteses dos resultados dos passos 7, 8 e 9.

Tabela 11 - Resultados dos passos 7 a 9 do *framework* para o estudo de caso

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						

Fonte: Elaborado pela autora

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, foi constatado 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 a 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.

6.2.2. Validação do Sistema

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.

Como 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, os quais são apresentados nos APÊNDICES I e II.

Os resultados esperados com estes testes de validação do sistema eram:

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

Para o primeiro objetivo, foram cadastrados o projeto, os ricos e os requisitos pela interface do sistema. Em seguida, as associações foram plenamente realizadas na aplicação. O APÊNDICE I traz o resultado de uma consulta SPARQL realizada sobre a ontologia populada, mostrando as associações desde o requisito até a técnica de teste.

Os cálculos foram submetidos ao sistema, a fim de atender o segundo objetivo, e os resultados obtidos foram conforme os esperados. O APÊNDICE II traz o resultado de uma consulta SPARQL realizada sobre a ontologia, que mostra os valores calculados para as métricas, os atributos e o nível de qualidade do produto.

6.3.Avaliação dos resultados

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. Além disso, com base nos resultados obtidos para as medidas que foram realizadas, constatou-se que as fórmulas e processo propostos para a certificação do produto apresentaram resultados coerentes.

O resultado QL = 0,9311 mostra-se coerente com os resultados dos testes, visto que a maioria das métricas obteve aderência de 100%. Apenas as medidas coletadas para as

métricas M6 e M7, do atributo Confiabilidade, que tem relação com os tempos de indisponibilidade e de recuperação em caso de falhas, ficaram com aderência inferior. O sistema apresentou em média 120 segundos de indisponibilidade em relação ao número de quedas provocadas, enquanto o esperado eram 60 segundos, e 35 segundos de tempo médio para recuperação, enquanto o esperado eram 30 segundos.

Como o atributo Confiabilidade teve peso maior, do que Segurança, os resultados destas métricas tiveram maior impacto no resultado final, do que se fosse uma simples média aritmética. Porém, outro fator que impactou no nível de qualidade (QL) calculado foi o fato de a amostra ser pequena, com apenas dois atributos sendo considerados no estudo de caso, de modo que o peso de cada um deles acabou sendo muito significativo na nota global.

Quanto ao sistema, foi observada uma agilidade significativa na realização das atividades relacionadas a cadastros e associações. Além disso, a automação dos cálculos demonstrou facilidade e segurança ao processo.

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

- 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;
- 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, e nem 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 cerificado por outro processo para o estudo de caso.

7. CONCLUSÃO

Este trabalho apresentou um protótipo de sistema que visa apoiar a certificação de qualidade de produtos de software, baseado em um *framework* orientado a riscos. O sistema se apoia em tecnologias de ontologias e agentes de software para representação do conhecimento e manipulação dos indivíduos.

Fundamentada na análise dos resultados obtidos na validação, a conclusão que se chega é de que o sistema de apoio à certificação de qualidade de produtos de software apresentado neste trabalho mostrou- se plenamente aplicável ao que se propõe. Os resultados demonstraram que o *framework* viabiliza a especialização do modelo e certificação de qualidade do produto, da mesma forma que o sistema de apoio conseguiu prover automação, agilidade e segurança ao processo.

A ontologia do certificado de qualidade mostrou-se adequada a proposta do *framework* e para uso no sistema. Porém, no desenvolvimento do trabalho foi observada certa dificuldade na modelagem da ontologia, principalmente para identificar as propriedades e as restrições das classes e indivíduos. Para superar esta dificuldade foi fundamental a adoção de um processo organizado, iterativo e principalmente o reuso e adaptação de uma ontologia já existente sobre o mesmo domínio.

O conjunto de ferramentas composto por OWL, Jena e SPARQL mostrou-se adequado para atender aos objetivos propostos, porém, demandou certa curva de aprendizagem, além de conhecimento significativo em orientação a objetos e Java.

Quanto ao agente de certificação da qualidade, destaca-se o encapsulamento das funções manipulação da ontologia que fornece flexibilidade, na medida em que permite que a mesma seja usada por outras aplicações. Constatou-se que a tecnologia de agentes também permitiu que os objetivos fossem atendidos, embora não tenham sido esgotadas as possibilidades de uso desta tecnologia, visto que foi implementado um agente reativo bastante simples.

Jade, por sua vez, é uma tecnologia com boa documentação e exemplos disponíveis, mas também demandou certa curva de aprendizado.

Nas Subseções seguintes são descritas as principais contribuições do trabalho e algumas expectativas de trabalhos futuros.

7.1.Principais Contribuições

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 automação do processo de certificação do produto, 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.

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

Outro diferencial desta proposta, 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 o sistema de apoio à 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 sutilmente para a disseminação sobre uso de ontologias em aplicações orientadas a engenharia de software, evidenciando as potencialidades desta tecnologia.

7.2.Trabalhos Futuros

Ao final da pesquisa vislumbra-se algumas oportunidades de continuação e trabalhos futuros.

Como as principais limitações observadas no estudo de caso foram o fato dos testes do sistema alvo não terem sido realizados com objetivo de certificar o produto e, como consequência, o experimento deu-se sobre uma amostra pequena, uma oportunidade de

trabalho futuro é, justamente, validar em um sistema ou projeto de software para o qual os testes sejam planejados conforme o modelo de qualidade e executados visando coletar medidas para todas as métricas, de modo a gerar uma amostra maior para calcular o grau de conformidade (QL).

Ainda no contexto de aprofundar a validação do *framework* e do sistema, seria oportuno realizar testes com usuários (arquiteto de software, analista de requisitos, analista de testes e testador) e aplicar um *survey* visando identificar as dificuldades e oportunidades de melhoria na identificação das técnicas de testes, tarefa esta que também se mostrou pouco trivial no experimento realizado.

Uma oportunidade de evolução no *framework* seria a inclusão de uma etapa para computar critérios relacionados ao processo de desenvolvimento do produto de software, tais como nível de maturidade, qualificação da equipe, esforço em testes, etc..

Em um nível mais alto, para a certificação propriamente dita, uma oportunidade de pesquisa seria a criação de categorias de selos de qualidade, baseadas com conjuntos de atributos que sejam representativos conforme o contexto do domínio da aplicação de software, a exemplo do Programa Qualidade do Café (ABIC, 20013) da ABIC. Para tal, um produto a ser gerado poderia ser um *survey* para identificar a percepção dos usuários de sistemas por segmento.

Para o sistema também vislumbra-se oportunidades de evolução. A mais trivial seria incluir uma funcionalidade de importação dos requisitos de algum formato alternativo (texto ou banco de dados), visando agilizar o processo para sistemas maiores, considerando que o modelo e requisitos tradicionalmente é armazenado em sistemas dedicados.

O sistema poderia, em trabalho futuro, incluir uma funcionalidade para apoiar a análise riscos baseada na técnica *MoSCoW priority* (*Must test, Shoud test, Could test, Would test*) (PINKSTER *et al*., 2006), partindo daí para a ponderação dos atributos no passo 3.

Ainda sobre os pesos dos atributos, uma oportunidade de pesquisa é representar a importância relativa de cada um deles por meio de termos linguístico e números f*uzzy*. Acredita-se que representação do conhecimento por meio de números e conjuntos *fuzzy* seja adequada para expressar a importância dos atributos da qualidade na medida em que expressam a vagueza.

REFERÊNCIAS

ABIC. **PQC – Programa de Qualidade do Café**. 2013. Disponível em: <http://www.abic.com.br/publique/cgi/cgilua.exe/sys/start.htm?sid=15>. Acesso em 01/02/2013.

ABNT. **Conheça a ABNT**. Associação Brasileira de Normas Técnicas. 2003. Disponível em: <http://www.abnt.org.br/m3.asp?cod_pagina=929>. Acesso em Maio/2012.

ALEM FILHO, Pedro. **O impacto da certificação de software e serviços na exportação**. PAINEL SETORIAL PROGRAMA NACIONAL DE CERTIFICAÇÃO DE SOFTWARES E SERVIÇOS. INMETRO. Rio de Janeiro, 2007. Disponível em: <http://www.inmetro.gov.br/painelsetorial/software.asp>. Acesso em: 01/05/2012.

ASSESPRO. **Teste OK!**. Associação das Empresas Brasileiras de Tecnologia da Informação, 2009. Disponível em: <http://www.testeok.com.br/>. Acesso em: 13/09/2009.

BACH, James. **The Challenge of "Good Enough" Software**. 2003. Disponível em: <http:// www.satisfice.com/articles/gooden2.pdf>. Acesso em: 04/12/2009.

BALCI, O. **A methodology for certification of modeling and simulation applications. Transactions on Modeling and Computer Simulation (TOMACS)**, October 2001. Disponível em: <http://delivery.acm.org/10.1145/510000/508369/p352-balci.pdf?ip=200.188.160.135&id=508369&acc=ACTIVE%20SERVICE&key=C2716FEBFA981EF12CF6C8BC048E031E4B5D0D9418CA09DD&CFID=233186484&CFTOKEN=30850098&__acm__=1373768761_bf1bfa1aa658e048789677da7db4c8df>. Acesso em 01/05/2012.

BASILI, Victor R., CALDIERA, Gianluigi, e ROMBACH, H. Dieter. **The Goal Question Metric Approach**. 1994. Disponível em: <ftp://ftp.cs.umd.edu/pub/sel/papers/gqm.pdf>. Acesso em: 02/12/2009.

BELCHIOR, Arnaldo Dias. **Um modelo fuzzy para Avaliação da Qualidade de Software**. Tese (Doutorado em Computação). Departamento de Engenharia de Sistemas e Computação – COPPE. Universidade Federal do Rio de Janeiro, Rio de Janeiro, 1997.

CAPOVILLA, Izilda Gomes Garcez. **Elementos Intrínsecos do Software e sua Influência na Qualidade do Processo de Desenvolvimento**. 1999. Dissertação (Mestrado em Qualidade) - IMECC - Instituto de Matemática, Estatística e Computação Científica, UNICAMP - Universidade Estadual de Campinas, Campinas.

CMU/SEI-2006-TR-008. **CMMI for Development, Version 1.2. CMMI-DEV, V1.2. Improving process for better products**, CMU, 2006.

COLOMBO, Regina Maria Thiene e GUERRA, Ana Cervigni. **Qualidade de Produto de Software**. PBQP/MCT, 2008. Disponível em: <http://www.mct.gov.br/index.php/content/view/2867.html#lista>. Acesso em: 13/09/2009.

CROSBY, Philip. **Quality is Free**. New York: McGraw-Hill, 1979.

CHRISSIS, Mary Beth; KONRAD, Mike; SHRUM, Sandy. **CMMI for development – Guidelines for Process Integration and Product Improvement**. Boston: Addison-Wesley, 2004.

DIAS NETO, Arilo Cláudio. **Introdução a teste de software**. Engenharia de Software Magazine. Ano 1, edição 1. 2007. Disponível em <www.devmedia.com.br/esm >. Acesso em: 13/09/2009.

DUARTE, K. C., FALBO, R. A. **Uma Ontologia de Qualidade de Software**. In: VII WORKSHOP DE QUALIDADE DE SOFTWARE. XIV SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, Anais: pp. 275-285, João Pessoa, 2000.

FABRINI, Fabrizio, FUSANI, Mario, LAMI, Giuseppe. **Basic Concepts of Software Certification**. CERTSOFT06 - FIRST INTERNATIONAL WORKSHOP ON SOFTWARE CERTIFICATION. A satellite event of FM2006. McMaster University, 2006.

FERRAZ, Priscila e PROENÇA, Adriano. **Serviços e exportação e a certificação de software**. PAINEL SETORIAL PROGRAMA NACIONAL DE CERTIFICAÇÃO DE SOFTWARES E SERVIÇOS. INMETRO. Rio de Janeiro, 2007. Disponível em: <http://www.inmetro.gov.br/painelsetorial/software.asp>. Acesso em: 01/05/2012.

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 noviembre 2006. Universidad de Castilla-la Mancha.

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: IX Encontro Anual de Computação, 2011, Catalão. ENACOMP. Disponível em: <http://www.enacomp.com.br/2011/anais/trabalhos-aprovados/pdf/enacomp2011_submission_24.pdf>. Acesso em: 01/02/2013

GUSMÃO, C. M. G.; GUEDES, M. S.; MONTEIRO, M.; CAMPELLO, A.; AMORIM, L. **OntoPRIME: Ontologia de Riscos para Ambientes de Desenvolvimento de Software Multiprojetos**. 2004. Universidade Federal de Pernambuco, Recife, Brasil.

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

HYDER, Elaine B., HESTON, Keith M., HEFLEY, Bill, PAULK, Mark C. **eSourcing Capability Model for Service Providers (eSCM-SP)**. ITSqc. Van Haren Publishing. 2009.

HUMPHREY, Watts. **Bugs or Defects. Carnegie Mellon Univesity - Software Engineering Intitute**. News at SEI, Abril, 1999. Disponível em: <http://www.sei.cmu.edu/news-at-sei/columns/watts_new/1999/March/watts-mar99.htm>. Acesso em: 01/05/2012.

IEEE. **Standard Glossary of Software Engineering Terminology**. Institute of Electrical and Electronics, Inc. 1990.

INMETRO. **A atividade de avaliação da conformidade**. Disponível em <http://www.inmetro.gov.br/qualidade/>. Acesso em: 01/05/2012.

ISO. **About ISO**. International Organization for Standardization. 2003. Disponível em: <http://www.iso.org/iso/about.htm>. Acesso em: 13/09/2009.

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.

ISO/IEC TR 9126-2. **Software engineering - Product quality - Part 2: External metrics**. ISO/IEC, 2003.

JENA. **An Introduction to RDF and the Jena RDF API**. 2010. Disponível em: <http://jena.sourceforge.net/tutorial/RDF_API/index.html>, Acesso em: 26/06/2011.

LUCK, Michael , MCBURNEY, Peter, SHEHORY, Onn, WILLMOTT, Steven. **Agent Technology: Computing as Interaction - A Roadmap for Agent Based Computing**. 2005. Disponível em: <http://www.agentlink.org/roadmap/al3rm.pdf>. Acesso em: 08/04/2012.

MAITINGUER, Sônia Thereza. **Um Método de Avaliação Especialista para produtos de Software, desenvolvido a partir dos requisitos de um Edital.** Dissertação (Mestrado Profissional em Engenharia Mecânica). Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica, Campinas, 2004. Disponível em: <http://libdigi.unicamp.br/document/?code=vtls000332940>. Acesso em: 09/11/2009.

MCCALL, Jim A., RICHARDS, Paul, WALTERS, Gene F. **Factors in Software Quality**. Tree Volumes, Novembro, 1977. In Pressman.

MCGUINNESS, D. L. e 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>. Acessado em: 26/06/2011.

MYERS, Glenford. **The Art of Software Testing**. John Wiley and Sons, New York, 1979.

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

NBR ISO/IEC 14598-1. **Tecnologia da Informação – Avaliação de produto de Software. Parte 1: Visão geral**. Associação Brasileira de Normas Técnicas, 2001.

OLIVEIRA, Kelly Rejane. **AdeQuaS: Ferramenta fuzzy para avaliação da qualidade de software**. Dissertação (Mestrado em Engenharia de Software) - MIA/UNIFOR. Fortaleza, 2002.

ORACLE. **Java SE at a Glance**. 2013. Disponível em: <http://www.oracle.com/technetwork/java/javase/overview/index.html>. Acesso em: 02/02/2013.

OXFORD. Oxford Advanced Learner's Dictionary of Current English. Oxford University Press, sixth edition, Oxford, 2000.

PINKSTER, Iris, BURGT, Bob van de, JANSEN, Denis. e VEENENDAAL, Eric van,. **Successful Test Management: An Integral Approach**. Springer, Berlin, 2006.

PMI. **Um guia do conhecimento em gerenciamento de projetos (Guia PMBOK) - quarta edição**. PMI Inc, Pennsylvania, 2008.

PROTEGÉ. **Welcome to Protegé**. 2013. Disponível em: < http://protege.stanford.edu/>. Acesso em: 01/05/2012.

PRESSMANN, Roger S. **Engenharia de Software**. Makron Books, 1995.

REIS, Luís Paulo**. Coordenação em Sistemas Multi-Agente: Aplicações na Gestão Universitária e Futebol Robótico**. PhD Thesis (Tese de Doutorado). Faculdade de Engenharia da Universidade do Porto. 2003. Disponível em: <http://paginas.fe.up.pt/~lpreis/Tese/>. Acesso em: 01/05/2012.

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: 13/09/2009.

SANTOS, Adriana dos. **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: 23/08/2009.

SANTOS, Lizandra Bays e PRETZ, Eduardo. ***Framework* para especialização de modelos de qualidade de produtos de software**. In: SBQS 2010 - IX SIMPÓSIO BRASILEIRO E QUALIDADE DE SOFTWARE. SBC – Sociedade Brasileira de Computação. Belém do Pará, Brasil, 2010. Anais, artigos técnicos, páginas de 57 a 71.

SIBISI, Mbusi e WAVEREN, Cornelis Cristo van. **A process framework for customising software quality models**. Proc. AFRICON 2007, páginas 1-8.

SIDDIQUI, F. e ALAM, M. A. **Web Ontology Language Design and Related Tools: A Survey**. Hamdard University, India. 2010. Disponível em <http://www.ojs.academypublisher.com/index.php/jetwi/article/view/03014759>. Acessado em: 15/05/2011.

SILVA, João Pablo S. **Um sistema multiagente baseado em ontologias para apoio às inspeções de garantia da qualidade de software**. 2010. Dissertação (Mestrado em

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

SOFTWARE CERTIFICATIONS. **Guide to the CSTE Common Body of Knowledge**. Quality Assurance Institute, 2006, version 6.2.

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 18/12/2009.

SPIDER, Projeto. **Sobre o SPIDER**. 2009. Disponível em: <http://www.spider.ufpa.br/ibndex.php?id=sobre>. Acesso em 02/02/2013.

TOTUM. **Selos e Programas de Auto-Regulamentação**. 2013. Disponível em: <http://www.institutototum.com.br/selos_parceiros.php>. Acesso em 02/02/2013.

W3C. **OWL Web Ontology Language, Overview**. 2004. Disponível em: <http://www.w3.org/TR/owl-features/>. Acesso em: 26/06/2011.

W3C. **SPARQL Query Language for RDF**. 2008. Disponível em: <http://www.w3.org/TR/rdf-sparql-query/>. Acesso em: 26/06/2011.

WOOLDRIDGE, Michael. **An Introduction to MultiAgent Systems**. John Wiley & Sons Ltd, United Kingdom, 2009.

APÊNDICE I - RESULTADO DE CONSULTA SPARQL QUE MOSTRA TODAS AS ASSOCIAÇÕES

Requisito_nome	Risco_nome	Atributo_nome	Subcaracteristica_nome	Questao_descricao	Medida_nome	Tecnica_nome
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Disponibilidade"	"Qual é o tempo médio em que o sistema fica indisponível quando uma falha ocorre, antes da inicialização?"	"Tempo médio indisponível"	"Teste_de_indução_de_falhas"
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Disponibilidade"	"Qual é o tempo médio em que o sistema fica indisponível quando uma falha ocorre, antes da inicialização?"	"Tempo médio indisponível"	"Teste_de_longa_duração"
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Disponibilidade"	"Quão disponível é o sistema para uso durante um período de tempo específico?"	"Tempo de reparo"	"Teste_de_indução_de_falhas"
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Disponibilidade"	"Quão disponível é o sistema para uso durante um período de tempo específico?"	"Tempo de reparo"	"Teste_de_longa_duração"
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Disponibilidade"	"Quão disponível é o sistema para uso durante um período de tempo específico?"	"Disponibilidade"	"Teste_de_indução_de_falhas"
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Disponibilidade"	"Quão disponível é o sistema para uso durante um período de tempo específico?"	"Disponibilidade"	"Teste_de_longa_duração"
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Maturidade"	"Qual a proporção de falhas detectadas que foram corrigidas?"	"Remocao de falhas"	"Teste_de_indução_de_falhas"
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Maturidade"	"Qual a proporção de falhas detectadas que foram corrigidas?"	"Remocao de falhas"	"Teste_funcional"
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Recuperabilidade"	"Qual o tempo médio que o sistema leva para completar a recuperação desde o início?"	"Tempo médio de recuperação"	"Teste_de_indução_de_falhas"
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Recuperabilidade"	"Qual o tempo médio que o sistema leva para completar a recuperação desde o início?"	"Tempo médio de recuperação"	"Teste_de_longa_duração"
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Tolerancia_a_falhas"	"Quantas vezes o produto de software causa a queda de todo o ambiente de produção?"	"Prevenção de quedas"	"Teste_de_indução_de_falhas"
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Tolerancia_a_falhas"	"Quantas vezes o produto de software causa a queda de todo o ambiente de produção?"	"Prevenção de quedas"	"Teste_funcional"
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Tolerancia_a_falhas"	"Quantas vezes o produto de software causa a queda de todo o ambiente de produção?"	"Prevenção de quedas"	"Teste_de_longa_duração"
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Tolerancia_a_falhas"	"Quantos padrões de faltas são mantidos sob controle para evitar falhas criticas/sérias?"	"Preveção de falhas"	"Teste_de_indução_de_falhas"
"REQUISITO01"	"RISCO01"	"Confiabilidade"	"Tolerancia_a_falhas"	"Quantos padrões de faltas são mantidos sob controle para evitar falhas criticas/sérias?"	"Preveção de falhas"	"Teste_de_longa_duração"
"REQUISITO01"	"RISCO02"	"Eficiencia"	"Utilizacao_de_recursos"	"O sistema é capaz de desempenhar tarefas dentro da capacidade de transmissão esperada?"	"Utilização da capacidade de transmissão"	"Teste_de_penetração"
"REQUISITO01"	"RISCO02"	"Eficiencia"	"Utilizacao_de_recursos"	"Qual é o limite absoluto de transmissões necessárias para cumprir uma função?"	"Utilização máxima de transmissão"	"Teste_de_penetração"
"REQUISITO01"	"RISCO03"	"Seguranca"	"Confidencialidade"	"Quão controlável é o acesso ao sistema?"	"Controlabilidade de acesso"	"Teste_de_penetração"
"REQUISITO01"	"RISCO03"	"Seguranca"	"Confidencialidade"	"Quão controlável é o acesso ao sistema?"	"Controlabilidade de acesso"	"Teste_funcional"
"REQUISITO01"	"RISCO03"	"Seguranca"	"Integridade"	"Qual é a frequência de eventos de corrupção de dados?"	"Prevenção à corrupção de dados"	"Teste_de_penetração"
"REQUISITO01"	"RISCO03"	"Seguranca"	"Responsabilidade"	"Quão completa é a trilha de auditoria sobre o acesso do usuário ao sistema e dados?"	"Auditabilidade do acesso"	"Teste_de_penetração"
"REQUISITO01"	"RISCO03"	"Seguranca"	"Responsabilidade"	"Quão completa é a trilha de auditoria sobre o acesso do usuário ao sistema e dados?"	"Auditabilidade do acesso"	"Teste_funcional"
"REQUISITO01"	"RISCO04"	"Seguranca"	"Confidencialidade"	"Quão controlável é o acesso ao sistema?"	"Controlabilidade de acesso"	"Teste_de_penetração"
"REQUISITO01"	"RISCO04"	"Seguranca"	"Confidencialidade"	"Quão controlável é o acesso ao sistema?"	"Controlabilidade de acesso"	"Teste_funcional"
"REQUISITO01"	"RISCO04"	"Seguranca"	"Integridade"	"Qual é a frequência de eventos de corrupção de dados?"	"Prevenção à corrupção de dados"	"Teste_de_penetração"
"REQUISITO01"	"RISCO04"	"Seguranca"	"Responsabilidade"	"Quão completa é a trilha de auditoria sobre o acesso do usuário ao sistema e dados?"	"Auditabilidade do acesso"	"Teste_de_penetração"
"REQUISITO01"	"RISCO04"	"Seguranca"	"Responsabilidade"	"Quão completa é a trilha de auditoria sobre o acesso do usuário ao sistema e dados?"	"Auditabilidade do acesso"	"Teste_funcional"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Disponibilidade"	"Qual é o tempo médio em que o sistema fica indisponível quando uma falha ocorre, antes da inicialização?"	"Tempo médio indisponível"	"Teste_de_indução_de_falhas"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Disponibilidade"	"Qual é o tempo médio em que o sistema fica		

				indisponível quando uma falha ocorre, antes da inicialização?"	"Tempo médio indisponível"	"Teste_de_longa_duração"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Disponibilidade"	"Quão disponível é o sistema para uso durante um período de tempo específico?"	"Tempo de reparo"	"Teste_de_indução_de_falhas"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Disponibilidade"	"Quão disponível é o sistema para uso durante um período de tempo específico?"	"Tempo de reparo"	"Teste_de_longa_duração"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Disponibilidade"	"Quão disponível é o sistema para uso durante um período de tempo específico?"	"Disponibilidade"	"Teste_de_indução_de_falhas"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Disponibilidade"	"Quão disponível é o sistema para uso durante um período de tempo específico?"	"Disponibilidade"	"Teste_de_longa_duração"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Maturidade"	"Qual a proporção de falhas detectadas que foram corrigidas?"	"Remocao de falhas"	"Teste_de_indução_de_falhas"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Maturidade"	"Qual a proporção de falhas detectadas que foram corrigidas?"	"Remocao de falhas"	"Teste_funcional"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Recuperabilidade"	"Qual o tempo médio que o sistema leva para completar a recuperação desde o início?"	"Tempo médio de recuperação"	"Teste_de_indução_de_falhas"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Recuperabilidade"	"Qual o tempo médio que o sistema leva para completar a recuperação desde o início?"	"Tempo médio de recuperação"	"Teste_de_longa_duração"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Tolerancia_a_falhas"	"Quantas vezes o produto de software causa a queda de todo o ambiente de produção?"	"Prevenção de quedas"	"Teste_de_indução_de_falhas"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Tolerancia_a_falhas"	"Quantas vezes o produto de software causa a queda de todo o ambiente de produção?"	"Prevenção de quedas"	"Teste_funcional"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Tolerancia_a_falhas"	"Quantas vezes o produto de software causa a queda de todo o ambiente de produção?"	"Prevenção de quedas"	"Teste_de_longa_duração"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Tolerancia_a_falhas"	"Quantos padrões de faltas são mantidos sob controle para evitar falhas criticas/sérias?"	"Preveção de falhas"	"Teste_de_indução_de_falhas"
"REQUISITO02"	"RISCO01"	"Confiabilidade"	"Tolerancia_a_falhas"	"Quantos padrões de faltas são mantidos sob controle para evitar falhas criticas/sérias?"	"Preveção de falhas"	"Teste_de_longa_duração"
"REQUISITO02"	"RISCO02"	"Eficiencia"	"Utilizacao_de_recursos"	"O sistema é capaz de desempenhar tarefas dentro da capacidade de transmissão esperada?"	"Utilização da capacidade de transmissão"	"Teste_de_penetração"
"REQUISITO02"	"RISCO02"	"Eficiencia"	"Utilizacao_de_recursos"	"Qual é o limite absoluto de transmissões necessárias para cumprir uma função?"	"Utilização máxima de transmissão"	"Teste_de_penetração"
"REQUISITO02"	"RISCO03"	"Seguranca"	"Confidencialidade"	"Quão controlável é o acesso ao sistema?"	"Controlabilidade de acesso"	"Teste_de_penetração"
"REQUISITO02"	"RISCO03"	"Seguranca"	"Confidencialidade"	"Quão controlável é o acesso ao sistema?"	"Controlabilidade de acesso"	"Teste_funcional"
"REQUISITO02"	"RISCO03"	"Seguranca"	"Integridade"	"Qual é a frequência de eventos de corrupção de dados?"	"Prevenção à corrupção de dados"	"Teste_de_penetração"
"REQUISITO02"	"RISCO03"	"Seguranca"	"Responsabilidade"	"Quão completa é a trilha de auditoria sobre o acesso do usuário ao sistema e dados?"	"Auditabilidade do acesso"	"Teste_de_penetração"
"REQUISITO02"	"RISCO03"	"Seguranca"	"Responsabilidade"	"Quão completa é a trilha de auditoria sobre o acesso do usuário ao sistema e dados?"	"Auditabilidade do acesso"	"Teste_funcional"
"REQUISITO02"	"RISCO04"	"Seguranca"	"Confidencialidade"	"Quão controlável é o acesso ao sistema?"	"Controlabilidade de acesso"	"Teste_de_penetração"
"REQUISITO02"	"RISCO04"	"Seguranca"	"Confidencialidade"	"Quão controlável é o acesso ao sistema?"	"Controlabilidade de acesso"	"Teste_funcional"
"REQUISITO02"	"RISCO04"	"Seguranca"	"Integridade"	"Qual é a frequência de eventos de corrupção de dados?"	"Prevenção à corrupção de dados"	"Teste_de_penetração"
"REQUISITO02"	"RISCO04"	"Seguranca"	"Responsabilidade"	"Quão completa é a trilha de auditoria sobre o acesso do usuário ao sistema e dados?"	"Auditabilidade do acesso"	"Teste_de_penetração"
"REQUISITO02"	"RISCO04"	"Seguranca"	"Responsabilidade"	"Quão completa é a trilha de auditoria sobre o acesso do usuário ao sistema e dados?"	"Auditabilidade do acesso"	"Teste_funcional"

APÊNDICE II - RESULTADO DE CONSULTA SPARQL QUE MOSTRA OS CÁLCULOS

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