UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

UMA ABORDAGEM PARA AVALIAÇÃO DA QUALIDADE DE ARTEFATOS DE SOFTWARE

DISSERTAÇÃO DE MESTRADO

Gelson Bertuol

Santa Maria, RS, Brasil

2014

UMA ABORDAGEM PARA AVALIAÇÃO DA QUALIDADE DE ARTEFATOS DE SOFTWARE

Gelson Bertuol

Dissertação apresentada ao Curso de Mestrado do Programa de
Pós-Graduação em Informática (PPGI), Área de Concentração em Computação
Aplicada, da Universidade Federal de Santa Maria (UFSM, RS),
como requisito parcial para obtenção de grau de
Mestre em Ciência da Computação

Orientadora: Profª. Drª. Lisandra Manzoni Fontoura

Santa Maria, RS, Brasil

2014

Bertuol, Gelson
Uma Abordagem para Avaliação da Qualidade de Artefatos de Software / Gelson Bertuol – 2014.
106 p.; 30cm

Orientadora: Lisandra Manzoni Fontoura
Dissertação (mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Programa de Pós-Graduação em Informática, RS, 2014

1. Engenharia de Software 2. Qualidade de Software 3. Modelos de Qualidade 4. Metamodelos de Qualidade 5. Avaliação de Software I. Fontoura, Lisandra Manzoni II. Título.

Endereço: Universidade Federal de Santa Maria, Cidade Universitária, “Prof. Mariano da Rocha Filho”, Av. Roraima, 1000, LaCA, Sala 388 Anexo B - CT, Bairro Camobi, CEP: 97105-900, End. Eletrônico: gelson.bertuol@gmail.com

UFSM – UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

UMA ABORDAGEM PARA AVALIAÇÃO DA QUALIDADE DE ARTEFATOS DE SOFTWARE

elaborada por

Gelson Bertuol

como requisito parcial para obtenção do grau de
Mestre em Ciência da Computação

COMISSÃO EXAMINADORA:

Lisandra Manzoni Fontoura, Dra.
(Presidente/Orientadora)

Eduardo Kessler Piveta, Dr.
(Universidade Federal de Santa Maria – UFSM)

Marcelo Soares Pimenta, Dr.
(Universidade Federal do Rio Grande do Sul – UFRGS)

Santa Maria, 27 de Agosto de 2014

AGRADECIMENTOS

Aos meus pais, Anselmo e Sirlei, e à minha irmã Patrícia – pelo carinho e apoio incondicional em todas as etapas de minha vida, especialmente nesta última.

Agradeço à professora Lisandra Fontoura pela oportunidade de ingressar no mestrado; pela paciência, apoio e estímulo no decorrer deste. Sem sua excelente orientação e confiança, este trabalho jamais seria concluído.

Agradeço à Maria e à Jade pelo companheirismo, apoio e incentivo durante todo o tempo que durou o desenvolvimento desta dissertação.

Aos meus colegas de mestrado, Camila, Darciele, Ruan, Wagner, Miguel, Maicon, Aline, Denise, Fabrício, Nielsen, Catherine, Daniel Biasoli (quem mais me incentivou a ingressar no curso) e a todos que, de alguma forma, me ajudaram e torceram pelo sucesso deste trabalho.

Ao Programa de Pós-Graduação em Informática da UFSM, por proporcionar a estrutura necessária para esta pesquisa.

A Deus.

“O mais importante na vida não é a situação em que nos encontramos,
mas a direção para a qual nos movemos.”

Oliver Wendell Holmes

RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria

UMA ABORDAGEM PARA AVALIAÇÃO DA QUALIDADE DE ARTEFATOS DE SOFTWARE

AUTOR: GELSON BERTUOL
ORIENTADORA: DRA. LISANDRA MANZONI FONTOURA
Local e Data da Defesa: Santa Maria, 27 de agosto de 2014.

Ao mesmo tempo em que as aplicações e os sistemas de software vêm evoluindo e tornando-se mais complexos, devido, principalmente, à crescente exigência dos clientes e usuários, as organizações que os produzem ou os adquirem têm buscado alternativas para reduzir custos e prazos de entrega sem que a qualidade do produto final seja afetada. Contudo, para que a avaliação desses produtos seja mais eficaz, é importante utilizar um modelo de qualidade que permita estruturá-la de forma que satisfaça, entre outros requisitos, as expectativas heterogêneas dos interessados. Paralelamente, recomenda-se iniciar essa avaliação o mais cedo possível, já nos primeiros estágios de um processo de desenvolvimento com o objetivo de detectar e corrigir os problemas encontrados antes que se propaguem. Neste sentido, este trabalho apresenta um estudo sobre modelos de qualidade empregados na avaliação de produtos de software ao mesmo tempo em que propõe a avaliação dos artefatos, gerados e/ou transformados pelas atividades, ao longo do ciclo de vida de um processo de desenvolvimento. A proposta é baseada em um *framework* de qualidade, estruturado a partir de um metamodelo, que relaciona o processo de avaliação às diversas características que envolvem os artefatos, tais como seus propósitos, interessados, métodos e métricas correspondentes. O trabalho é composto, ainda, por uma ferramenta de apoio cujo objetivo é guiar os avaliadores na definição de um plano de avaliação da qualidade de tais artefatos. Por fim, a proposta foi avaliada e validada por meio de um estudo de caso envolvendo estudantes de pós-graduação em informática na avaliação de três aplicações reais desenvolvidas por acadêmicos de graduação da Universidade Federal de Santa Maria.

Palavras-chave: Qualidade de Software, Modelos de Qualidade, Artefatos de Software, Metamodelo de Qualidade, Metas de Qualidade.

ABSTRACT

Master's Dissertation
Graduation Program in Computer Science
Federal University of Santa Maria

AN APPROACH FOR ASSESSING THE QUALITY OF SOFTWARE ARTIFACTS

AUTHOR: GELSON BERTUOL
ADVISOR: PROF. DRA. LISANDRA MANZONI FONTOURA
Place and Date: Santa Maria, August 27th, 2014.

While applications and software systems have evolved and becoming more complex, mainly due to the increasing demands of customers and users, organizations that produce or acquire have sought alternatives to reduce costs and deliveries without affect the final product quality. However, in order to make the evaluation of these products more effective, it is important to use a quality model that allows structure it in a way that satisfies, among other requirements, the heterogeneous expectations of stakeholders. At same time, it is recommended starting this evaluation as soon as possible since the early stages of a development process in order to detect and fix any problems before they propagate. In this sense, this work presents a study on quality models used in the evaluation of software products, proposing at the same time the assessment of software artifacts, generated and/or transformed by activities throughout the lifecycle of a software process. The proposal is based on a quality framework, structured from a metamodel, which relates the process of evaluating the several characteristics that involve the artifacts, such as their purposes, stakeholders, methods and corresponding metrics. The work is also composed by a supporting tool which purpose is to guide evaluators in defining a plan for assessing the quality of those artifacts. Finally, the proposal was submitted to validation through a case study involving graduate students of Federal University of Santa Maria.

Keywords: Software Quality, Quality Models, Software Artifacts, Quality Metamodel, Quality Metrics.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ATL	*Atlas Transformation Language*
BPMN	*Business Process Modeling Notation*
ES	Engenharia de Software
GQM	*Goal Question Metric*
HCI	*Human-Computer Interaction*
IEC	*International Electrotechnical Commission*
ISO	*International Organization for Standardization*
MDE	*Model-Driven Engineering*
MDWE	*Model-Driven Web Engineering*
OCL	*Object Constraint Language*
OOP	*Object-Oriented Programming*
PMBOK	*Project Management Body of Knowledge*
PMI	*Project Management Institute*
QUIM	*Quality in Use Integrated Measurement*
RUP	*Rational Unified Process*
SEQUAL	*SEmiotic QUALity*
SOA	*Service Oriented Architecture*
SPL	*Software Product Line*
SQA	*Software Quality Assurance*
SQuaRE	*Software product QUAlity Requirements and Evaluation*
TAM	*Technology Acceptance Model*
UML	*Unified Modeling Language*
XP	*eXtreme Programming*
xUML	*Executable Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

A Garantia da Qualidade de Software (*Software Quality Assurance* – SQA), um dos pilares da disciplina de Engenharia de Software (ES), visa avaliar objetivamente processos executados e produtos de trabalho em relação às suas descrições, especificações e requisitos, sendo fundamental para o sucesso de um projeto de software. Segundo Pressman (2010), para que a SQA alcance seus objetivos, é preciso definir explicitamente o significado de qualidade. Contudo, o conceito sobre "o que é qualidade" tem evoluído ao longo do tempo com o objetivo de incorporar visões e propósitos diferentes que, combinados, possam abranger o máximo de características possíveis dentro de um mesmo produto.

Inicialmente, a qualidade de um produto de software estava relacionada apenas à ausência de defeitos (medida pela quantidade de erros por linhas de código) ou à sua confiabilidade (horas de operação sem falhas) (ELBERZHAGER; MÜNCH, 2011; KAN, 2002). Posteriormente, outras definições mais abrangentes foram propostas, incluindo, por exemplo, a conformidade e aderência a padrões ou especificações (SINGH; KANNOJIA, 2013); a completude no atendimento aos requisitos especificados no projeto (GUERRA; COLOMBO, 2009; IEEE, 2009); a otimização na utilização de recursos (PRESSMAN, 2010); e, por fim, a satisfação das expectativas e necessidades dos clientes e usuários (IEEE, 2009; SURYN; ABRAN, 2003). Porém, apesar das inúmeras definições descritas na literatura, o termo *qualidade* é naturalmente vago e ambíguo, visto que é composto por elementos cuja avaliação depende de uma série de fatores subjetivos, entre eles, a interpretação humana (KAN, 2002; ROSENBERG; SHEPPARD, 1994; WAGNER; DEISSENBOECK, 2007).

Contudo, mesmo possuindo aspectos que são inerentemente subjetivos e complexos, a avaliação da qualidade de softwares deve ser formalizada por meio de um processo que empregue métodos sistemáticos e cujo objetivo seja atender ao máximo as expectativas heterogêneas que envolvem todos os *stakeholders*[1], além de fornecer entendimento para cada

[1] *Stakeholder* refere-se a todos os envolvidos no projeto, desde os membros da equipe de desenvolvimento até clientes e usuários. A tradução do termo para *interessados* também será usado ao longo do texto quando não houver perda de sentido.

um e comunicabilidade entre eles (TRENDOWICZ; PUNTER, 2003). Além disso, esse processo deve, preferencialmente, ser definido com base em um metamodelo, para que os elementos mais comuns possam ser elencados e relacionados com o objetivo de orientar os engenheiros de qualidade no cumprimento das suas tarefas (MOHAGHEGHI; DEHLEN; NEPLE, 2008).

Uma forma de contornar esse problema é – ao invés de se tentar medir a qualidade do software como um todo – buscar-se a avaliação de atributos que compõem o produto e que, quando combinados, possam fornecer uma noção geral sobre a sua qualidade (AL-KILIDAR; COX; KITCHENHAM, 2005). Neste sentido, os *artefatos de software* assumem um papel importante na Garantia da Qualidade de Software. Isso se deve, primeiramente, porque tais artefatos são gerados – ou transformados – constantemente pelas atividades que compõem as iterações de um processo de software. Além disso, a avaliação pode ser decomposta considerando cada fase do ciclo de vida e, assim, ajudar na predição da qualidade do produto final já nas primeiras fases do projeto, reduzindo custos e mitigando riscos.

Outro ponto importante é que durante a criação das várias versões de um produto, e à medida que o ciclo de vida do processo avança, é possível comparar o estado corrente dos artefatos gerados em relação à qualidade pretendida para o produto finalizado. Dessa forma, esses artefatos podem ser usados no controle de qualidade, também. Cada artefato tem certos atributos que afetam a qualidade do produto intermediário da próxima etapa e, consequentemente, afetam a qualidade do produto final.

Paralelamente, entre as diversas abordagens descritas na literatura, as que propõem modelos de qualidade (*quality models*) têm sido bem aceitas para descrever e definir a qualidade em produtos de software, uma vez que buscam estrutura-la em fatores pontuais e fáceis de serem avaliados, ao mesmo tempo em que fornecem uma boa caracterização desses fatores (DEISSENBOECK et al., 2009).

Uma das vantagens dos modelos de qualidade está na sua estrutura relativamente simples se comparada com outras abordagens matemáticas como: *multiple regression models* (BRIAND; WÜST, 2002); *logistic regression models* (KHOSHGOFTAAR; ALLEN, 1999) ou que envolvam inteligência artificial tais como: *neural networks* (AGGARWAL et al., 2008); *data mining* (KHOSHGOFTAAR et al., 1999); *multilayer fuzzy* (LIU; PANG, 2010) ou *fuzzy expert systems* (LIU; PANG, 2010). Ao mesmo tempo, tem-se que os Modelos de Qualidade são capazes de avaliar quantitativamente, e de forma objetiva, os diversos atributos de qualidade relacionados a fatores qualitativos. Outro ponto importante a favor dessa abordagem está em

sua maturidade. As primeiras propostas datam de meados dos anos 70 e, desde então, vêm evoluindo ao longo do tempo. Um exemplo disso é a ISO/IEC 9126 (ISO/IEC, 2001) que, depois de várias revisões, tornou-se um consenso e é uma das principais referências para definição de características de qualidade para produtos de software (AL-KILIDAR; COX; KITCHENHAM, 2005).

1.1 Definição do Problema

Contudo, embora *qualidade* seja um assunto recorrente na disciplina de Engenharia de Software, as empresas – especialmente aquelas de pequeno e médio porte – carecem de especialistas capazes de definir as características de qualidade para os seus produtos (GUERRA; COLOMBO, 2009). Mais do que isso, quando essa definição ocorre, a avaliação geralmente acontece de forma *ad-hoc* ou somente sobre o código-fonte, por meio de testes e muitas vezes, somente no final do processo de desenvolvimento, sem o uso de padrões ou modelos adequados capazes de nortear os avaliadores na definição dos propósitos de qualidade pretendidos (ARRUDA, 2014; DOMINGUEZ-MAYO et al., 2010; ELBERZHAGER; MÜNCH, 2011).

Por outro lado, Trendowicz e Punter (2003) argumentam que é irreal a ideia de se definir uma visão prescritiva das características de qualidade necessárias e suficientes que abranja uma descrição de requisitos de qualidade adequada que envolva todas as organizações, projetos e *stakeholders*. Provavelmente algumas dessas características, e suas relações, possam até ser verdadeiras para a maioria dos casos, porém, a regra é que existem diferenças entre as organizações e mesmo entre os projetos. Consequentemente, ao contrário dos modelos de melhoria de processos de software cujo gerenciamento e a execução são normalmente independentes dos produtos gerados, as medidas e técnicas usadas na avaliação da qualidade dos produtos e artefatos precisam ser adaptadas e/ou ajustadas para cada contexto e propósitos específicos.

Contudo, uma deficiência recorrente na maioria dos modelos de qualidade existentes é a sua falta de conformidade com um metamodelo explícito (DEISSENBOECK et al., 2009). Nesses casos, a semântica dos elementos pode não ser definida de forma clara e precisa, deixando a interpretação do modelo a cargo do leitor e, assim, dificultando esses ajustes. Mais

do que isso, os modelos de qualidade tradicionais não integram as diversas tarefas realizadas durante um processo de software. Por exemplo, a especificação da qualidade dos requisitos e a avaliação da qualidade do código-fonte não são, normalmente, baseadas em um mesmo modelo (WAGNER, 2007).

Em relação aos artefatos de software, a definição e a classificação das características de qualidade, por si só, não são suficientes sem uma discussão de quais os meios necessários para alcançá-las e quais são os papéis interessados (MOHAGHEGHI; DEHLEN, 2008a). Assim, a avaliação, nesses casos, depende muito de *o que* é medido; do *porque* um artefato de software deve ser avaliado; e *quem* são os interessados em tal avaliação. Modelos UML e código-fonte, por exemplo, diferem em vários fatores, entre eles, no nível de abstração, precisão, completude, consistência e correspondência com o produto final (LANGE; CHAUDRON, 2005). Portanto, métricas projetadas para medir um tipo de artefato, podem não servir para medir outro. Ao mesmo tempo, tem-se que o propósito de cada artefato pode diferir dependendo da abordagem de desenvolvimento usada, tornando-se mais ou menos importante. Ao contrário de um processo planejado como o *Rational Unified Process* (RUP), os modelos UML podem assumir pouca importância em métodos ágeis como *SCRUM* ou *eXtreme Programming* (XP).

Finalmente, tem-se que o conhecimento adquirido na avaliação da qualidade não deve ser perdido. Assim, a definição de uma base de conhecimento capaz de armazenar as experiências passadas pode ser útil em projetos futuros, além de reter esse aprendizado dentro da organização.

Sendo assim, o problema definido para esta dissertação é *como avaliar os diversos artefatos gerados pelas atividades de um processo de software, considerando, sobretudo, suas características, propósitos e interessados; isso tudo ajustado a uma abordagem específica de desenvolvimento e com vistas à melhoria da qualidade do produto final?*

1.2 Escopo e Contribuições da Pesquisa

Neste trabalho, defende-se a ideia de que a avaliação da qualidade de um produto de software pode ser iniciada já nos primeiros estágios de um processo de software, independentemente do modelo de processo adotado. Além disso, essa avaliação deve ser

baseada em três requisitos básicos: flexibilidade, reusabilidade e transparência. A escolha e o propósito destes requisitos é descrita e justificada no Capítulo 3.

Para isso, definiu-se um *framework* de qualidade cujo objetivo é estruturar a avaliação de artefatos de software gerados durante esse processo. A proposta visa abranger tanto artefatos já finalizados quanto aqueles ainda em desenvolvimento, ditos *artefatos intermediários*, de forma que a maturidade de cada artefato é considerada, porém não restritiva à avaliação. O *framework*, por sua vez, é composto por um metamodelo e a sua aplicação por meio de uma sequência de passos (um processo de avaliação), além de uma ferramenta de apoio, elaborados com vistas a facilitar a definição de metas de qualidade relacionadas à abordagem de desenvolvimento usada, aos propósitos de qualidade dos artefatos e à seleção das principais métricas e métodos de avaliação descritos na literatura.

Assim, a principal contribuição do trabalho está na proposta do mecanismo de avaliação, capaz de apoiar equipes de desenvolvimento de software – especialmente as menos experientes – na definição dos propósitos de qualidade almejados para o produto final por meio da avaliação dos artefatos de software. Outras contribuições incluem:

- Desenvolvimento de uma abordagem teórica para avaliação da qualidade dos artefatos gerados durante um processo de software;
- Proposta de um metamodelo de qualidade com o objetivo de estruturar a avaliação dos artefatos baseado nas metas de qualidade pretendidas, relacionando-as com os seus possíveis interessados, propósitos de qualidade, métodos e métricas de avaliação compatíveis;
- Definição de uma sequência lógica de ações (processo de avaliação) com o objetivo de guiar os avaliadores, e demais interessados, na estruturação de um plano de qualidade que identifique quais métodos e métricas são mais apropriados para avaliar a qualidade dos artefatos gerados;
- Desenvolvimento de um protótipo de ferramenta cujo objetivo é auxiliar os avaliadores na definição de um plano de avaliação da qualidade dos artefatos de software;
- Criar uma base de conhecimento inicial, como sugestão, que possa ser reusada pelas equipes de desenvolvimento de software na busca por formas de avaliação e controle de qualidade.

A validação da proposta foi realizada por meio de um estudo de caso com base em três projetos desenvolvidos por estudantes de graduação do curso de Sistemas de Informação da Universidade Federal de Santa Maria (UFSM) e avaliada por estudantes de pós-graduação em Informática da mesma universidade.

Por fim, o escopo da pesquisa limita-se à definição da estrutura proposta para a avaliação, bem como a sua adequação aos processos de avaliação de produtos de software já conhecidos e descritos na literatura. A aplicação das métricas e/ou práticas de avaliação para cada artefato está fora do escopo deste trabalho e fica a cargo de pesquisas futuras ou de possíveis usuários da abordagem. Contudo, referências bibliográficas de como essas métricas e práticas podem ser usadas são apresentadas ao longo do texto.

1.3 Estrutura da dissertação

Este trabalho está organizado da seguinte forma: no Capítulo 2 são apresentados e comparados alguns dos modelos de qualidade e processos de avaliação de produtos de software mais referenciados na literatura e que, de alguma forma, influenciaram e/ou serviram de base para a construção desta dissertação. No Capítulo 3, descreve-se o metamodelo de qualidade proposto para estruturar a avaliação dos artefatos de software. No Capítulo 4 é especificado o processo de avaliação construído para avaliar os artefatos de software, este, referenciado na ISO/IEC 14598 e estruturado de forma compatível com o metamodelo apresentado no Capítulo 3. Ainda no Capítulo 4, descreve-se o protótipo da ferramenta de avaliação implementada. No Capítulo 5 apresenta-se um estudo de caso realizado como forma de avaliação e validação da proposta dissertada. Por fim, no Capítulo 6 são relacionadas as considerações finais, resultados e trabalhos futuros desta dissertação.

2 MODELOS DE QUALIDADE

A qualidade de um produto de software, segundo a literatura de Engenharia de Software (ES), está relacionada a conjuntos de características capazes de refletir aspectos de qualidade desejados. Vários modelos tentam descrever, classificar e organizar essas características para que possam ser sistematicamente medidas e avaliadas. Assim, neste capítulo é apresentada, primeiramente, uma contextualização do termo *qualidade*. Em seguida, são descritos alguns dos modelos de qualidade mais referenciados na literatura e que serviram de base para a construção deste trabalho, iniciando pelos Modelos de Estrutura Hierárquica, Modelos Conceituais e Modelos de Contextos Específicos. Ainda, alguns processos de avaliação da qualidade de produtos de software são abordados e relacionados ao contexto deste trabalho. Ao final do capítulo é feita uma síntese das principais contribuições de cada abordagem.

2.1 Qualidade

Conforme mencionado no capítulo anterior, não há uma definição absoluta para o conceito de qualidade em ES. Isso se deve, principalmente, pelos fatores subjetivos que compõem a qualidade de um produto de software. Kan (KAN, 2002) atribui três razões principais para essa subjetividade. Primeiramente, qualidade não é uma ideia simples, mas sim, um conceito multidimensional que envolve pessoas com expectativas e pontos de vista diferentes. Em segundo lugar, para qualquer conceito há vários níveis de abstração. E, quando pessoas pensam na qualidade de algo, podem estar se referindo tanto a um sentido mais amplo quanto a um mais específico. Por fim, o termo qualidade faz parte da linguagem cotidiana e perspectivas populares e técnicas podem divergir significativamente.

Dessa forma, algumas taxonomias têm tentado segmentar a visão de qualidade com o objetivo de avaliá-la mais precisamente. Garvin (GARVIN, 1984), por exemplo, aborda cinco perspectivas diferentes que relacionam os conceitos de qualidade aos produtos de software. As abordagens propostas são: i) *visão transcendental* – visão filosófica que sustenta a ideia de que qualidade é algo que pode ser reconhecida, mas não definida explicitamente; ii) *visão do usuário* – visão pela qual a qualidade é vista em termos das metas específicas de um usuário

final, ou seja, se o produto atende as necessidades estabelecidas por quem vai usá-lo; iii) *visão do fabricante* – define a qualidade em termos da especificação original do produto; iv) *visão do produto* – a qualidade é tida como um conjunto de características, funções e recursos apresentados pelo produto; e finalmente, v) *visão baseada em valores* – mede a qualidade tomando como base o quanto um cliente estaria disposto a pagar pelo produto.

Essas definições, apesar de bastante abrangentes, ajudam a explicar as diferentes visões de qualidade atribuídas para cada um dos envolvidos no desenvolvimento de software, seus propósitos e preferências em relação aos fatores que julgam essenciais ou importantes, e que devem ser atendidos na implantação do produto. Em outras palavras, a visão dos envolvidos em um projeto de software deve ser considerada, bem como seus anseios e pretensões relacionados à qualidade. Outras contextualizações sobre qualidade de software, incluindo as de Crosby, Deming, Feigenbaum, Ishikawa, Juran e Shewhart, foram compiladas e descritas por Berander *et al.* (2005).

2.2 Modelos de Qualidade Hierárquicos

Dentre as primeiras propostas descritas para conceituar características de qualidade[2], estão os Modelos de Qualidade Hierárquicos, também conhecidos como Modelos Fixos (*Fixed-models* (FENTON; PFLEEGER, 1996)). Esses modelos normalmente descrevem a relação entre um conjunto fixo de qualidade de alto nível (fatores de qualidade), atributos do produto e métricas apropriadas para alcançar esses fatores (TRENDOWICZ; PUNTER, 2003). A estrutura hierárquica é o ponto chave desses modelos, pois permite organizar a qualidade em vários pilares que são, então, decompostos e refinados até atributos de qualidade específicos, capazes de serem avaliados quantitativamente por meio de métricas apropriadas (SINGH; KANNOJIA, 2013; TRENDOWICZ; PUNTER, 2003; VANDEROSE, 2012). Contudo, o conjunto de fatores de qualidade abordado e a forma como é decomposto depende de cada proposta. Assim, esta seção aborda alguns dos modelos hierárquicos mais relevantes a esta

[2] Alguns modelos usam o termo *fator de qualidade* como sinônimo de *característica de qualidade* para se referirem às características que influenciam na qualidade do produto de software. A fim de preservar as definições originais de cada modelo, ambos os termos são usados com o mesmo propósito nesta dissertação.

pesquisa e que, ao mesmo tempo, serviram como base teórica para o desenvolvimento de outros mais complexos e elaborados, além de normas e padrões internacionais de qualidade.

2.2.1 Modelo de Qualidade de McCall

Um dos primeiros modelos hierárquicos foi proposto por McCall *et al.* (MCCALL; RICHARDS; WALTERS, 1977). Também conhecido como o modelo da *General Eletric*, teve sua origem nos ambientes militares americanos no final da década de 70 (AL-QUTAISH, 2010; GUERRA; COLOMBO, 2009; SAMADHIYA; WANG; CHEN, 2010). A principal proposta desse modelo foca na tentativa de preencher a lacuna existente entre as expectativas dos usuários, descrita como *visão externa* (fatores de qualidade desejados e percebidos por quem usa o software) e as prioridades dos desenvolvedores, chamada de *visão interna* (critérios de qualidade que melhoram a construção e/ou manutenção do software). O modelo introduz onze fatores de qualidade, cuja avaliação está relacionada a três diferentes aspectos de qualidade de software:

- *Revisão do Produto*: habilidade pela qual um produto submete-se às mudanças, incluindo correções de erros e adaptações. Fazem parte desta perspectiva os fatores de qualidade: *manutenibilidade, flexibilidade* e *testabilidade*;
- *Transição do Produto*: relacionada à adaptabilidade do sistema a novos ambientes operacionais ou de *hardware*. É composta pelos fatores: *portabilidade, reusabilidade* e *interoperabilidade*;
- *Operação do Produto*: relacionada às características básicas de operação do produto, como a habilidade de ser facilmente entendido, operado e capaz de fornecer os resultados esperados pelos usuários. Os fatores de qualidade são: *correção, confiabilidade, eficiência, integridade e usabilidade*.

Cada um dos onze fatores de qualidade é estruturado para que possam ser hierarquicamente refinados em 23 critérios de qualidade internos ao produto e, então, associados a métricas de avaliação. Na Figura 2-1 é ilustrada a estrutura proposta por esse modelo de qualidade.

A ideia do modelo proposto por McCall é que os fatores de qualidade sintetizados forneçam um retrato completo da qualidade do software. Dessa forma, o processo de avaliação consiste em o avaliador responder “sim” ou “não” a uma série de questões relacionadas aos critérios de qualidade com o objetivo de medir até que ponto esses critérios podem ser alcançados (MOHAGHEGHI; DEHLEN, 2008a; SINGH; KANNOJIA, 2013).

Segundo Vanderose (2012), uma das maiores contribuição dessa proposta é a relação criada entre os fatores de qualidade e algumas métricas, embora existam críticas afirmando que nem todas as métricas são objetivas e que o modelo não aborda diretamente alguns fatores importantes como *funcionalidade* (AL-KILIDAR; COX; KITCHENHAM, 2005; AL-QUTAISH, 2010; SAMADHIYA; WANG; CHEN, 2010). Além disso, o processo de avaliação desse modelo é fortemente dependente da experiência do avaliador que deve identificar quais dos fatores de qualidade são ou não são atendidos.

Figura 2-1 – Modelo de qualidade proposto por McCall *et al.*, adaptado de (MCCALL; RICHARDS; WALTERS, 1977).

2.2.2 Modelo de Qualidade de Boehm

Seguindo uma estrutura similar à de McCall, o modelo proposto por Boehm *et al.* (BOEHM; BROWN; LIPOW, 1976; BOEHM et al., 1978) tenta definir qualitativamente um artefato de software segundo um conjunto pré-definido de atributos e métricas. A diferença é que este, além de abranger um conjunto maior de características, as separa em três níveis:

- *Características de alto nível*: representam requisitos básicos de alto nível no uso real de software. Em outras palavras, representam as expectativas dos usuários em relação à avaliação da qualidade de software. Essas características são classificadas

em *utilidades gerais* que incluem *portabilidade* e *manutenibilidade*, e características que os autores consideraram essenciais, porém não suficientes – chamadas de *as-is utilities*;

- *Características intermediárias*: representam sete fatores de qualidade, decompostos das características de alto nível, que juntos descrevem as qualidades esperadas para um sistema de software. As *as-is utilities* são decompostas em *confiabilidade*, *eficiência* e *engenharia humana* (característica que descreve a influência das condições de trabalho na produtividade). Outros fatores de qualidade deste nível são *portabilidade*, *testabilidade*, *compreensibilidade* e *flexibilidade* (ou *modificabilidade*) – estas três últimas decompostas de *manutenibilidade*;
- *Características primitivas*: fornecem a fundamentação para definir as métricas de qualidade relacionadas a um ou mais dos fatores descritos no nível anterior.

Outra diferença entre os dois modelos é que, enquanto McCall se concentra principalmente na medição das características de alto nível, o modelo de Boehm é baseado em uma ampla gama de características, além da ênfase na *manutenibilidade* dos produtos de software e, portanto, a inclusão de atributos específicos para essa característica. Em contrapartida, uma das críticas em relação a essa proposta, é que ela apresenta somente a hierarquia dos fatores de qualidade sem descrever nenhum processo de avaliação (SINGH; CHAWLA, 2012; VANDEROSE, 2012). Na Figura 2-2 é descrita a estrutura em árvore proposta pelo modelo de Boehm *et al.*

Figura 2-2 – Modelo de qualidade proposto por Boehm *et al.* Adaptado de (BOEHM et al., 1978).

2.2.3 ISO/IEC 9126

Embora os modelos de McCall e de Boehm possam parecer muito similares, existem diferenças com relação à hierarquia das características. Entretanto, pode-se afirmar que esses dois modelos inspiraram e serviram de base para uma série de outros modelos hierárquicos, incluindo a norma ISO/IEC 9126 (ISO/IEC, 2001). Proposta em 1991, ela foi posteriormente revisada e estabelecida como padrão internacional para avaliação da qualidade de produtos de software em 2001 (AL-KILIDAR; COX; KITCHENHAM, 2005; VANDEROSE, 2012). Esse modelo fornece uma estrutura hierárquica dividida em quatro partes:

- ISO/IEC 9126-1: descreve o *framework* do modelo de qualidade, que explica a relação entre as diferentes abordagens da qualidade, bem como identifica as seis características de qualidade abrangidas pela norma – *funcionalidade*, *confiabilidade*, *usabilidade*, *eficiência*, *manutenibilidade* e *portabilidade* – e as suas respectivas subcaracterísticas nos produtos de software;
- ISO/IEC 9126-2: descreve as métricas externas usadas para medir atributos de qualidade das seis características e subcaracterísticas, definidas na ISO/IEC 9126-1;
- ISO/IEC 9126-3: descreve as métricas internas usadas para medir atributos de qualidade das seis características e subcaracterísticas, definidas na ISO/IEC 9126-1;
- ISO/IEC 9126-4: identifica as métricas usadas para medir os efeitos das características de qualidade com relação ao contexto do usuário. Essa parte abrange quatro características próprias de qualidade: *eficácia*, *produtividade*, *segurança* e *satisfação*.

Dessa forma, o produto de software pode ser avaliado de acordo com três perspectivas: i) suas *características externas*, cujas métricas de avaliação são definidas na segunda parte da norma e têm por objetivo avaliar questões comportamentais e aspectos dinâmicos do produto finalizado; ii) suas *características internas*, cujas métricas de avaliação são definidas na terceira parte da norma e têm como proposta a avaliação de atributos do produto em si; e por fim, iii) *qualidade em uso* que identifica características de qualidade consideradas importantes sob o ponto de vista dos *stakeholders*. Na Figura 2-3 é ilustrada a estrutura da norma ISO/IEC 9126, sendo em (a) as características e subcaracterísticas de qualidade interna e externa, enquanto em (b) as características para qualidade em uso.

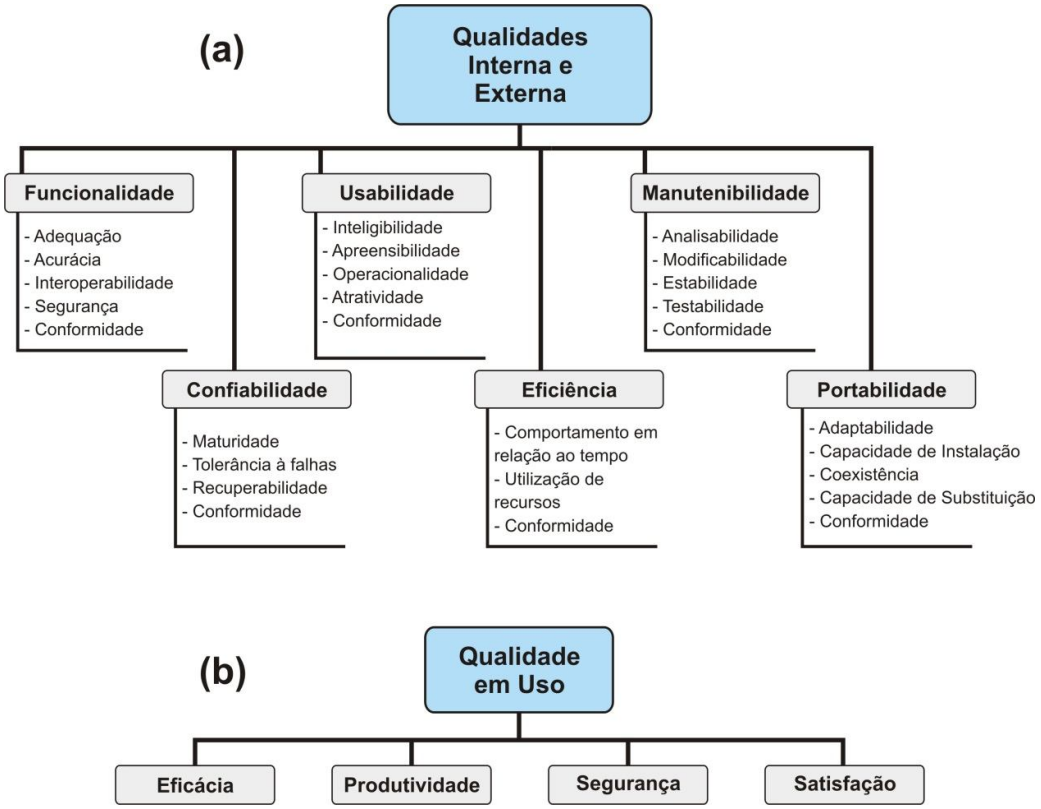

Figura 2-3 – ISO/IEC 9126: (a) características e subcaracterísticas relacionadas às qualidades *internas* e *externas*; (b) características relacionadas à *qualidade em uso*. Adaptado de (ABNT, 2003).

De acordo com Vanderose (VANDEROSE, 2012), o modelo de qualidade da ISO/IEC apresenta dois pontos positivos importantes. O primeiro é identificar características internas e externas de um produto de software, especificando quais atributos devem ser avaliados para que essas características sejam alcançadas. O segundo é oferecer a possibilidade de ser adaptado para diferentes domínios. Contudo, segundo o mesmo autor, esse modelo apresenta a desvantagem de não mostrar claramente como essas características podem ser medidas, dado que não há um processo de avaliação definido na norma. Outra crítica vem do fato que, por se tratar de um padrão internacional, ela deve ser abrangente o suficiente para contemplar os diversos tipos de artefatos produzidos durante o ciclo de vida do desenvolvimento. Essa generalidade pode levar a ambiguidades e sobreposições de termos, como por exemplo: *completude da implementação funcional* e *cobertura da implementação funcional*, além de dificuldades ao se definir quais métricas são mais adequadas para determinados propósitos de qualidade. Outras dificuldades na interpretação dessa norma são descritas por Al-Kilidar *et al.* (2005).

Recentemente a ISO/IEC 9126 foi atualizada e integrada à série ISO/IEC 25000, também conhecida como *Software product QUAlity Requirements and Evaluation* (SQuaRE)

(ISO/IEC, 2005). Essa nova geração do padrão tem por objetivo satisfazer às crescentes necessidades dos usuários por meio de um conjunto melhorado e unificado de documentos normativos que abrangem três processos complementares de qualidade: *especificação de requisitos*, *medição* e *avaliação* (SURYN; ABRAN, 2003), além de incluir o padrão ISO/IEC 14598 como processo de avaliação das características de qualidade descritas pela ISO/IEC 9126.

Além disso, a SQuaRE alterou a hierarquia de algumas das características de qualidade, incluindo *compatibilidade* e *segurança* (esta última, antes parte da qualidade em uso na ISO/IEC 9126) como características internas do produto. *Eficiência* deixou de ser vista como uma qualidade interna para ser considerada uma qualidade em uso, que por sua vez admitiu *ausência de riscos* e *abrangência do contexto* como novas características. Ao mesmo tempo, algumas outras características apenas trocaram sua nomenclatura, por exemplo: *funcionalidade* passou a chamar-se *adequação funcional*; *eficiência* tornou-se *eficiência de desempenho*. Por fim, algumas características passaram a ser classificadas como subcaracterísticas, como por exemplo: *interoperabilidade* tornou-se uma subcaracterística de *compatibilidade*. De qualquer forma, estas alterações não exercem influência significativa na composição deste trabalho, visto que as características de qualidade a serem usadas aqui não seguem um único modelo.

2.2.4 Modelo de Qualidade de Dromey

O modelo descrito por Dromey (DROMEY, 1995) é outra proposta que ajuda a construir uma avaliação operacional da qualidade por intermédio das características do produto. Tecnicamente, o modelo é instanciado por um *framework* de qualidade que, por sua vez, é dividido em três entidades principais: i) um conjunto de atributos de qualidade de alto nível ou metas de qualidade; ii) propriedades do produto que são importantes para alcançar as metas de qualidade, chamadas de propriedades que agregam qualidade (*quality-carrying properties*); e iii) a ligação entre essas propriedades e as metas de qualidade.

Por ser uma abordagem focada, principalmente, em sistemas baseados em componentes, esse modelo identifica, de acordo com Mohagheghi *et al*. (MOHAGHEGHI; DEHLEN; NEPLE, 2008), quatro propriedades de componentes que impactam na qualidade de software:

- *Propriedades de correção*: relacionadas à implantação de componentes e cujas regras não sejam violadas, nem internamente nem quando associadas ao contexto de utilização;
- *Propriedades internas*: descrevem o quão bem um componente é implantado de acordo com suas intenções de uso ou necessidades, cobrindo tanto correção quanto outras propriedades;
- *Propriedades contextuais*: propriedades relacionadas ao posicionamento dos componentes em um determinado contexto;
- *Propriedades descritivas*: propriedades comportamentais que sugerem que requisitos, projetos, implementações e interfaces de usuários devem ser fáceis de entender e usar, considerando os seus propósitos.

Ao contrário das citadas anteriormente, essa abordagem é mais flexível, visto que os critérios de qualidade são definidos e avaliados em relação aos seus propósitos e às posições dos artefatos de software dentro do processo de desenvolvimento (fases do ciclo de vida), o que permite relacionar as metas de qualidade ao contexto do desenvolvimento. Ainda, esse modelo permite identificar quais propriedades do produto são importantes e em quais metas de qualidade elas impactam, dado que o modelo considera que as características de alto nível somente podem ser alcançadas claramente se determinadas propriedades que agregam qualidade forem preenchidas.

O processo de avaliação do modelo de qualidade proposto por Dromey é composto por cinco passos:

1. Identificar o conjunto de atributos de qualidade de alto nível que se deseja avaliar;
2. Identificar e listar os componentes/módulos do sistema que serão submetidos à avaliação;
3. Identificar e classificar as mais significativas e tangíveis *propriedades que agregam qualidade* para cada componente/módulo, isto é, propriedades dos componentes que têm maior impacto na qualidade do produto;
4. Determinar como cada propriedade do produto está relacionada aos respectivos atributos de qualidade;
5. Avaliar o modelo identificando suas forças e fraquezas.

2.2.5 *Framework* de Lange e Chaudron

Outro trabalho que segue a estrutura hierárquica para avaliar a qualidade de artefatos de software foi proposto por Lange e Chaudron (LANGE; CHAUDRON, 2005). Apesar de essa proposta descrever somente a avaliação de modelos UML, ela enfatiza duas ideias importantes: i) que cada artefato tem um propósito específico; e ii) que esse propósito está diretamente relacionado à fase do ciclo de vida do processo ao qual o artefato está inserido. Consequentemente, o modelo de qualidade deve ser ajustado para adequar a avaliação às métricas e aos meios apropriados de medição, semelhante ao que foi apresentado pelo modelo de Dromey.

Assim, esse modelo é estruturado em quatro níveis lógicos. O primeiro considera o *uso* proposto para o artefato a ser avaliado e é classificado em:

- *Operação*: uso que combina características de qualidade que são observadas na execução do produto, considerando que o sistema já está implementado;
- *Transição*: uso que considera as características de qualidade relacionadas ao produto quando este é movido para outro ambiente operacional;
- *Manutenção*: uso relacionado às características de qualidade do produto quando este sofre alterações;
- *Desenvolvimento*: uso que combina características de qualidade relacionadas ao produto e seus artefatos nas fases anteriores ao produto ser finalizado.

O segundo nível do modelo, por sua vez, contém os *propósitos* para os quais os artefatos foram criados, visto que um artefato geralmente não é construído para suprir todos os propósitos ao mesmo tempo, da mesma forma que um propósito pode não ser atendido por um único artefato. Alguns exemplos de propósitos descritos são: *modificação*, *testes*, *compreensão*, *comunicação*, *análise*, *predição*, *implementação* e *geração de código*. Vale lembrar que o modelo foi inicialmente construído para contemplar modelos UML, portanto, os propósitos apresentados pelos autores estão mais intimamente relacionados a essa linguagem. Porém, a ideia de atribuir propósitos aos artefatos em avaliação é genérica e pode ser estendida a outros artefatos de software.

O terceiro nível do modelo de qualidade contém as características inerentes aos artefatos. Em seu trabalho, os autores distinguem entre características de qualidade relacionadas

aos modelos UML e características relacionadas ao sistema produzido, contudo, o modelo não apresenta tal distinção, sendo as características listadas de forma genérica.

Por fim, o quarto nível descreve as métricas e regras relacionadas à avaliação dos artefatos descritos nos níveis anteriores. Na Figura 2-4 é ilustrada a ideia proposta por Lange e Chaudron para arquitetar as características de qualidade considerando o propósito dos artefatos e o motivo pelo qual eles foram criados.

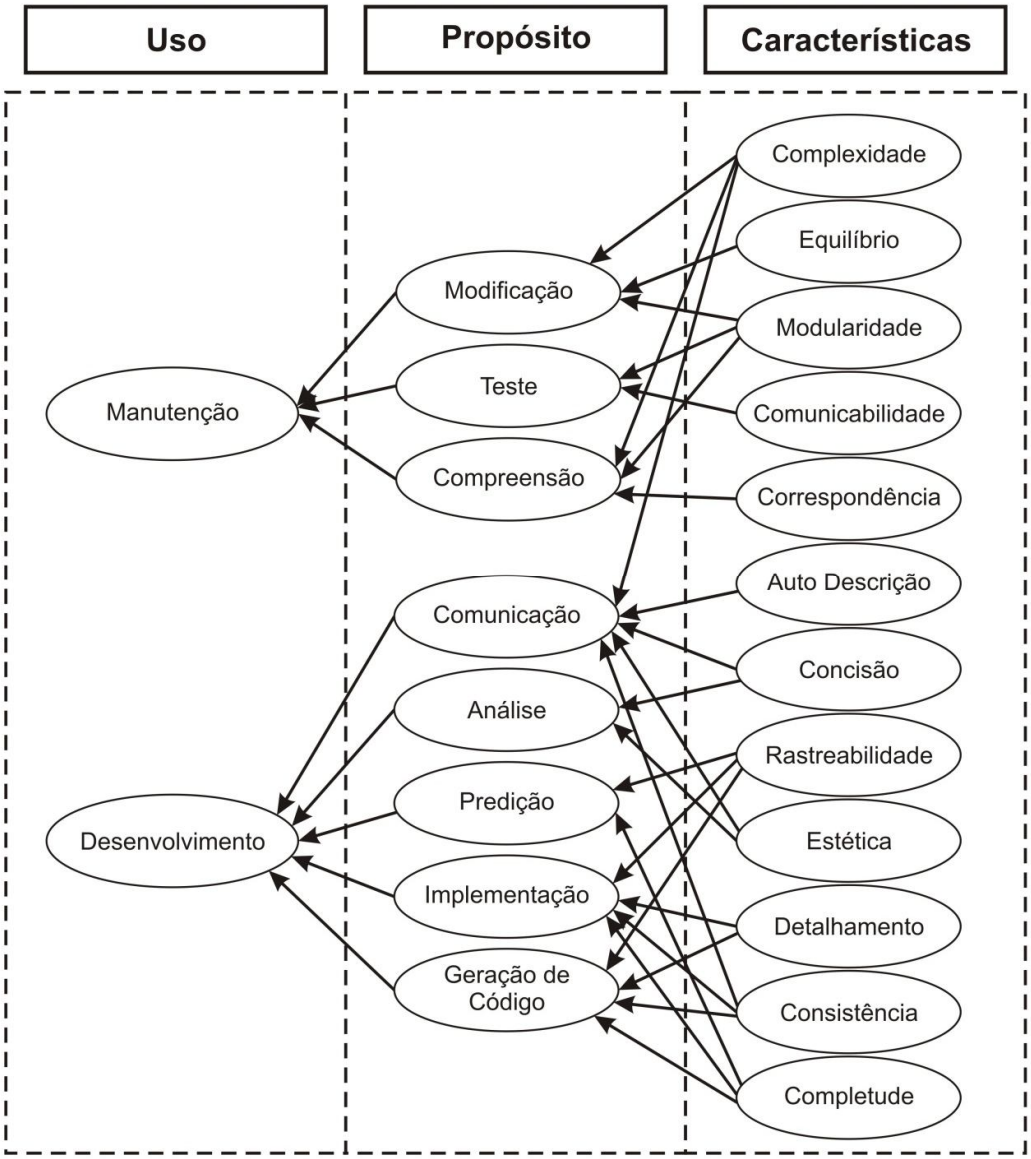

Figura 2-4 – Arquitetura do Modelo de Qualidade proposto por Lange e Chaudron. Adaptado de Lange e Chaudron (LANGE; CHAUDRON, 2005).

A principal contribuição desse modelo é a ideia de considerar os diferentes usos e propósitos dos artefatos na definição das características de qualidade que devem ser avaliadas,

bem como a fase do ciclo de vida do processo em que eles estão inseridos. Dessa forma, os autores argumentam ser possível identificar com maior precisão quais características influenciam, positiva ou negativamente, na qualidade do produto final. Não há um processo de avaliação expressamente definido para o modelo, porém, os autores descrevem uma ferramenta usada para, a partir do *uso* e do *propósito* dos modelos UML, definir as características de qualidade que melhor se adequam.

2.2.6 Comparação entre os Modelos Hierárquicos

Conforme descrito até agora, os Modelos Hierárquicos focam, basicamente, nas características de qualidade dos artefatos de software a serem avaliadas, sendo que o conjunto de características varia para cada modelo. Entretanto, por meio de uma comparação tabular, exibida na Tabela 2.1, pode-se perceber que as características mais comuns são aquelas abordadas pela norma ISO/IEC 9126, especialmente *confiabilidade*, *portabilidade* e *manutenibilidade*, sendo esta última comum a todos os modelos observados.

Paralelamente, tem-se que, enquanto algumas características são abordadas diretamente por alguns modelos, as mesmas são descritas como subcaracterísticas em outros. É o caso da *testabilidade*, tida como uma característica de alto nível nos modelos de Boehm e McCall, porém, descrita como uma subcaracterística de *manutenibilidade* na ISO/IEC 9126 e como um propósito de qualidade no modelo de Lange e Chaudron. Por outro lado, outras características divergem quanto à nomenclatura, mas seguem as mesmas ideias, tais como a *engenharia humana* – influência das condições de trabalho na produtividade – no modelo de Boehm e que segue a maioria dos princípios descritos em *usabilidade* por McCall, ISO/IEC 9126, SQuaRE e Dromey. Por fim, embora o modelo proposto por Lange e Chaudron descreva características incomuns aos outros modelos, a camada *propósito* intenciona alcançar as mesmas ideias de qualidade das outras abordagens.

No que tange a estrutura geral dos modelos, tem-se que o modelo de McCall é preferível para uma abordagem de avaliação ascendente (*bottom-up*), pois a avaliação inicia-se a partir das medidas de qualidade até os fatores de qualidade que, sintetizados, fornecem um panorama geral da qualidade do software, porém, esse modelo torna mais difícil a especificação de requisitos de qualidade.

Tabela 2.1 – Comparação entre os fatores/características de qualidade abrangidos pelos Modelos Hierárquicos.

Fatores/ Características	McCall	Boehm	ISO/IEC 9126	SQuaRE	Dromey	Lange & Chaudron
Manutenibilidade	*	*	*	*	*	*
Flexibilidade	*	*		*		
Testabilidade	*	*	*	*		*
Correção	*					
Eficiência	*	*	*	*	*	
Confiabilidade	*	*	*	*	*	
Integridade	*			*		
Usabilidade	*	*	*	*	*	
Portabilidade	*	*	*	*	*	
Reusabilidade	*			*	*	
Interoperabilidade	*		*	*		
Compreensibilidade		*				*
Funcionalidade			*	*	*	
Desempenho			*	*		
Segurança			*	*		
Comunicabilidade		*				*
Eficácia			*	*		
Satisfação			*	*		
Compatibilidade				*		
Eficiência			*	*		
Ausência de Riscos				*		
Abrangência do Contexto				*		
Complexidade						*
Equilíbrio						*
Modularidade						*
Correspondência						*
Auto Descrição						*
Concisão						*
Rastreabilidade						*
Estética						*
Detalhamento						*
Consistência						*
Completude						*

O modelo de Boehm, em contrapartida, é preferível para avaliações descendentes (*top-down*), mas seu suporte, segundo Samadhiya *et al*. (SAMADHIYA; WANG; CHEN, 2010), é muito efêmero por não possuir um processo de avaliação definido. A decomposição desse modelo direciona as características de alto nível aos usuários finais, enquanto as características de baixo nível são direcionadas à equipe de desenvolvimento.

A ISO/IEC 9126, juntamente com a SQuaRE, por serem modelos construídos graças à colaboração de diversas entidades internacionais, são tidos como os modelos de qualidade mais consistentes entre os apresentados, abrangendo tanto as características internas do produto quanto as características relacionadas às expectativas dos usuários.

Por fim, os modelos de Dromey e Lange e Chaudron caracterizam-se por serem modelos que reconhecem as diferenças relacionadas à avaliação da qualidade para cada produto e possuem uma ideia mais dinâmica para modelagem de processos, capazes de serem aplicados em diversos sistemas. Dromey foca na relação entre características, atributos e propriedades que agregam qualidade. Lange e Chaudron estabelecem uma relação direta entre as características de qualidade, seus propósitos e como elas podem ser usadas. Ambos os modelos se diferenciam dos anteriores por possuírem uma maior preocupação contextual com os artefatos de software a serem avaliados.

Em relação à estrutura dos modelos hierárquicos, muitas críticas têm sido apresentadas em várias publicações, entre elas a de que essa abordagem, de modo geral, falha ao não estabelecer uma definição ampla, precisa e aceitável sobre qualidade, principalmente porque mistura critérios de diferentes dimensões, causando, muitas vezes, sobreposições ou mau-entendimento em relação à definição de características e subcaracterísticas de qualidade. Além disso, ao assumir um conjunto estático de fatores de qualidade e suas relações, esses modelos comprometem a flexibilidade da avaliação.

Outro problema descrito está na falta de transparência imposta pela arquitetura desses modelos, visto que eles não fornecem uma análise racional de apoio e, em alguns casos, não descrevem como fatores de qualidade de alto nível são decompostos em subfatores e métricas. Ou seja, esses modelos não seguem uma estratégia definida e, portanto, podem ser considerados arbitrários (DEISSENBOECK et al., 2009; MOHAGHEGHI; DEHLEN, 2008a; WAGNER; DEISSENBOECK, 2007). Finalmente, tem-se que o nível de reusabilidade do conhecimento adquirido e armazenado sobre qualidade depende fortemente da similaridade entre projetos passados e futuros, o que é geralmente limitado pela falta de indicadores que relacionem essa similaridade (TRENDOWICZ; PUNTER, 2003).

Por outro lado, os Modelos Hierárquicos, por serem os primeiros modelos de qualidade apresentados, tiveram um papel decisivo no que concerne a avaliação da qualidade de produtos de software. Isso porque eles fornecem uma visão compreensiva de quais requisitos não funcionais (relacionados às necessidades implícitas) deveriam ser avaliados e, além disso, qual a relação desses requisitos com seus métodos de avaliação. Essa ideia serviu de ponto de partida para a construção de outros modelos mais complexos e abrangentes, ou mesmo no aperfeiçoamento de alguns deles, como é o caso do padrão SQuaRE que absorveu, entre outros, as características de qualidade definidas pela ISO/IEC 9126 com o processo de avaliação descrito pela ISO/IEC 14598.

Em relação a este trabalho, os modelos de qualidade descritos acima nortearam alguns aspectos importantes, tais como: i) a relação das características e subcaracterísticas de qualidade apresentada pela ISO/IEC 9126; ii) a avaliação de artefatos de software em relação às fases do ciclo de vida, conforme aborda o modelo de Dromey, corroborado por Lange e Chaudron; e, iii) a necessidade de especificação de propósitos de qualidade para os artefatos avaliados, conforme introduz Lange e Chaudron.

Outros exemplos de modelos hierárquicos são o FURPS (GRADY; CASWELL, 1987) e o modelo da IEEE (IEEE, 2009). Ambos possuem uma arquitetura muito similar ao que foi apresentado até aqui. A diferença é que o primeiro modelo separa as características de qualidade em funcionais e não-funcionais e o seu processo de avaliação permite a priorização das características mais importantes para o contexto. O segundo modelo, por sua vez, considera fatores ambientais organizacionais, padrões requeridos, regulamentações ou leis na composição e definição das características que devem ser abordadas pelo processo de avaliação. Outras comparações entre abordagens hierárquicas podem ser encontradas em Al-Qutaish (AL-QUTAISH, 2010), Breivold e Crnkovic (BREIVOLD; CRNKOVIC, 2009), Malhotra e Pruthi (MALHOTRA; PRUTHI, 2012), Samadhiya *et al.* (SAMADHIYA; WANG; CHEN, 2010), Singh e Kannojia (SINGH; KANNOJIA, 2013) e Singh e Chawla (SINGH; CHAWLA, 2012).

2.3 Modelos Conceituais de Qualidade

Os modelos descritos até aqui apresentam a definição das características de qualidade por meio de uma estrutura hierárquica decomposta de um nível alto de abstração até subcaracterísticas e métricas relacionadas. A avaliação pode ser feita tanto usando abordagens *top-down* quando *bottom-up*. Porém, a maioria dos modelos restringe-se às etapas de implementação ou manutenção do desenvolvimento.

Contudo, a busca pela redução do retrabalho e, consequentemente, a redução de custos dos projetos de software tem levado a um aumento da participação dos usuários no desenvolvimento desses projetos. Com isso, surge a ideia de representar propriedades funcionais e não-funcionais de um domínio do problema já no início do processo, com o objetivo de verificar e avaliar suas intenções quanto ao produto final. Em outras palavras, é durante a análise de requisitos que a noção de software em desenvolvimento se mostra mais forte (MOODY et al., 2003). Essa etapa é normalmente composta pela modelagem do sistema em desenvolvimento.

Com este propósito, os *Modelos Conceituais de Qualidade* são uma das abordagens usadas para avaliação tanto do processo como do ambiente de desenvolvimento como um todo (MEHMOOD; CHERFI; COMYN-WATTIAU, 2009). Porém, essa abordagem permite não somente a avaliação dos artefatos, como também do conhecimento da equipe de desenvolvimento, do domínio modelado, das linguagens de modelagem, dentre outros aspectos que envolvem o processo de construção de software. Isso permite uma visão holística a respeito do que se pretende construir e da qualidade esperada para o produto final, tais como *eficiência*, *eficácia*, além de *estabilidade*, *precisão* e *rastreabilidade* (BOMMEL et al., 2007). O resultado é uma representação que contém as informações críticas, necessárias para a concepção e aplicação de estratégias organizacionais eficazes, processos e gestão do conhecimento, além da base teórica necessária para a construção de bases de conhecimento (NELSON et al., 2011).

Nesta seção são apresentas duas das mais importantes propostas a cerca de Modelos Conceituais: os *frameworks* propostos por Lindland *et al.* e Krogstie *et al.*, além de descrever sua influência na busca por artefatos de software que contemplem os requisitos de qualidade desejados.

2.3.1 *Framework* de Lindland *et al.*

Um dos primeiros Modelos Conceituais de Qualidade apresentados foi o *framework* introduzido por Lindland *et al.* (LINDLAND; SINDRE; SOLVBERG, 1994). Esse *framework* é baseado na teoria semiótica (teoria dos sinais proposta por Morris (MORRIS, 1938)) e foi desenvolvido para avaliar conceitualmente modelos de vários propósitos, tais como modelos de dados, modelos de processos e modelos de negócio (MOODY et al., 2003). Contudo, a proposta pode ser ajustada a outros tipos de artefatos, entre eles, códigos fonte e documentação, uma vez que a principal ideia proposta por Lindland *et al.* é a clara separação entre as *metas de qualidade*, que são os fatores de qualidade almejados para um determinado produto de software, dos *meios* propostos para que sejam alcançados. Assim, o *framework* distingue três conceitos iniciais de qualidade: *qualidade sintática*, *qualidade semântica* e *qualidade pragmática*, como ilustrado na Figura 2-5, que são:

- *Qualidade Sintática*: relaciona o modelo à linguagem de modelagem/programação usada, verificando o quão bem um determinado modelo está de acordo com a sintaxe dessa linguagem. Ou seja, todas as declarações de um modelo devem estar de acordo com a sua sintaxe. Existem três meios: prevenção, detecção e correção de erros. Os dois primeiros são preventivos e mais fáceis de serem automatizados; já a correção é corretiva e dificilmente pode ser automatizada completamente, necessitando de atividades de verificações formais da sintaxe com o auxílio de ferramentas;
- *Qualidade Semântica*: relaciona o modelo ao domínio, verificando o quão bem um determinado modelo corresponde ao domínio que descreve. A falta desse tipo de qualidade acontece quando falta no modelo algo que existe no domínio ou quando o modelo contém algo que não está presente no domínio. A principal atividade usada para alcançar essa meta é a verificação da consistência entre modelos que descrevem o mesmo domínio. Porém, caso haja alguma inconsistência, como por exemplo, regras conflitantes, o *framework* estabelece como práticas a exclusão de uma delas ou a inserção de outra que resolva o conflito;
- *Qualidade Pragmática*: relaciona o modelo à compreensão do usuário, isto é, os modelos construídos para esse propósito devem possuir características que facilitem o entendimento do domínio pelo usuário. Por exemplo, executabilidade, poucas

expressões e boa estruturação, tendo como atividades relacionadas aquelas que facilitem a compreensão do domínio (inspeção, execução, simulação, entre outras).

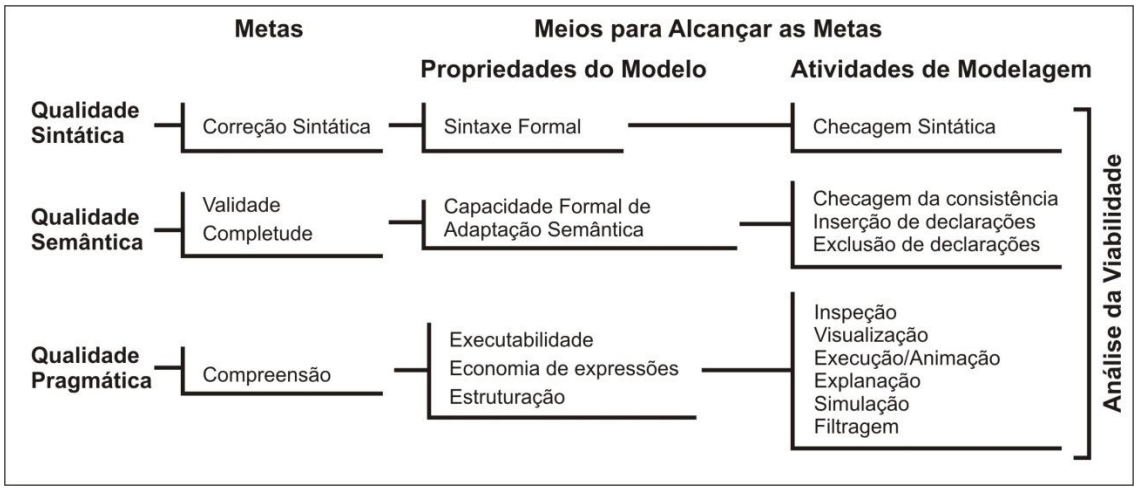

Figura 2-5 – *Framework* de qualidade proposto por Lindland *et al.* Adaptado de Lindland *et al.* (LINDLAND; SINDRE; SOLVBERG, 1994)

O *framework* avalia, para cada uma das visões de qualidade, o propósito, separando os modelos em propriedades e atividades de modelagem. Porém, isso não significa que certa propriedade ou atividade contribua exclusivamente para uma única visão. Por exemplo: melhorar a qualidade pragmática pode melhorar a qualidade semântica, visto que tornando mais fácil a compreensão do domínio, torna-se mais fácil também a detecção de inconsistências ou incompletudes. Ainda, melhorar a sintaxe pode melhorar a qualidade pragmática pelo fato de tornar o modelo mais compreensível.

Contudo, segundo Lindland *et al.*, alcançar totalmente características de validade e completude requer uma quantidade de tempo e dinheiro ilimitada, tornando inaceitável, e talvez impossível, essa tentativa. Assim, a medida dessas qualidades é aceitar o modelo quando os benefícios resultantes de mais verificações são insignificantes diante do seu estado atual. Em vista disso, os autores incluíram o conceito de *análise da viabilidade*, que busca responder duas questões importantes: i) o modelo (ou artefato) alcançou uma qualidade satisfatória? ii) se não, qual meta de qualidade é mais importante para o momento, para o público alvo e qual é o meio mais eficaz para alcançar essa meta?

Essa mesma ideia de viabilidade também foi descrita por Bach em uma série de três artigos (BACH, 1996, 1997, 1998) que afirmam que um artefato não precisa ser necessariamente livre de erros, mas deve atingir um grau de confiabilidade que satisfaça às necessidades dos usuários e do sistema. Segundo os autores, a noção de viabilidade depende exclusivamente da interpretação humana.

2.3.2 Abordagem SEQUAL

O trabalho de Lindland *et al.* foi posteriormente estendido e refinado por Krogstie *et al.* (KROGSTIE; LINDLAND; SINDRE, 1995a; KROGSTIE, 2003) em uma abordagem chamada SEQUAL (*SEmiotic QUALity*) para cobrir outras camadas conceituais – *qualidade semântica percebida* (Figura 2-6). Esse tipo de qualidade é definida como a correspondência entre as informações que os usuários pensam que existe (Interpretação do Usuário) e as informações que os usuários pensam que deveriam existir no modelo/artefato, baseado no conhecimento que possuem sobre o domínio do problema (Conhecimento do Domínio). A qualidade semântica percebida serve como um substituto operacional da qualidade semântica, visto que não requer verificação da correspondência entre o *modelo* e o *domínio*, mas entre o *conhecimento* que os usuários têm do modelo (ou seja, Interpretação do Usuário) e o *domínio* (isto é, Conhecimento do Domínio) (MAES; POELS, 2007).

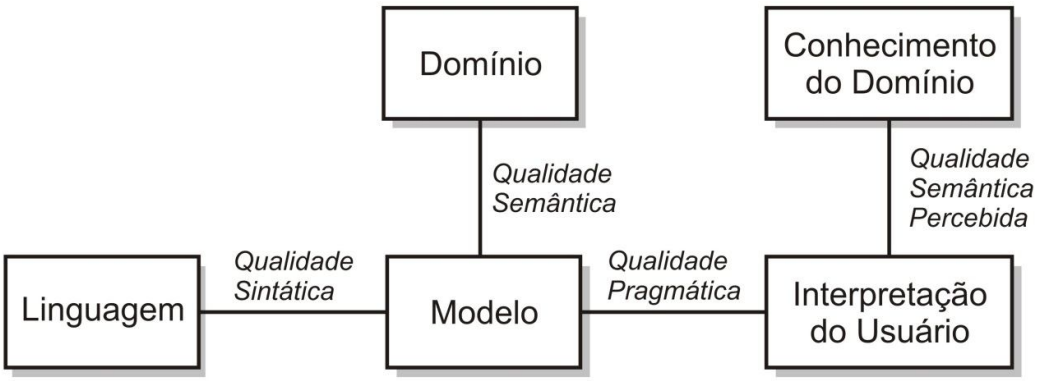

Figura 2-6 – Representação conceitual do *framework* de Krogstie *et al.* (1995). Adaptado de Maes *et al.* (MAES; POELS, 2007).

O argumento para o ajuste dessa nova abordagem vem do fato que o *framework* de Lindland *et al.* é muito estático na sua visão sobre qualidade semântica, principalmente por considerar somente os modelos e não as atividades de modelagem, além de comparar os modelos de um domínio específico. Além disso, a visão de qualidade pragmática foca somente no entendimento, sem considerar como o modelo é usado e qual o seu impacto na visão dos usuários. Dessa forma, o *framework* proposto por Krogstie *et al.* pode representar como as perspectivas de qualidade, consideradas relevantes para um determinado domínio, podem ser alcançadas por meio de tarefas de modelagem.

Embora a proposta nesta dissertação não seja especificamente relacionada à modelagem de sistemas e/ou de domínios, a abordagem conceitual de Krogstie *et al.* dá a dimensão de como

cada *stakeholder* percebe a qualidade de um produto de software e, consequentemente, quais os caminhos mais apropriados para que essas perspectivas possam ser alcançadas por meio de métricas e métodos de avaliação. Além disso, Shanks e Darke (SHANKS; DARKE, 1997) argumentam que a garantia da qualidade na modelagem conceitual necessita atender tanto o *processo* – as tarefas de aquisição e modelagem dos requisitos do sistema – quanto o *produto* – o resultado do modelo conceitual.

Como um exemplo, tem-se o conhecimento sobre uma determinada linguagem (tanto de modelagem quanto de programação). Este conhecimento é determinante nas qualidades sintática e semântica do artefato produzido, pois, quanto mais conhecimento um desenvolvedor tiver sobre a linguagem usada, melhor será a sua externalização sobre a percepção da realidade.

Dessa forma, segundo Bommel *et al*. (BOMMEL et al., 2007), alguns aspectos de qualidade podem ser mais bem alcançados por intermédio de um bom processo de modelagem, mais que apenas impondo regras e restrições ou por meio de testes no produto final. Além disso, modelos e códigos-fonte não são os únicos produtos de um processo de desenvolvimento de software, há também outros, como o conhecimento adquirido pelos desenvolvedores, acordos, documentações, protótipos, entre outros, que são tão importantes quanto.

Krogstie *et al.* argumentam, ainda, que a maioria das perspectivas de qualidade depende do conhecimento e do esforço de cada envolvido durante o desenvolvimento de um artefato de software, e que este conhecimento não é estático e pode ser melhorado e educado ao longo do processo.

Em vista disso, pode-se usar diretamente os conceitos SEQUAL para expressar metas subjetivas de qualidade; sejam elas, metas relacionadas aos produtos finais ou mesmo aos artefatos. Em outras palavras, é possível associar as metas e submetas de qualidade em vista das perspectivas dos interessados com o objetivo de definir a noção de benefício em relação aos custos envolvidos para atingi-las. Note-se que o custo pode ser tanto em termos financeiros quanto de tempo, e que pode ser calculado em vista de um processo de trabalho, das pessoas e dos recursos envolvidos.

2.3.3 Considerações sobre os Modelos Conceituais

Embora os modelos conceituais possuam um alto nível de abstração e, portanto, sejam difíceis de serem aplicados na prática, eles podem ser uma boa alternativa para formalizar intenções de qualidade, além de contribuírem para o entendimento de questões de qualidade em relação à modelagem conceitual, ou seja, o modelo torna-se realmente uma representação formal de um sistema organizacional. Mais do que isso, a abordagem SEQUAL pode ser usada não somente para expressar o quão bem um processo deve ser desenvolvido como um todo para alcançar suas metas, mas também quais os passos específicos que contribuem para alcançar metas específicas de qualidade. Dessa forma, as metas de qualidade podem ser identificadas por meio de níveis semióticos, como: *correção sintática*, *viabilidade de compreensão* e *viabilidade de concordância* que estão relacionadas, respectivamente, às visões *sintáticas*, *pragmática* e *semântica*. Uma descrição mais completa e aprofundada dos elementos que compõe a abordagem conceitual pode ser encontrada em Bommel *et al*. (BOMMEL et al., 2007), Maes e Poels (MAES; POELS, 2007), Moody *et al.* (MOODY, 2003) e Nelson *et al.* (NELSON et al., 2011).

2.4 Modelos de Qualidade Adaptados para Contextos Específicos

Seguindo as duas abordagens descritas anteriormente, Modelos Hierárquicos e Modelos Conceituais, outros trabalhos foram propostos para melhorar a qualidade de produtos e processos de software ao longo do tempo. Esses trabalhos abordam a avaliação da qualidade em contextos específicos, normalmente, por meio de um *framework* instanciado a partir de um metamodelo de qualidade.

Entre eles, Seffah *et al.* (SEFFAH et al., 2006) propõem um modelo para avaliar a usabilidade em interações humano-computador (*Human-Computer Interaction* – HCI) chamada *Quality in Use Integrated Measurement* (QUIM) associando fatores de qualidade com critérios e métricas. O objetivo dessa proposta é facilitar a avaliação das interfaces de software para que tanto especialistas quanto não-especialistas possam diagnosticar problemas de usabilidade e estética. Contudo, mais que apenas um exemplo, essa ideia descreve que a avaliação da qualidade vai além da especificação de métricas. Isto é, o modelo deve relacionar

os dados de forma que possam ser facilmente interpretados para que a equipe de desenvolvimento tenha os parâmetros necessários para tomada de decisão em prol da melhoria da qualidade dos produtos avaliados.

Ao mesmo tempo, Mohagheghi *et al.*, por intermédio de uma série de trabalhos (MOHAGHEGHI; DEHLEN; NEPLE, 2008, 2009; MOHAGHEGHI; DEHLEN, 2007, 2008a, 2008b, 2009), descrevem a avaliação de modelos UML tendo como objetivo o desenvolvimento baseado na abordagem Engenharia Dirigida por Modelos (*Model-Driven Engineering* – MDE). Nesses trabalhos eles propõem a avaliação de seis fatores de qualidade em modelos – que nesta abordagem podem ser considerados como artefatos de software – sendo eles: *correção*, *completude*, *consistência*, *compreensibilidade*, *restrição* e *suporte a mudanças*. Contudo, a principal contribuição está na avaliação da relação entre os propósitos dos modelos e as expectativas dos *stakeholders*. Além disso, essa proposta considera tanto as características de qualidade e métricas abordadas pelos Modelos Hierárquicos, quanto a descrição dos fatores contextuais relacionados aos Modelos Conceituais. Ou seja, a proposta estabelece uma relação entre as características de qualidade e métricas de avaliação e o mundo real (domínio do problema e organização), como pode ser visto na Figura 2-7.

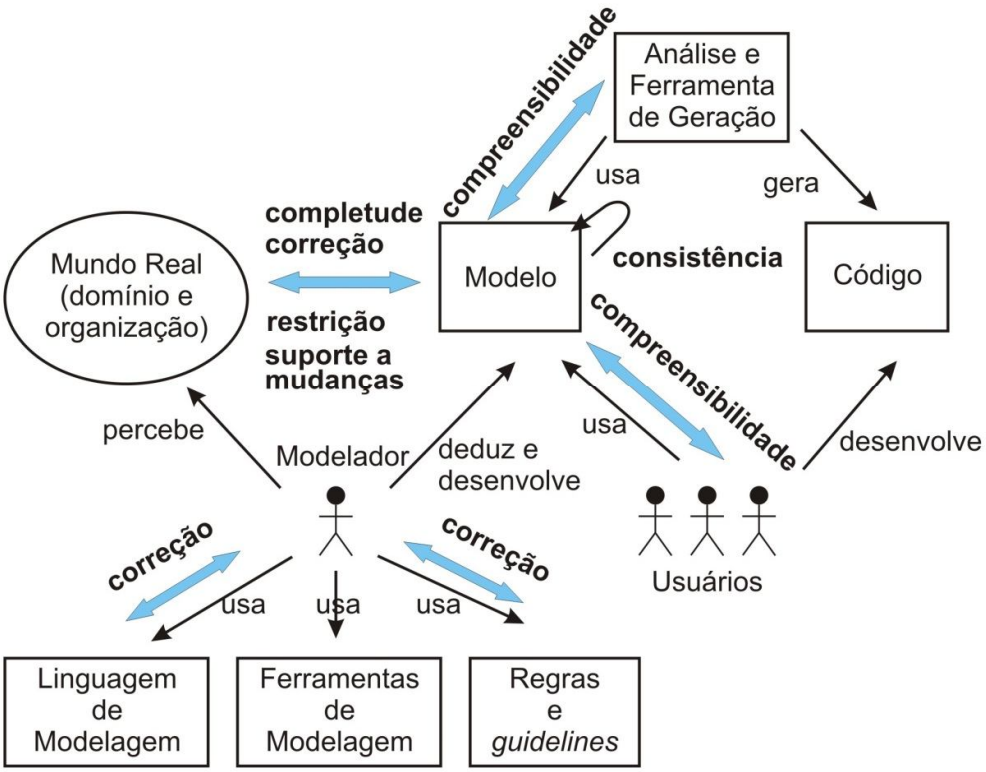

Figura 2-7 – *Framework* de qualidade proposto por Mohagheghi e Dehlen (MOHAGHEGHI; DEHLEN, 2009).

Outra proposta envolvendo avaliação da qualidade e a abordagem MDE é descrita por Solheim e Neple (SOLHEIM; NEPLE, 2006). Em seu trabalho, os autores simplificaram e adaptaram as ideias dos Modelos Conceituais para identificar *transformabilidade* e *manutenibilidade* como duas metas de qualidade importantes em MDE, que são decompostas em várias características. Domínguez-Mayo *et al*. (DOMINGUEZ-MAYO et al., 2010), por sua vez, descrevem um *framework* de avaliação de modelos para a metodologia *Model-Driven Web Engineering* (MDWE).

Em comum, nos trabalhos envolvendo *qualidade* e a abordagem *model-driven*, os autores incluem a avaliação das linguagens de modelagem usadas, a qualidade das ferramentas de modelagem e transformação, o conhecimento dos desenvolvedores (ou modeladores) em relação ao problema abordado, a qualidade do processo de modelagem, e a garantia das técnicas aplicadas para descobrir falhas e fraquezas nos modelos.

Paralelamente, *frameworks* de qualidade também podem ser usados para avaliar artefatos usados em *Software Product Lines* (SPL), como descreve Trendowicz e Punter (TRENDOWICZ; PUNTER, 2003). Nesse trabalho, os autores propõem um método orientado a metas, chamado *Prometheus*, que integra abordagens qualitativas e quantitativas para controle de qualidade em SPL. Os autores defendem, ainda, que a avaliação dos produtos deve iniciar o quanto antes dentro do ciclo de vida de desenvolvimento, respeitando-se os seus propósitos. Além disso, o *framework* de avaliação deve ser: *flexível* ao ponto de poder ser ajustado para projetos e organizações específicas, *transparente* para permitir uma visão clara sobre seus fundamentos, bem como o significado de cada característica e a relação entre elas, e, *reusável* para poder ser melhorado e reaproveitado em outros projetos.

2.5 Modelos de Processos de Avaliação da Qualidade de Softwares

Nesta seção são apresentados alguns processos usados para avaliação de produtos de software disponíveis na literatura, entre os quais, as abordagens GQM, TAM e algumas propostas alternativas descritas em trabalhos publicados. Além disso, é abordada a norma ISO/IEC 14598 e sua utilização como referência para compor o processo de avaliação proposto nesta dissertação e descrito no Capítulo 4.

2.5.1 *Goal Question Metric*

Entre os diversos processos de avaliação de qualidade de software pesquisados, alguns são bastante referenciados e, portanto, consolidados. Esse é o caso da abordagem *Goal Question Metric* (GQM) proposta por Basili *et al.* (BASILI; CALDIERA; ROMBACH, 1994). Essa abordagem define uma avaliação *top-down* estruturada de forma hierárquica na qual é definido um conjunto de metas a serem alcançadas e, com base nisso, refinadas em várias questões com o propósito de coletar os dados necessários para a avaliação. Esses dados são então submetidos a um conjunto de métricas cujo objetivo é avaliar se a qualidade do objeto a ser medido está dentro das especificações desejadas. As três etapas são classificadas, respectivamente, como: nível conceitual (*goal*), nível operacional (*question*) e nível quantitativo (*metric*). O resultado da aplicação da abordagem GQM é a especificação de um sistema de medição que objetiva um conjunto particular de propósitos e regras para a interpretação e medição dos dados (BASILI; CALDIERA; ROMBACH, 1994).

Embora essa seja uma abordagem bastante completa, capaz de abranger muitos dos aspectos necessários para uma avaliação de artefatos de software, tais como: metas de qualidade, propósitos, pontos de vista, artefatos (objetos) e, evidentemente, métricas, ela é estruturada hierarquicamente de forma que as métricas são relacionadas diretamente com os propósitos. Dessa forma, segundo Mohagheghi (2010), a abordagem não define explicitamente como os métodos de avaliação se relacionam com cada métrica. Além disso, GQM aborda o problema da identificação de métricas quando os objetos são predefinidos, ou seja, quando as respostas para as questões já são conhecidas de antemão. Assim, pode-se dizer que GQM é uma abordagem útil quando se tem metas claras para a introdução de uma metodologia ou produto; por exemplo, ao avaliar o impacto de uma atividade de melhoria de processo de software. No entanto, torna-se difícil de aplicar quando as metas não são explícitas ou o impacto de um novo método é desconhecido (MOHAGHEGHI, 2010).

Outro ponto importante está no fato da abordagem GQM não endereçar a ligação entre as metas medidas e os contextos que envolvem as diversas abordagens de desenvolvimento e/ou os contextos organizacionais de um projeto. Contudo, uma nova versão, chamada *GQM*$^{+}$ *Strategies*® (BASILI et al., 2007), foi proposta com o intuito de incorporar requisitos e estratégias organizacionais de alto nível e, assim, suprir essa deficiência.

Todavia, alguns dos trabalhos estudados nesta pesquisa relataram o uso, com sucesso, da abordagem GQM como forma de avaliação de suas propostas. Entre eles Berti-Équille *et al.* (BERTI-ÉQUILLE et al., 2011), Bouwers *et al*. (BOUWERS; VAN DEURSEN; VISSER, 2013), Cachero *et al.* (CACHERO; CALERO; POELS, 2007), Kläs *et al.* (KLÄS; LAMPASONA; MÜNCH, 2011; KLÄS et al., 2010) e, Vanderose e Habra (VANDEROSE; HABRA, 2011).

2.5.2 *Technology Acceptance Model* (TAM)

A exemplo da abordagem GQM, outras propostas para avaliação de produtos de software, porém não tão populares, podem ser encontradas na literatura. Entre elas, a abordagem *Technology Acceptance Model* (TAM), originalmente desenvolvida por Davis (DAVIS, 1989) e estendida por outros, é um modelo genérico que pode ser aplicado para medir a aceitação e infusão de uma tecnologia ou sistema. O modelo original foi proposto para explicar as intenções dos usuários no uso de novos sistemas por meio de duas percepções: *percepção de utilidade* e *percepção de facilidade de uso*. O processo é focado em questionários cujo propósito é avaliar características internas de qualidade dos produtos, tais como: robustez, consistência, confiabilidade. Porém, as críticas a respeito desta abordagem apontam-na como essencialmente teórica e de heurística questionável, além do limitado poder preditivo, trivialidade e falta de qualquer valor prático (MOHAGHEGHI, 2010).

2.5.3 ISO/IEC 14598

Normalmente, o foco primário de uma avaliação da qualidade em produtos de software não é o processo de avaliação, mas sim, a aplicação de uma ferramenta ou técnica específica ou a descrição de um procedimento de trabalho (PUNTER et al., 2004). Porém, uma referência explícita para a definição de um processo de avaliação tende a tornar a visão da qualidade mais consistente e gerenciável. Em vista disso, uma publicação que foca explicitamente no processo de avaliação é o padrão ISO/IEC 14598 (*Software engineering – product evaluation*) (ISO/IEC, 1999).

Esse padrão é composto por um conjunto de seis partes: Parte 1 – apresenta uma visão geral da estrutura da série de normas e dos processos de avaliação; Parte 2 – descreve o planejamento e gestão do processo de avaliação; Parte 3 – aborda as atividades de avaliação durante o processo de desenvolvimento do software (perspectiva dos desenvolvedores); Parte 4 – aborda as atividades de avaliação no processo de seleção para aquisição de software (perspectiva dos adquirentes); Parte 5 – determina o ciclo de vida da avaliação, com definição das atividades, incluindo as relações entre avaliador e cliente (perspectiva dos avaliadores); e Parte 6 – especifica os pacotes estruturados de métodos e ferramentas para apoio das partes relacionadas. Paralelamente, o processo de avaliação definido pela norma é dividido em quatro fases com subfases relacionadas, conforme ilustrado na Figura 2-8. As fases definidas pela ISO/IEC 14598 são:

- *Estabelecer os requisitos de avaliação*: durante esta fase, os requisitos para a avaliação são estabelecidos com base nas características de qualidade definidas pela ISO/IEC 9126. Níveis de avaliação são usados para expressar a importância das características com base nos seus usos e riscos associados. Três atividades são realizadas durante esse subprocesso: i) estabelecer o propósito da avaliação; ii) identificar tipos de produtos a serem avaliados; e, iii) especificar o modelo de qualidade.
- *Especificar a avaliação*: durante essa segunda fase, as métricas para a avaliação devem ser selecionadas, e seus valores de aceitação estabelecidos, tendo como base os requisitos especificados na primeira etapa. Além disso, são definidos, também, quais os critérios para julgamento tendo em vista a criticidade do produto em avaliação. Os subprocessos definidos nessa fase são: i) selecionar métricas (provenientes da ISO/IEC 9126); ii) estabelecer níveis de pontuação para as métricas; iii) estabelecer critérios para a avaliação.
- *Projetar a avaliação*: essa fase está relacionada à documentação dos procedimentos que serão utilizados pelo avaliador para realizar a medição dos produtos, os recursos necessários, a especificação de quem fará a avaliação e quais os métodos que serão usados, bem como as diferentes atividades que deverão ser executadas durante a avaliação. Como resultado, a saída dessa fase é um plano que descreve o procedimento de avaliação e o cronograma das ações do avaliador. O único subprocesso definido nessa fase é: produzir o plano de avaliação.

- *Executar a avaliação*: é nessa fase que a avaliação realmente é realizada por meio da execução das medições. Ou seja, os valores para cada métrica estabelecida são coletados e, então, interpretados com base nas especificações dos valores limites estabelecidos na segunda fase. Por fim, um relatório de avaliação é produzido. Os subprocessos definidos são: i) obter as medidas; ii) comparar com critérios; iii) avaliar os resultados.

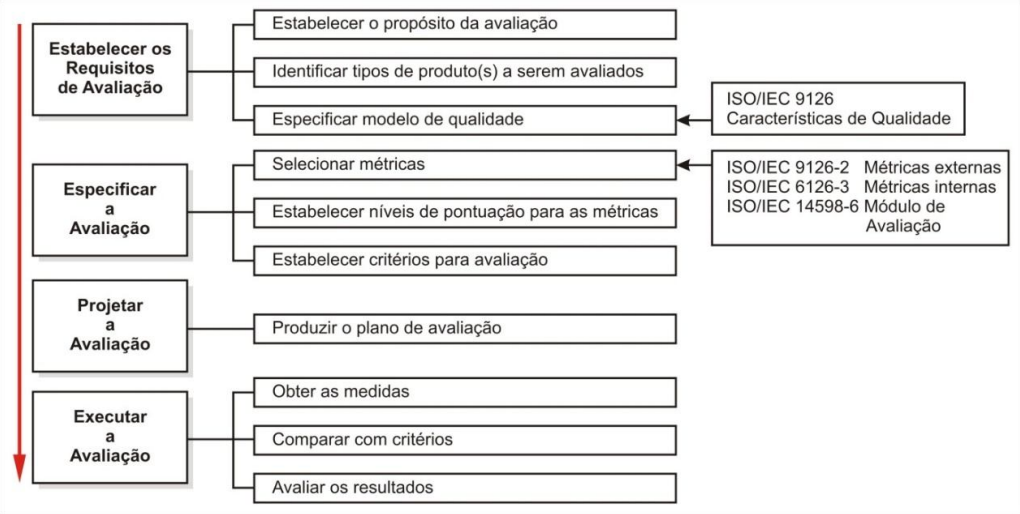

Figura 2-8 – Visão geral do processo de avaliação definido pela ISO/IEC 14598. Adaptado de (PUNTER et al., 2004).

2.5.4 Processos Alternativos

Por fim, alguns trabalhos definem suas próprias sequências de passos para compor a avaliação dos produtos. Esse é o caso, por exemplo, do trabalho de Dominguez-Mayo *et al.* (DOMINGUEZ-MAYO et al., 2010) para a avaliação de artefatos usadas na abordagem MDWE; Trendowicz e Punter (TRENDOWICZ; PUNTER, 2003) para a avaliação de artefatos usados na composição de Linhas de Produtos de Software; e Kläs *et al.* (KLÄS; LAMPASONA; MÜNCH, 2011) para avaliar modelos. Embora sejam definidos pelos próprios autores, esses processos seguem uma sequência de passos semelhantes, que envolve a identificação dos artefatos a serem medidos; a escolha de determinadas características de qualidade; a escolha das métricas para cada característica; e, uma proposta hipotética de operacionalização do modelo definido.

2.6 Considerações

Neste capítulo foram apresentados, inicialmente, alguns conceitos referentes à qualidade de software. De modo geral, a literatura os descreve segundo duas perspectivas. A primeira delas relaciona a conformidade dos produtos e/ou serviços aos requisitos especificados no projeto e cujas características possam ser medidas. A segunda, por outro lado, é subjetiva e visa avaliar a satisfação dos usuários quanto ao produto final, com base em quesitos como *utilidade*, *facilidade de uso*, *eficácia*, entre outros. Contudo, existe a necessidade de se formalizar um processo de avaliação, independentemente da natureza dos requisitos a serem validados.

Consequentemente, a abordagem de Modelos de Qualidade foi escolhida para nortear este trabalho, visto que oferece uma boa estruturação dos fatores e metas de qualidade, permitindo a sua decomposição em subfatores e métricas de forma a facilitar a compreensão e uso destes modelos, além de serem capazes de avaliar quantitativamente diversos atributos de qualidade relacionados a fatores qualitativos.

Como abordado neste capítulo, diversas propostas de modelos de qualidade surgiram ao longo do tempo, descrevendo conceitos importantes sobre avaliação de artefatos de software. Todas essas propostas possuem vantagens e desvantagens. Porém, alguns dos conceitos descritos vão ao encontro da ideia deste trabalho, tais como: i) a decomposição de características de qualidade em subcaracterísticas e métricas; ii) a diferenciação entre *metas de qualidade* e *meios* usados para alcançá-las; iii) a avaliação dos artefatos de software segundo os seus propósitos e o ciclo de vida do processo no qual estão inseridos; e por fim, iv) a avaliação adaptada ao contexto organizacional e do projeto.

Embora todas essas premissas façam parte dos modelos apresentados nas seções anteriores, elas estão fragmentadas entre eles. Assim, nos próximos capítulos é descrito o *framework* de qualidade proposto neste trabalho como alternativa para a avaliação de artefatos de software tendo em vista os conceitos apresentados até agora. Esse *framework* é composto pela definição dos elementos que compõem a estrutura de avaliação, estruturados a partir de um metamodelo, e de um processo de avaliação cujo objetivo é especificar uma sequência de passos capaz de guiar os avaliadores na associação de tais elementos. Além disso, é apresentada, também, uma ferramenta de apoio.

3 METAMODELO DE QUALIDADE

Segundo Kläs *et al.* (2010), a avaliação de produtos de software pode ser dividida logicamente em dois momentos. O primeiro é a especificação dos elementos que permeiam e definem os propósitos de qualidade almejados, separando qualitativamente as características de um produto dos aspectos que influenciam na sua qualidade final. O segundo momento diz respeito à avaliação em si, onde é definido um processo capaz de guiar os avaliadores na quantificação das características de qualidade, definidas na primeira etapa, por meio de métodos e métricas específicos para tal. Seguindo essa abordagem, a proposta explorada nesta dissertação descreve, neste capítulo, um metamodelo cuja estrutura define os elementos considerados pertinente na avaliação de artefatos de software. No Capítulo 4, então, é descrito como estes elementos podem ser organizados para permitir tal avaliação.

Um metamodelo tem o objetivo de ajudar os vários interessados na adoção de uma visão comum acerca dos requisitos de qualidade pretendidos para um projeto específico, ao mesmo tempo em que permite a descrição estruturada dos elementos, conceitos e relacionamentos necessários a essa visão. A estrutura apresentada a seguir, bem como suas relações, foi inspirada nos diversos modelos de qualidade – descritos no Capítulo 2 – e visa organizar o conhecimento dos avaliadores na busca por métricas e métodos de avaliação que melhor representem os seus propósitos de qualidade.

Porém, a definição de um metamodelo deve levar em conta alguns parâmetros. De acordo com Trendowicz e Punter (TRENDOWICZ; PUNTER, 2003), um mecanismo para avaliar a qualidade de produtos de software deve ser baseado em três requisitos básicos: *flexibilidade, reusabilidade* e *transparência*.

A flexibilidade está associada à dependência do contexto de qualidade do software. Partindo-se do princípio de que cada organização, projeto e mesmo processos, estão envolvidos em ambientes dinâmicos, e que se diferenciam por suas particularidades, o modelo de avaliação deve poder ser suficientemente flexível ao ponto de adequar-se às diferentes abordagens que envolvem o ambiente organizacional de cada projeto de software. Por exemplo, se forem considerados que os paradigmas de desenvolvimento *orientado a objetos* e *estruturado* possuem características diferentes envolvendo uma série de fatores, como tamanho, complexidade, formalidade e controle (KLÄS; LAMPASONA; MÜNCH, 2011) – para citar alguns, a estrutura de avaliação deve ser capaz de contemplar essas diferenças.

Por outro lado, a flexibilidade, no contexto deste trabalho, também está associada às diferenças existentes entre os próprios artefatos que são produzidos durante as fases do processo de desenvolvimento do software. Como já citado anteriormente, os artefatos possuem particularidade entre si que são evidentes. Documentação, modelos UML, código-fonte, entre outros artefatos, possuem características próprias e, assim, o *framework* de avaliação deve permitir, aos avaliadores, identificarem essas características e definirem quais as melhores propostas de avaliação para cada uma. Ainda segundo Trendowicz e Punter (2003), à medida que o processo de desenvolvimento evolui, novos produtos são criados, novos processos são aplicados e mais artefatos podem ser medidos.

Paralelamente, a reusabilidade está associada a dois fatores importantes dentro da Engenharia de Software. O primeiro deles é a necessidade de se reter o conhecimento com experiências passadas e usá-lo em projetos futuros, o que acaba impactando diretamente no segundo fator, que é a necessidade de se desenvolver projetos no menor tempo – e custo – possível e, consequentemente, aumentar a rentabilidade desses projetos. Em outras palavras, quanto mais informações e experiências puderem ser aprendidas e retidas, menor será o esforço necessário para se avaliar artefatos semelhantes em projetos futuros. Obviamente que a reutilização de um processo de avaliação depende diretamente da similaridade entre os projetos, porém, o reuso pode ser definido tanto no que se referem aos dados de medições, quanto às características de qualidade e seus relacionamentos. Outra vantagem associada à reusabilidade do modelo diz respeito ao próprio modelo, pois, à medida que vai sendo reusado, é possível ajustá-lo, tornando-o mais preciso e eficiente.

Por fim, um modelo de avaliação deve fornecer uma análise racional e transparente de como as características e subcaracterísticas de qualidade se relacionam entre si e qual o impacto de uma sobre a outra. Por exemplo, a equipe de desenvolvimento deve perceber que a modularização de um diagrama de classes – que posteriormente servirá para definir a estrutura das tabelas de um banco de dados – permite um melhor entendimento da estrutura do software. Contudo, a normalização excessiva dessas tabelas pode, reconhecidamente, afetar o desempenho do banco de dados e do sistema como um todo.

Uma solução encontrada para contornar o problema da transparência é permitir que os próprios *stakeholders* definam, baseados em consenso, as métricas e métodos de avaliação mais relevantes e que melhor representem cada artefato e, ao mesmo tempo, usem o metamodelo para solucionar possíveis ambiguidades ou redundâncias. Dessa forma, pode-se dizer que os requisitos de *flexibilidade* e *transparência* são complementares e intimamente relacionados.

3.1 Definição do Metamodelo

Como forma de estruturar os conceitos teóricos destacados até agora, esta seção apresenta o metamodelo desenvolvido para a avaliação da qualidade de artefatos de software. O metamodelo – ilustrado na Figura 3-1 – elenca e relaciona os elementos considerados necessários para compor o *framework* de avaliação. Esses elementos são descritos nas subseções seguintes.

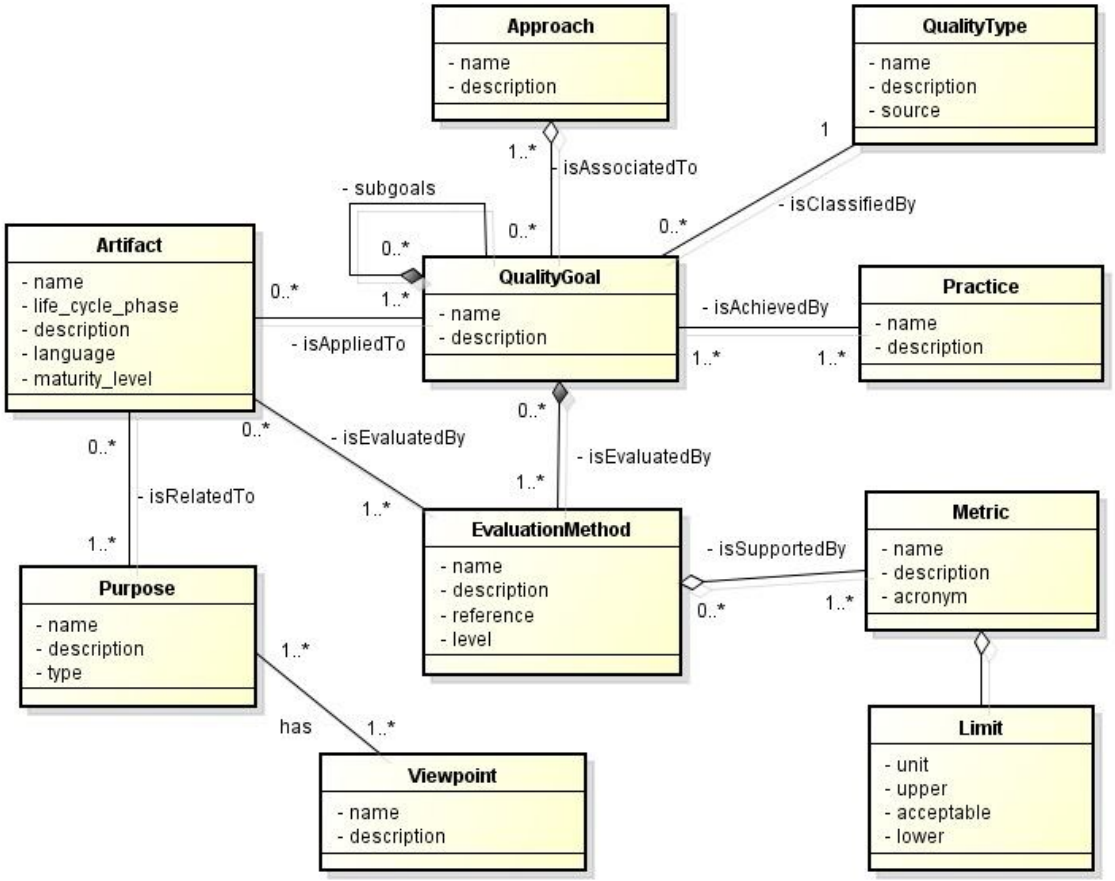

Figura 3-1 – Metamodelo de qualidade.

3.1.1 Abordagem (*Approach*)

Na maioria dos trabalhos descritos na literatura, a avaliação da qualidade está focada em um único contexto operacional. Em outras palavras, os modelos de avaliação são, geralmente, especializados e têm por objetivo avaliar a qualidade de entidades que compõem uma situação específica. Exemplos disso são modelos desenvolvidos exclusivamente para medir a usabilidade de interface de usuários (FREY et al., 2011, 2012; SEFFAH et al., 2006); a consistência de modelos UML (KROGSTIE, 2003; LANGE; CHAUDRON, 2005; USMAN et al., 2008); requisitos de projetos (THAKURTA, 2012); modelos de dados (MOODY; SHANKS, 1994; MOODY, 2003); entre outros.

Paralelamente, a avaliação da qualidade pode, também, ser expandida para abordar paradigmas ou abordagens específicas de desenvolvimento, tais como: programação orientada a objetos (OOP) (BRIAND; WÜST, 2002; EL-KORANY; NABIL; ELDIN, 2010; MENS; LANZA, 2002); *Model-Driven Engineering* (MDE) (MOHAGHEGHI; AAGEDAL, 2007; MOHAGHEGHI; DEHLEN; NEPLE, 2008; MOHAGHEGHI, 2010; SOLHEIM; NEPLE, 2006); *Software Product Lines* (SPL) (TERNITÉ, 2009; TRENDOWICZ; PUNTER, 2003); *WEB Development* (CACHERO; CALERO; POELS, 2007; DOMINGUEZ-MAYO et al., 2010); e *Service-Oriented Architecture* (SOA) (ADJOYAN; SERIAI; SHATNAWI, 2014).

Com a consciência de que o domínio do projeto exerce influência sobre os artefatos e métodos de avaliação, a metaclasse *Approach* é especificada para descrever, genericamente, as diferentes metodologias ou paradigmas que podem ser usados no desenvolvimento de softwares. Apesar das muitas propostas de modelos de qualidade existentes, conforme mencionado, ainda há a carência de uma que relacione os produtos a serem avaliados aos diversos contextos de desenvolvimento. A maioria das propostas pesquisadas apresenta ou um nível de abstração muito alto, dificultando sua aplicação na prática, ou as propostas são destinadas a contextos muito específicos, forçando a adoção, muitas vezes, de vários mecanismos de avaliação, o que pode levar a problemas de inconsistências e/ou sobreposições de métodos. Por outo lado, tem-se que diferentes abordagens de desenvolvimento possuem metas de qualidade que são inerentes e, portanto, a forma como os artefatos são gerados não pode ser ignorada durante o processo de avaliação.

Assim, primeiramente, tem-se o fato que há uma dependência direta entre a abordagem usada e os artefatos produzidos. Uma abordagem *model-driven*, por exemplo, foca

prioritariamente na transformação de modelos escritos em linguagens de modelagem análogas à UML (*Atlas Transformation Languages* – ATL; *executable UML* – xUML) e/ou linguagens para especificação de restrições em objetos (*Object Constraint Language* – OCL) (ARPAIA et al., 2011; SCHMIDT, 2006). Ou seja, nesse caso, todos os códigos-fonte são gerados automaticamente por ferramentas no final do processo. Consequentemente, a avaliação da qualidade deve priorizar os modelos ao invés do código-fonte como em um processo de desenvolvimento estruturado ou orientado a objetos. Além disso, há o fato de que embora diferentes abordagens possam compartilhar algumas metas de qualidade, o conjunto de métricas e métodos usados na sua avaliação é distinto para cada proposta. Esta visão estende-se às demais abordagens de desenvolvimento e, por isso, o metamodelo propõe esta relação entre as metas de qualidade e uma ou mais abordagem de desenvolvimento.

3.1.2 Meta de Qualidade (*QualityGoal*)

Uma Meta de Qualidade, por sua vez, descreve o foco dirigido por um mecanismo de avaliação da qualidade em produtos de software. Ou seja, é a definição, em alto nível de abstração, das necessidades a serem atendidas na visão de um determinado *stakeholder*, apresentada de forma clara e compreensível (MOHAGHEGHI; DEHLEN, 2008a).

Porém, uma vez que diferentes interessados possuem percepções ortogonais em relação à qualidade pretendida para um produto, e que essas percepções nem sempre podem ser medidas diretamente, este trabalho adota a ideia proposta por vários modelos (especialmente os modelos hierárquicos) que determinam a estruturação das metas de qualidade em sub-metas mais específicas com o objetivo relacioná-las a propósitos inerentes e, assim, facilitar sua medição.

Em um primeiro momento, essas metas podem representar os requisitos que sinalizam as várias características de qualidade descritas pelos Modelos de Qualidade, tais como as apresentadas pela ISO/IEC 9126 – funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade; pelo modelo de Dromey; Lange e Chaudron; entre outros (apresentados no Capítulo 2). Porém, a ideia de se adotar o termo *metas de qualidade* em detrimento de *características de qualidade* tem como objetivo tornar a proposta flexível e, assim, permitir que as equipes de avaliação definam as perspectivas de qualidade a serem atingidas com base nas suas próprias necessidades e/ou nas necessidades dos projetos. Dessa

forma, outras metas – além daquelas definidas por modelos publicados – podem ser adicionadas, incluindo metas abstratas, como por exemplo: *melhoria da comunicação*, *facilidade de aprendizagem*, *aumento da produtividade*, *diminuição do número de erros*, entre outras.

Ao mesmo tempo, com o objetivo de melhor estruturar a adaptação do processo de avaliação e organizar os artefatos com base em padrões organizacionais, o metamodelo permite que as metas de qualidade possam ser identificadas por meio de um *tipo* (*type*). Isso ajuda arquitetá-las em níveis de abstração diferentes, que podem ser criados pela equipe de avaliação ou baseados em algum dos modelos existentes. Como sugestão, mas não limitado a, pode-se pensar nos seguintes níveis:

- Nível *quantitativo* – representado pelos conceitos empíricos estabelecidos por padrões cujas metas podem ser definidas e medidas quantitativamente, incluindo as características internas e externas definidas pelas normas ISO/IEC 9126 (ISO/IEC, 2001) e SQuaRE (ISO/IEC, 2005); as características intermediárias e primitivas definidas por Boehm (BOEHM et al., 1978); ou as propriedades de correção, internas e descritivas estabelecidas no modelo de Dromey (DROMEY, 1995) (ver Seção 2.2);
- Nível *conceitual* – representado pelas visões de qualidade propostas pelos modelos conceituais de Lindland *et al*. (LINDLAND; SINDRE; SOLVBERG, 1994) e Krogstie *et al.* (KROGSTIE; LINDLAND; SINDRE, 1995b) que subdivide os tipos de qualidade em *física, empírica, sintática, semântica, pragmática, social* e *organizacional* (ver Seção 2.3);
- Nível *organizacional* – representado pelas expectativas da organização em relação ao software produzido, como: baixo custo de desenvolvimento, níveis de reusabilidade, completude da documentação, adequação ao cronograma, entre outros;

Dessa forma, a flexibilidade na definição das metas de qualidade pretendidas para um determinado artefato permite que seja possível tratar os vários aspectos de qualidade de uma forma uniforme tendo em vista a priorização e adequação destas metas em relação aos propósitos almejados para o produto final.

3.1.3 Artefato (*Artifact*)

Um artefato pode ser qualquer elemento criado ou transformado por uma atividade durante o ciclo de vida de um processo de software, incluindo: códigos-fonte, modelos, módulos, unidades, dados ou documentos (IEEE, 2009). Assim, no contexto deste trabalho, um artefato está basicamente associado às entregas que ocorrem durante um processo de software, não sendo restrito à equipe de desenvolvimento.

O objetivo de formalizar esses elementos tem em vista, primeiramente, determinar quais deles podem identificar as necessidades de qualidade previstas para o produto com base nas expectativas dos *stakeholders*. Posteriormente, os artefatos podem ser usados, também, para concentrar a avaliação proativa considerando os elementos que requerem atenção específica ou precisam ser refinados para que os objetivos principais do projeto sejam alcançados.

Contudo, devido à abrangência dos tipos de artefatos que se propõe avaliar, e pelo fato de evoluírem ao longo do processo de desenvolvimento, alguns atributos se fazem necessários para que a avaliação possa ser dimensionada. Entre eles, destaca-se o atributo *life_cycle_phase* que classifica o artefato em relação ao ciclo de vida do processo em que está sendo avaliado. A descrição das fases pode ser relacionada a algum modelo de processo específico, como RUP, XP, *SCRUM*, ou ser descrita de forma genérica como propõem alguns autores, entre eles Pressman (2010) que identifica como fases de um ciclo de vida: *comunicação*, *planejamento*, *modelagem*, *construção* e *entrega*. Dessa forma, essa estruturação permite que um artefato tenha suas metas de qualidade determinadas no início de cada fase do desenvolvimento e, então, ao longo do processo possa ser avaliada e comparada tendo em vista o seu estado atual e o propósito de qualidade projetado.

Outro atributo importante para a avaliação é a linguagem usada para expressar um artefato (*language*). Como já mencionado anteriormente, a teoria sobre os Modelos Conceituais (Seção 2.3) sugere que a linguagem usada para construir a instância de um determinado artefato tem forte influência na escolha das métricas e métodos de avaliação. Essa influência pode abranger tanto tipos diferentes – como códigos-fonte, modelos e documentações – quanto artefatos de um mesmo tipo. Por exemplo, um *caso de uso* pode ser descrito na forma de diagramas ou por meio de um documento textual; ao mesmo tempo, um algoritmo pode ser implementado em várias linguagens de programação (C, C#, Java, etc.) e essas diferenças podem impactar na escolha dos métodos usados para a avaliação dos artefatos.

O nível de maturidade (*maturity_level*), por sua vez, foi inspirado no trabalho de (VANDEROSE, 2012) e visa fornecer uma forma de caracterizar o estado das instâncias de cada artefato, tais como: versão, protótipo, nível de refatoramento, se o artefato está pronto ou em produção, e assim sucessivamente.

Por fim, tem-se que a identificação dos artefatos a serem avaliados é fundamental para que se possa interpretar o nível de qualidade em que o software se encontra. Todavia, existem outros aspectos que envolvem o desenvolvimento de um produto de software, tais como custo, prazos e riscos. Dessa forma, ao se definirem certas expectativas de qualidade para um determinado projeto, como usabilidade ou segurança, por exemplo, ficam mais evidentes quais os artefatos que, prioritariamente, devem concentrar os esforços de aprimoramento e, com isso, evitar o desperdício de tempo e recursos. Ao encontro dessa ideia, Elberzhager e Münch (2011) argumentam que na necessidade de se economizar esforços, por questões de tempo ou custo, é possível avaliar apenas os artefatos mais críticos. Por outro lado, se a abordagem é um sistema crítico, todos os artefatos devem ser avaliados extensivamente.

3.1.4 Ponto de Vista (*ViewPoint*)

Normalmente tem-se que as metas de qualidade tendem a ser definidas e avaliadas durante o processo de software pela equipe de desenvolvimento. Porém, Trendowicz e Punter (2003) argumentam que outros interessados, relacionados ao desenvolvimento do projeto, deveriam ser envolvidos também na avaliação como forma de aceitação e validação dessas metas.

Dessa forma, a metaclasse *ViewPoint* visa identificar, como um todo, quem são os envolvidos em um projeto de software: usuários, clientes, gerentes, desenvolvedores, entre outros. A própria organização pode ser vista como uma interessada, uma vez que existem questões ambientais e contextos de negócio que devem ser considerados quando um software é concebido.

À primeira vista pode-se pensar em uma relação direta entre os interessados e os propósitos de qualidade, tal como abordam alguns modelos. Por exemplo, a manutenibilidade – descrita como uma característica de qualidade interna pela ISO/IEC 9126 – está relacionada à equipe de desenvolvimento, ao contexto organizacional e à gerência de projetos, pois

determina a capacidade de um software ser modificado e quanto melhor for essa capacidade, menos recursos serão necessários na realização de alterações futuras. Por outro lado, esta mesma característica pode interessar, também, aos clientes, visto que um produto que preserva a facilidade de manutenção tende a ter um tempo de retorno menor entre as requisições de alterações vislumbradas depois da implantação. Esta mesma linha de raciocínio pode ser estendida às outras metas de qualidade o que torna a relação *artefatos-propósitos-interessados* multidimensional, pois os propósitos de um artefato podem ser variados e cada propósito pode ter diversos interessados.

Dessa forma, pode-se dizer que a adequação de uma meta de qualidade para suas métricas passa pela percepção dos atores envolvidos. Haja vista que um dos objetivos deste trabalho é justamente explorar o caráter multifacetado que envolve as metas de qualidade e os *stakeholders*, transformando-os em pontos de vista segundo um propósito específico, a definição de metas de qualidade para cada artefato gerado visa restringir o domínio do problema e, com isso, tornar mais clara a percepção de quais as métricas necessárias e os métodos de avaliação mais adequados para cada artefato de software.

3.1.5 Propósito (*Purpose*)

Paralelamente, a metaclasse *Purpose* tem o objetivo de identificar o propósito que descreve a intenção de um artefato dentro do ciclo de vida do processo de software, juntamente com o motivo pelo qual ele deve ser avaliado. Contudo, uma vez que intermediam a relação entre os artefatos e os seus possíveis interessados, os propósitos podem ser os mais diversos.

Como forma de ilustrar esta preposição, tem-se que durante a engenharia de requisitos, modelos UML podem ser usados para definir os atributos de qualidade e requisitos para planejar o sistema de software e, assim, constituem um método de concordância sobre qualidade entre o cliente e a organização. Por outro lado, durante a fase de implementação, eles são usados como base para guiar a equipe de desenvolvimento na codificação da aplicação, fornecendo padrões, diretrizes e recomendações diretas sobre como a implementação do sistema deve ser realizada e, assim, constituem uma abordagem construtiva para alcançar a qualidade. Ou seja, os mesmos artefatos – modelos UML – neste caso, podem ser usados tanto para documentar a comunicação entre analistas, gerentes e clientes, quanto como referência estrutural para a

equipe de desenvolvimento. Assim, tem-se que as metas de qualidade associadas a cada propósito podem ser diferentes, por exemplo: comunicabilidade para o primeiro caso e completude para o segundo (MOHAGHEGHI; DEHLEN, 2009).

Por outro lado, existem artefatos que são criados para um único propósito, tal como o Termo de Abertura de Projeto, proposto pelo PMBOK (PMI, 2013) cujo objetivo é formalizar o comprometimento e aceitação do projeto entre os clientes e a organização. Porém, independentemente da multiplicidade, definir quais são os objetivos de um artefato ajuda a equipe de avaliação a determinar como esse artefato deverá ser avaliado e quais são as metas de qualidade que melhor o caracteriza.

Deissenboeck *et al.* (2009) classificam o propósito de uma avaliação sob três objetivos básicos: *definição*, *avaliação* ou *predição* da qualidade. Seguindo essa abordagem, o metamodelo define, para a metaclasse *Purpose*, um atributo *type*, justamente para que os propósitos descritos pelos avaliadores possam, caso necessário, serem categorizados em um destes três objetivos ou outros que posteriormente se façam evidentes.

3.1.6 Método de Avaliação (*EvaluationMethod*)

De acordo com Mohagheghi e Dehlen (2008b), cada propriedade de um artefato deveria ser avaliada tanto quantitativamente quanto qualitativamente. Dessa forma, o metamodelo proposto aborda essa questão por meio da definição de uma metaclasse, cujo objetivo é estabelecer um conjunto apropriado de possíveis métodos, compatível com as metas de qualidade relacionadas, e que possa guiar a avaliação dos artefatos segundo critérios significativos baseados em métricas pré-estabelecidas e reconhecidas.

Alguns métodos de avaliação têm uma aplicação direta na relação entre as metas de qualidade e as métricas usadas. Este é o caso, por exemplo, de uma avaliação de desempenho, em que a medição está relacionada à execução de um determinado artefato (código-fonte ou componente de software, neste caso) e a sua medida pode ser determinada por meio de uma unidade de tempo. Em contrapartida, para outras metas, a definição de métodos e métricas apropriados não é tão simples, embora a teoria aponte o contrário. Algumas referências bibliográficas (ANTOLIÉ, 2010; BERTI-ÉQUILLE et al., 2011) apontam métodos como *checklists*, inspeções e testes para avaliar as mais diversas metas de qualidade, tais como:

completude, correções sintática e semântica, funcionalidade, reusabilidade, entre outros. Por fim, algumas metas de qualidade não possuem um consenso de como a avaliação deve ser realizada. Normalmente, elas são fortemente subjetivas, tais como os níveis compreensibilidade ou comunicabilidade, e a forma pela qual são avaliadas depende muito do conhecimento de especialistas ou ainda de experiências passadas, como a medição da quantidade de chamadas para o suporte técnico, o número de reclamações de clientes ou, até mesmo, a contabilização da quantidade de erros cometidos pelos usuários.

De qualquer forma, tem-se que a especificação de métodos e métricas apropriados para avaliar um determinado artefato depende, sobretudo, do conhecimento e experiência da equipe de avaliação. Além disso, outros fatores podem influenciar nesta escolha, tais como o contexto do negócio, políticas organizacionais e o uso de ferramentas apropriadas.

A definição de quais os métodos e as métricas se adequam a cada meta de qualidade e artefatos está fora do escopo deste trabalho. Contudo, o objetivo é estruturar as definições com o propósito de guiar os avaliadores nas suas escolhas. Para isso, alguns atributos, considerados inerentes aos métodos de avaliação são definidos. Entre eles tem-se o atributo *reference* cujo objetivo é especificar a origem de um determinado método. Além dos tradicionais, como inspeções, simulações e testes, existem trabalhos, descritos na literatura, focados na proposta de avaliação de artefatos em contextos específicos. Este é o caso, por exemplo, do trabalho de Elberzhager e Münch (ELBERZHAGER; MÜNCH, 2011) que propõe uma abordagem que integra inspeções e testes como parte de uma estratégia de prevenção e predição de defeitos em códigos-fonte desenvolvidos sob o paradigma de orientação a objetos. A ideia principal dessa abordagem é usar a inspeção de métricas para forçar as atividades de testes em classes específicas, classificadas segundo as suas probabilidades de erros ou riscos. Nesse trabalho, os autores associam, ainda, algumas métricas a sua abordagem, tais como quantidade de defeitos (número absoluto de defeitos por inspeção) e densidade de defeitos (número de defeitos por unidade – classe ou linha de código – inspecionadas).

Outros exemplos de métodos de avaliação específicos são abordados no trabalho de Usman *et al.* (USMAN et al., 2008) que descreve uma série de técnicas[3] usadas para validar a consistência de modelos UML em relação aos códigos-fonte desenvolvidos com base nesses

[3] Embora o trabalho original descreva as propostas como *técnicas*, elas podem ser entendidas como *métodos* e, dessa forma, são perfeitamente compatíveis com a proposta desta dissertação.

modelos. As técnicas apresentadas são organizadas e divididas em três categorias: representadas formalmente (*formally represented*), técnicas representadas pela extensão da UML (*extended UML representation techniques*) e técnicas sem representações intermediárias (*no intermediate representation*). Segundo os autores, a escolha de técnicas apropriadas para avaliar a consistência de modelos UML pode resultar em códigos-fonte mais maduros e representativos, além de melhorar a manutenibilidade desses códigos. Além disso, a relação entre o processo de software e a validação da consistência tende a ajudar as organizações financeiramente, pois minimiza custos e retrabalhos durante a fase do desenvolvimento de software.

Paralelamente, os métodos de avaliação podem ser classificados, também, segundo sua maturidade e completude. Neste sentido, Punter *et al.* (1997) argumentam que produtos com diferentes riscos de aplicação não devem ser avaliados com o mesmo rigor. Os autores citam, como um exemplo extremo e puramente ilustrativo, que processadores de texto têm uma criticidade menor quando comparados a um sistema de segurança de uma usina nuclear, portanto, os primeiros podem ser submetidos a uma avaliação menos criteriosa que os últimos. Baseado nisso, o atributo *level* pode ser usado para definir, em uma escala crescente, quatro níveis de critérios: A, B, C, D; sendo A os métodos considerados menos criteriosos. Dessa forma, métodos relacionados à avaliação da confiabilidade de um artefato, por exemplo, poderiam ser categorizados como: A – testes funcionais; B – análises de tolerância a falhas; C – modelo de crescimento da confiabilidade; e, D – prova formal. Assim, o conceito de nível de avaliação define a profundidade ou rigor da avaliação em termos de técnicas de avaliação a serem aplicadas e os resultados da avaliação a serem alcançados. (PUNTER; SOLINGEN; TRIENEKENS, 1997).

3.1.7 Métrica (*Metric*)

Entre as questões envolvendo a avaliação de produtos de software, as métricas certamente são as mais comuns e referenciadas na literatura. Vários estudos têm mostrado que métricas de software podem ajudar a melhorar o processo de desenvolvimento, compreender a complexidade, descobrir e prever falhas em unidades de software, além de ajudar a estimar e otimizar a alocação dos recursos e esforços necessários para o desenvolvimento ou manutenção dos sistemas (FENTON; PFLEEGER, 1996; MOHAGHEGHI; DEHLEN, 2009).

Assim, tem-se que métricas e metas de qualidade possuem uma relação estreita e, com base nisso, várias abordagens vêm sendo usadas para determinar e especificar essa relação, entre elas a abordagem GQM (BASILI, 1992) – descrita na Seção 2.5.1; a proposta de Lange e Chaudron (2005) – abordada na Seção 2.2.5; o trabalho de Mohagheghi e Dehlen (2009); entre outros. De uma forma geral, as métricas se dividem em dois tipos: as diretas, definidas em termos de atributos observáveis, como por exemplo, esforço, tamanho, custo e tempo; e as indiretas (ou derivadas), que podem ser obtidas por meio da combinação de outras com o objetivo de medir atributos qualitativos, tais como: complexidade, confiabilidade e facilidade de uso (DE ABREU; MOTA; ARAÚJO, 2010).

Porém, ao contrário da maioria das propostas que estabelecem uma relação direta entre as metas de qualidade observáveis e suas métricas correspondentes, este trabalho objetiva relacionar as métricas com métodos específicos de avaliação, e estes, então, com as metas de qualidade a serem alcançadas. Essa proposta se deve pelo fato de que a relação entre as métricas e as metas de qualidade é de *muitos-para-muitos*. Ou seja, uma ou mais métricas podem ser usadas para definir o nível de qualidade de um mesmo artefato, contudo, isso depende do método usado para a avaliação.

A escolha desse conjunto (métodos – métricas), e a decisão de quais métodos e métricas se relacionam entre si, é de responsabilidade do avaliador. Por isso, o objetivo é não restringi-los a um conjunto fixo, mas sim, padronizar o seu emprego de forma apropriada em relação ao artefato que se pretende medir.

Para isso, cada métrica é definida com base em uma *unidade* e nos valores-limite *máximo*, *mínimo* e um valor tido como *aceitável*. Dessa forma, por exemplo, o *desempenho* de um componente pode ser medido em unidades de tempo com seus limites máximo e aceitável definidos pelo usuário. Por outro lado, uma meta de qualidade abstrata, como: *facilidade de aprendizado*; pode ser medida pela frequência de acessos ao sistema de ajuda, tendo como unidade a quantidade de acessos/mês e, tal como o exemplo anterior, ter valores definidos. É importante destacar, porém, que em ambos os exemplos não existe a necessidade de se expressar os valores mínimos, da mesma forma que podem existir métricas cuja especificação de um ou mais valores absolutos não se faça necessária. Há ainda o caso em que os valores dependem do contexto da avaliação. Dessa forma, tendo em vista que cada métrica possui particularidades, vale ressaltar que a definição de valores e unidades, embora recomendada, é opcional e que os valores atribuídos na inserção de uma determinada métrica servirão, apenas,

como referência para os avaliadores, que poderão alterá-los no momento em que estiverem definindo o plano de qualidade, durante o processo de avaliação.

Portanto, tem-se que o uso de métricas tende a reduzir a subjetividade na avaliação e controle da qualidade do software, fornecendo uma base quantitativa para a tomada de decisões em relação à qualidade pretendida para esse software (IEEE, 2009). Todavia, o seu uso não elimina a necessidade de um julgamento humano, apenas torna visível e identificável o nível de qualidade do artefato.

3.1.8 Prática (*Practice*)

As práticas podem ser vistas como os meios para se alcançar determinadas metas de qualidade. Elas atuam como recomendações e estão diretamente relacionadas às metas internas de qualidade. Porém, ao contrário das métricas que podem ser definidas com o objetivo de especificar um determinado valor, as práticas tendem a não ter um fator avaliativo.

A proposta de um conjunto de práticas na definição do metamodelo visa tornar a abordagem, além de puramente avaliativa, um mecanismo capaz de prever e controlar a qualidade dos artefatos. Dessa forma, o conjunto de práticas é sugerido à equipe de desenvolvimento quando as métricas propostas para avaliar uma determinada meta de qualidade não atingiu o nível desejado. Ou então, as mesmas práticas podem ser sugeridas para manter a qualidade alcançada.

O conjunto de práticas sugeridas para as metas de qualidade é bastante difundido entre os trabalhos pesquisados, contudo suas sugestões são esparsas e dependentes da abordagem usada. Todavia, os trabalhos de Lindland *et al.* (KROGSTIE; LINDLAND; SINDRE, 1995a; LINDLAND, 1993), Krogstie *et al*. (KROGSTIE; LINDLAND; SINDRE, 1995b; KROGSTIE, 2003) e Moody *et al.* (MOODY, 2003; MOODY et al., 2003) descrevem conjuntos de práticas que podem ser usados em modelos UML. Por exemplo, para melhoria da comunicabilidade pode-se usar ontologias e/ou o envolvimento de especialista. Ao mesmo tempo, outras práticas, como modularização, podem servir tanto para se alcançar metas como manutenibilidade quanto reusabilidade. Além disso, embora tenha seu escopo definido para processos de software, o *Project Management Body of Knowledge (PMBOK Guide)* (PMI, 2013) pode ser uma fonte

possível de práticas a serem usadas como forma de auxiliar na manutenção e controle da qualidade de artefatos de software.

3.2 Ilustração de Uso

Conforme descrito até agora, a especificação de um metamodelo justifica-se pela necessidade de se definir, de forma clara e precisa, quais são os elementos que devem fazer parte de um *framework* de avaliação, por meio de uma representação sistemática dos conceitos envolvidos. Dessa forma, é possível integrar as instâncias relacionadas ao processo de avaliação além de explicar sua interpretação.

A proposta defendida neste trabalho visa representar, primeiramente, as características de qualidade em alto nível, abordando as necessidades identificadas pelos *stakeholders*. Em seguida, essas características podem ser decompostas em níveis mais pontuais com o objetivo de serem medidas quantitativamente por meio de métricas apropriadas. Atrelado a isso, busca-se associar um conjunto de métodos de avaliação e as práticas sugeridas para facilitar o alcance das características.

Seguindo esses preceitos, são apresentados, nas Figuras 3-2 e 3-3, dois exemplos ilustrativos do uso do metamodelo proposto. Na Figura 3-2 ilustra-se a instanciação de objetos para a avaliação de modelos de Casos de Uso com o propósito de melhorar a comunicação e a concordância sobre os requisitos de um projeto de software desenvolvido sob o paradigma *model-driven*. A meta de qualidade principal é a usabilidade, baseada na qualidade interna proposta pela ISO/IEC 9126. A partir dela são exploradas as submetas compreensibilidade, capacidade de aprendizado e atratividade. Essas metas e submetas podem ser avaliadas por métodos específicos e quantificadas por métricas relacionadas.

Paralelamente, na Figura 3-3, estrutura-se a avaliação de códigos-fontes escritos em uma linguagem orientada a objetos tendo manutenibilidade e confiabilidade como metas de qualidade. Ambas são decompostas em sub-metas mais específicas, sendo modificabilidade e estabilidade para a primeira e maturidade para a segunda. Da mesma forma que no primeiro exemplo, o diagrama ilustra instâncias de objetos para os outros componentes do metamodelo proposto.

Contudo, vale lembrar que a escolha dos atributos pode não refletir a realidade para todos os casos e ambos os exemplos têm apenas o propósito de ilustrar como o metamodelo pode ser instanciado. Escolhas de quais propósitos, métodos de avaliação, métricas e práticas são mais adequadas a cada situação é uma prerrogativa dos analistas de qualidade. Por fim, ressalta-se que o atributo *description* não foi informado nos exemplos por motivo de legibilidade da figura e do modelo, contudo, embora opcional, esse atributo é importante e serve para apresentar aos usuários uma pequena descrição sobre o objeto, o que ele faz ou como deve ser usado. Ainda como forma de exemplificação, o Anexo A apresenta uma sequência de dados representativos para cada entidade do metamodelo. Paralelamente, no Anexo B é apresentado o modelo de dados usado na implementação da ferramenta desenvolvida neste trabalho.

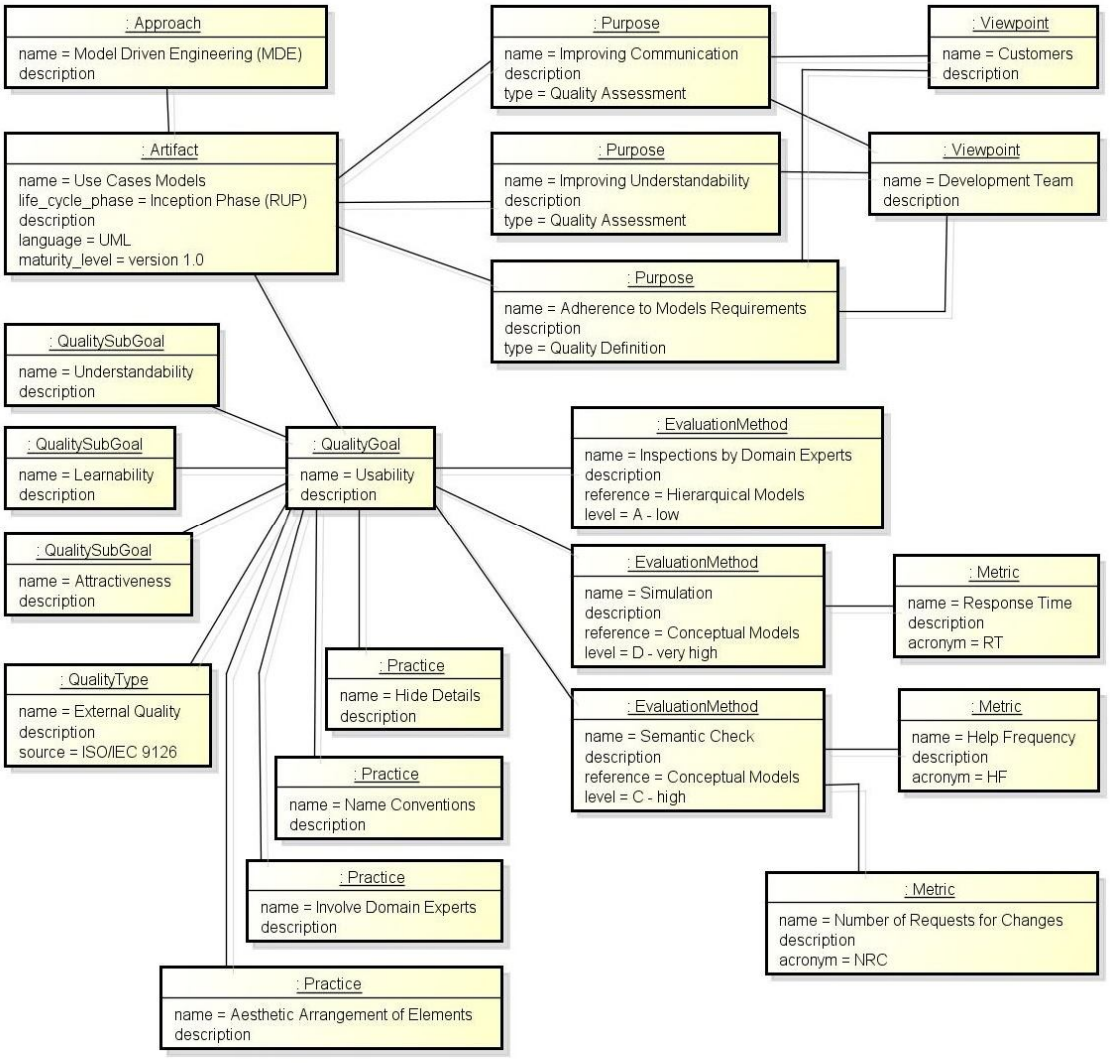

Figura 3-2 – Instâncias de exemplo do uso do metamodelo de qualidade para uma abordagem *model-driven*.

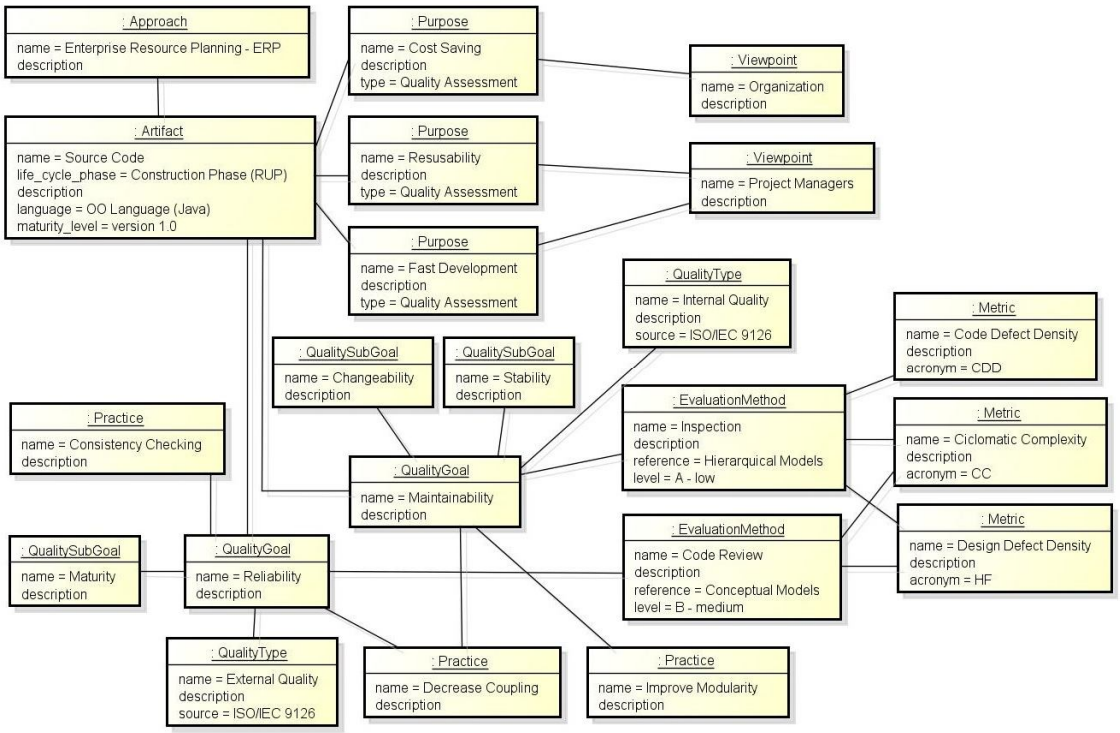

Figura 3-3 – Instâncias de exemplo do uso do metamodelos de qualidade para artefatos desenvolvidos sob o paradigma orientado a objetos.

4 PROCESSO DE AVALIAÇÃO

Atualmente, existe um variado conjunto de modelos e processos usados para avaliar produtos de software, apoiados por mecanismos que lidam com uma infinidade de questões relacionadas à qualidade. Porém, ainda há espaço para novos estudos, visto que existem deficiências quando se tenta descrever uma abordagem integrada que habilite os usuários para tratar todas essas questões de forma igualitária, mas com vistas a aspectos isolados de uma maneira uniforme (WAGNER; DEISSENBOECK, 2007).

Paralelamente, a avaliação de produtos de software deve ser objetiva – baseada em observações, não em opiniões (PUNTER; SOLINGEN; TRIENEKENS, 1997). Em outras palavras, a avaliação de um mesmo produto, com base nas mesmas especificações, deveria levar diferentes avaliadores a produzirem resultados que possam ser aceitos como idênticos e reproduzíveis.

Baseado nessas percepções, este capítulo descreve, como parte do *framework* proposto, um processo de avaliação para artefatos de software. O processo é baseado no padrão ISO/IEC 14598 (ISO/IEC, 1999), mas possui particularidades ajustadas às ideias deste trabalho, sendo o seu principal objetivo organizar os dados, estruturados pelo metamodelo descrito no Capítulo 3, de forma que os avaliadores tenham uma referência sólida e prática no momento de validar os artefatos produzidos durante o projeto com base nas metas de qualidade que julgam serem as mais importantes para o produto final.

Dentre os principais conceitos que envolvem a definição do processo, tem-se a necessidade da sua adaptação poder ser executada tanto em relação ao nível organizacional quanto ao individual de um projeto e, em seguida, armazenar essas informações com a expectativa de facilitar o reuso e, consequentemente, reduzir os esforços em projetos subsequentes. Ainda, pretende-se que o processo seja capaz de ajustar a avaliação para cada *stakeholder*, contudo, sem perder de vista o contexto da aplicação e as metas de qualidade prioritárias para que diferentes pretensões de qualidade possam ser confrontadas e, caso sejam contraditórias ou incompatíveis, possam ser então priorizadas no projeto. Por fim, o processo de avaliação objetiva fornecer um exame sistemático por meio de passos a serem seguidos por quem for avaliar a qualidade dos artefatos, separando os conceitos de qualidade dos métodos de avaliação com vistas, entre outros aspectos, aos objetivos do projeto, sua criticidade e as métricas disponíveis para tal avaliação.

Com base nessas ideias, este capítulo descreve o processo de avaliação que compõe parte do *framework* de qualidade proposto neste trabalho. Esse processo foi construído com base na norma ISO/IEC 14598, referenciada na Seção 2.5.3, além de aspectos particulares inerentes à a avaliação de artefatos de software. Ao mesmo tempo, fragmentos de telas da ferramenta *Quality Assessment Tool*, desenvolvida como parte da validação da abordagem, são mostrados como forma de ilustrar a implementação do *framework* de avaliação de artefatos de software.

4.1 Processo de Avaliação de Artefatos de Software

Depois de estabelecidos os elementos necessários para compor a avaliação dos artefatos produzidos durante o desenvolvimento de software, o próximo passo desta pesquisa é estruturar a sequência de ações a ser seguida para que os avaliadores definam um plano de qualidade que identifique métodos e métricas apropriados para avaliar e controlar a qualidade de tais artefatos.

O processo de avaliação é apresentado na Figura 4-1 por meio da notação *Business Process Modeling Notation* (BPMN). A escolha da ISO/IEC 14589 como referência deu-se pela sua compatibilidade tanto com os conceitos estabelecidos pelos modelos de qualidade quanto por aqueles defendidos ao longo do texto. Além disso, essa norma é parte complementar da ISO/IEC 9126, sendo ambas incorporadas à ISO/IEC 25000 (SQuaRE) o que a torna um modelo consistente e amplamente referenciado na comunidade acadêmica. Porém, alguns subprocessos foram ajustados de forma que pudessem refletir aspectos específicos da avaliação de artefatos de software.

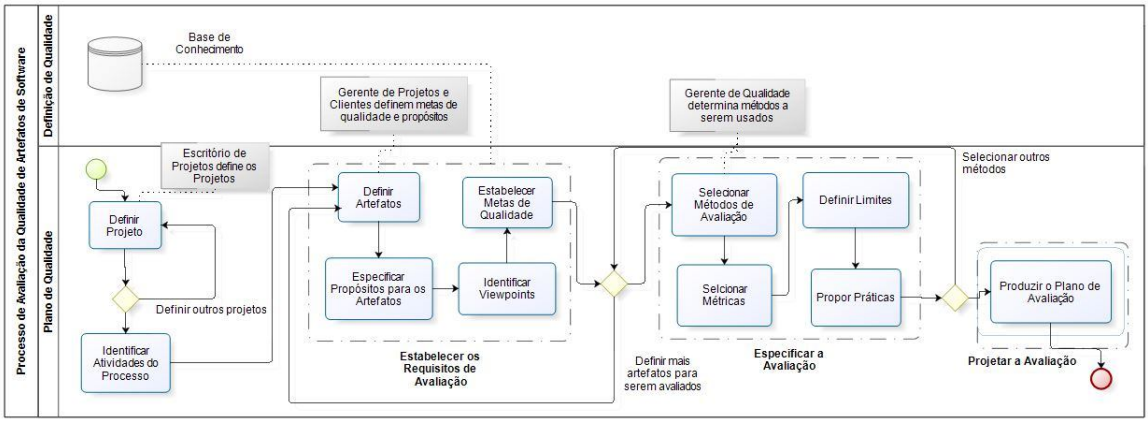

Figura 4-1 – Processo de avaliação de artefatos de software proposto neste trabalho.

Contudo, como primeiro passo para determinar o plano de qualidade, existe a necessidade de se povoar a base de conhecimento. Os dados podem ser provenientes das mais variadas fontes, incluindo os modelos descritos no Capítulo 2, trabalhos específicos ou mesmo da experiência de especialistas ou projetos anteriores desenvolvidos pela organização.

Como forma de fornecer a flexibilidade almejada pela proposta desta dissertação na composição da base de conhecimento, a ferramenta desenvolvida estabelece uma série de formulários correlacionados nos quais os avaliadores, e demais interessados, podem inserir e manter os dados referentes aos elementos que compreendem o processo de avaliação. Na Figura 4-2 é ilustrada a tela inicial da ferramenta, tendo à esquerda o menu de opções para a navegação entre esses elementos.

O menu, por sua vez, é dividido em três partes: a primeira delas refere-se à definição de elementos de qualidade (*Quality Definitions*) por onde os usuários terão acesso à inserção e manutenção dos dados, composto pelos seguintes itens: *Approaches, Artifacts, Metrics, Evaluation Methods, Quality Goals, Quality SubGoals, Quality Types, Purposes, ViewPoints* e *Practices*. Cada item leva a uma listagem dos itens já cadastrados e à possibilidade de manutenção de cada um deles ou inserção de novos.

Figura 4-2 – Tela inicial da ferramenta.

Como exemplo, nas Figuras 4-3 e 4-4 são apresentados, respectivamente, a tela de listagem e o formulário de manutenção para *metas de qualidade*, na qual são cadastradas propriedades como nome, descrição, tipo de qualidade e fonte proveniente, conforme descrito na seção 3.1.2. Contudo, a sequência lógica é semelhante para os demais itens que compõem a manutenção dos elementos da base de conhecimento. O Anexo A traz, ainda, uma exemplificação de possíveis dados que podem representar cada um dos itens descritos pelo metamodelo de qualidade e implementados pela ferramenta.

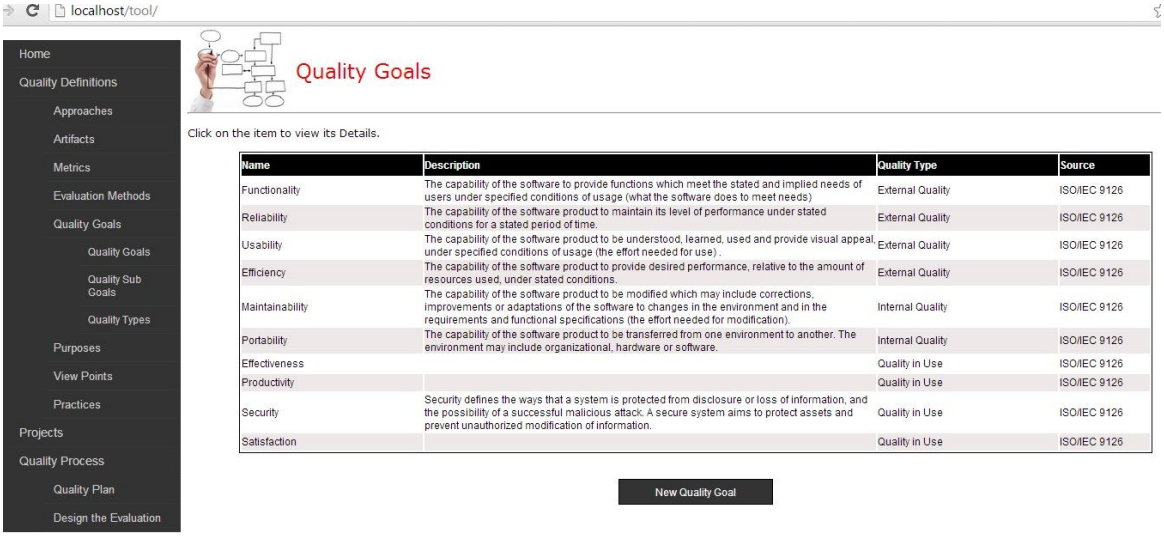

Figura 4-3 – Tela exemplificando a listagem de dados dos elementos do metamodelo.

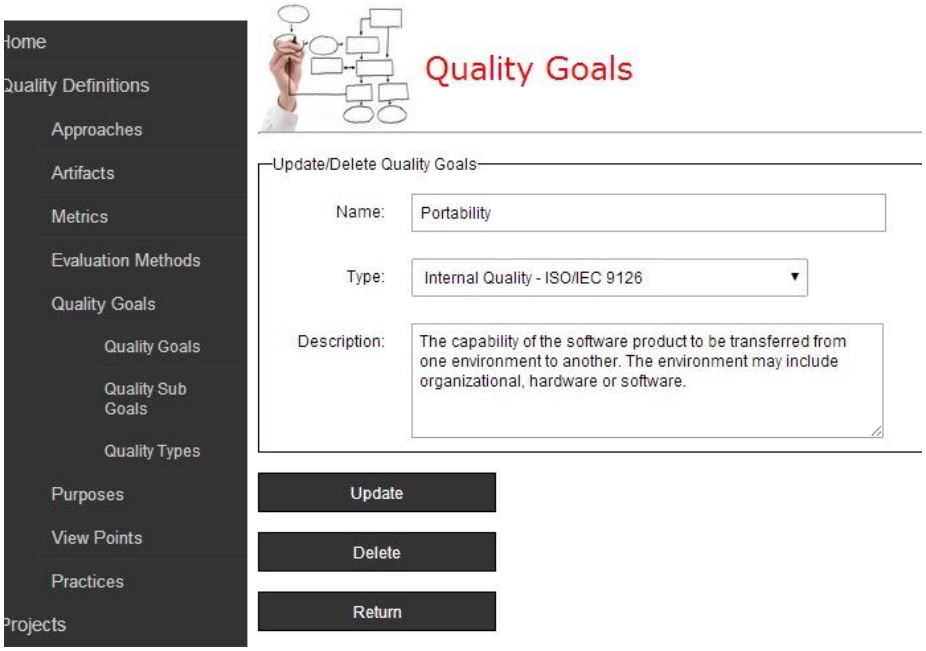

Figura 4-4 – Tela exemplificando a manutenção dos dados dos elementos do metamodelo.

O próximo passo é definir as informações referentes ao projeto a ser avaliado, tais como nome, descrição, abordagem ou paradigma de desenvolvimento, além do modelo de processo de software usado. Essas informações serão, mais tarde, relacionadas às fases do ciclo de vida e aos artefatos que compõem cada uma das fases. Na Figura 4-5 é apresentado o formulário de manutenção de projetos.

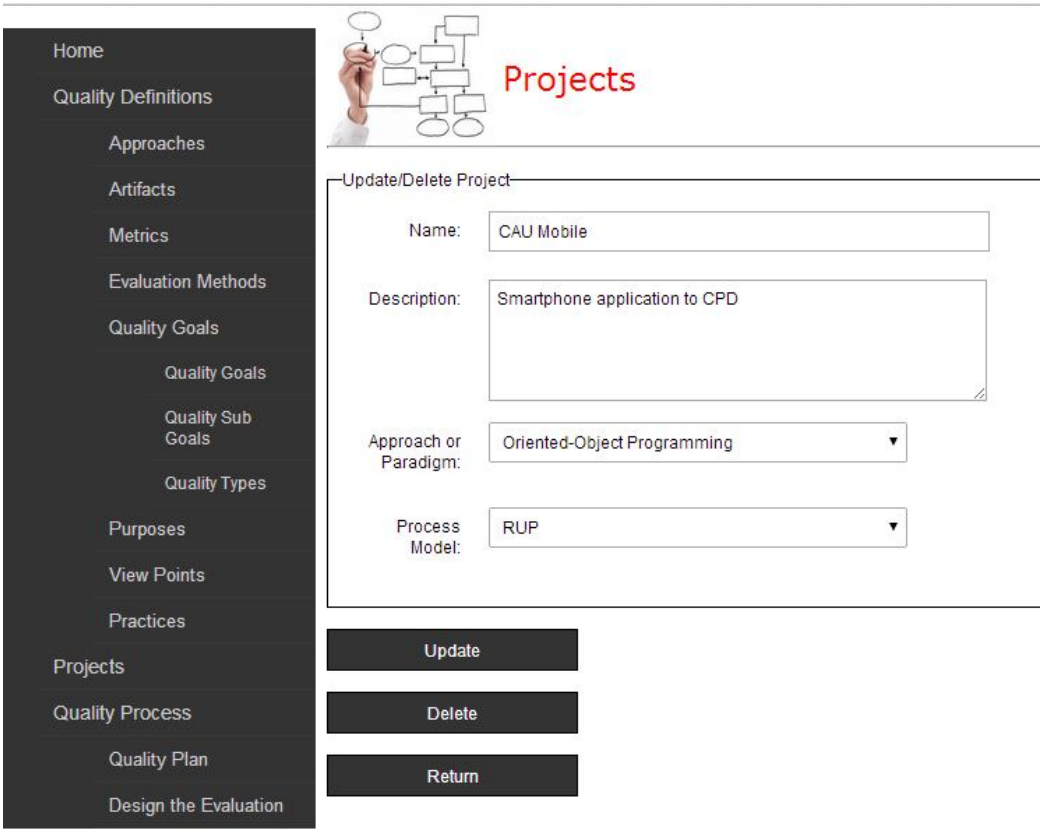

Figura 4-5 – Formulário para manutenção de projetos.

Por fim, a terceira etapa corresponde ao processo de avaliação em si. Esta etapa inicia-se pela seleção do projeto (cadastrado na etapa anterior) e a definição dos ciclos de vida que compõem o modelo de processo usado – Figura 4-6. Evidentemente, deve haver uma relação direta entre o modelo de processo informado no cadastro do projeto e a escolhas das fases correspondentes, porém, a ferramenta se encarrega de verificar esta restrição.

A partir de cada fase do projeto são estabelecidos os *requisitos de avaliação* para a definição do plano de avaliação da qualidade. Ou seja, para cada fase do ciclo de vida do processo, são selecionados os artefatos gerados e, para cada artefato, são definidos seus propósitos, interessados e metas de qualidade correspondentes. Dessa forma, busca-se estabelecer uma segmentação de granularidade fina para que o foco da avaliação esteja

relacionado diretamente ao artefato, porém, sem perder de vista o contexto que o envolve. A relação entre os propósitos, interessados e as metas de qualidade correspondentes a cada artefato é definida pelos avaliadores que ficam livres para estabelecer uma relação múltipla entre estes elementos. Na Figura 4-7 é ilustrada esta etapa do processo de avaliação.

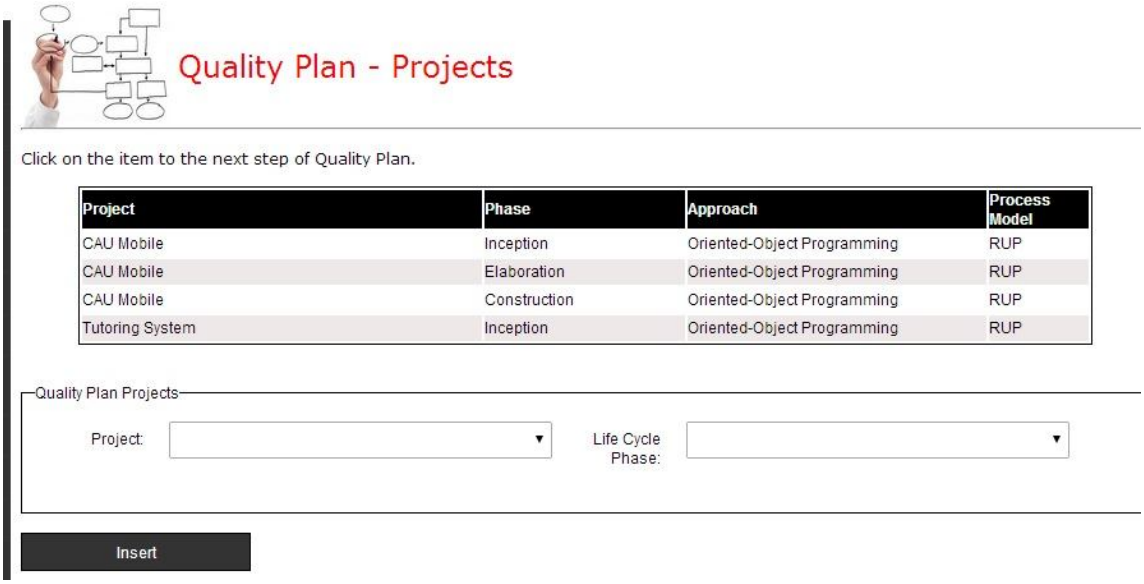

Figura 4-6 – Seleção do projeto e definição das fases do ciclo de vida desse projeto.

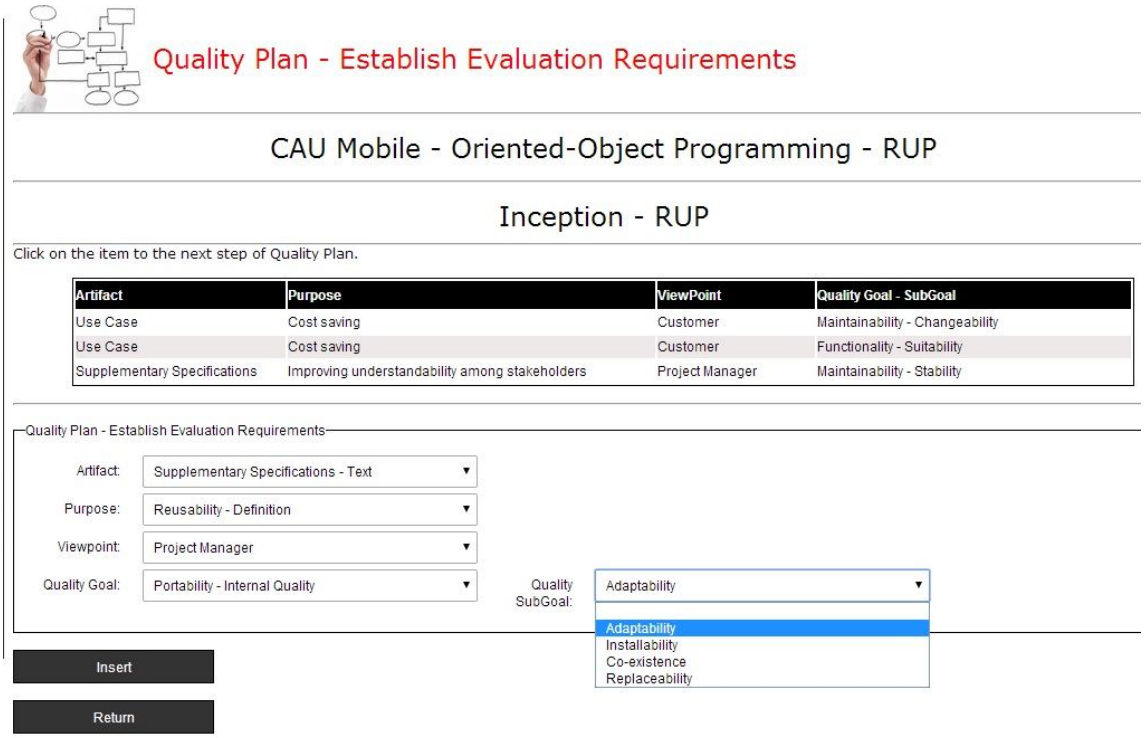

Figura 4-7 – Definição dos Requisitos de Avaliação para cada artefato de software.

Em seguida, tem-se a especificação da avaliação, que no contexto desta pesquisa envolve relacionar métodos de avaliação, métricas e práticas para cada artefato selecionado. A definição dos métodos tem como ponto de partida o nível de profundidade que se deseja especificar para cada artefato, tal como definido na Seção 3.1.6, e com base nisso, o avaliador pode selecionar um ou mais métodos compatíveis. Por exemplo, ao indicar que o rigor da avaliação será alto (*C – high*), a ferramenta disponibilizará para seleção somente os métodos relacionados a esse nível.

Nesta mesma etapa, são definidas, também, as métricas e possíveis valores limites. Vale lembrar que, conforme mencionado na Seção 3.1.7, a definição de valores limites para as métricas não é obrigatório, além disso, os limites definidos nos cadastros das métricas são usados como referência aqui, na especificação da avaliação, mas o avaliador pode alterar esses valores para cada especificação de avaliação. Em outras palavras, os avaliadores têm a liberdade de determinar os métodos de avaliação ao mesmo tempo em que definem limites apropriados para os níveis de qualidade pretendidos. Esta ideia é proposta para que os requisitos de flexibilidade e transparência sejam mantidos durante todo o processo avaliativo. Na Figura 4-8 é ilustrada a tela da ferramenta que compõe esta etapa da avaliação.

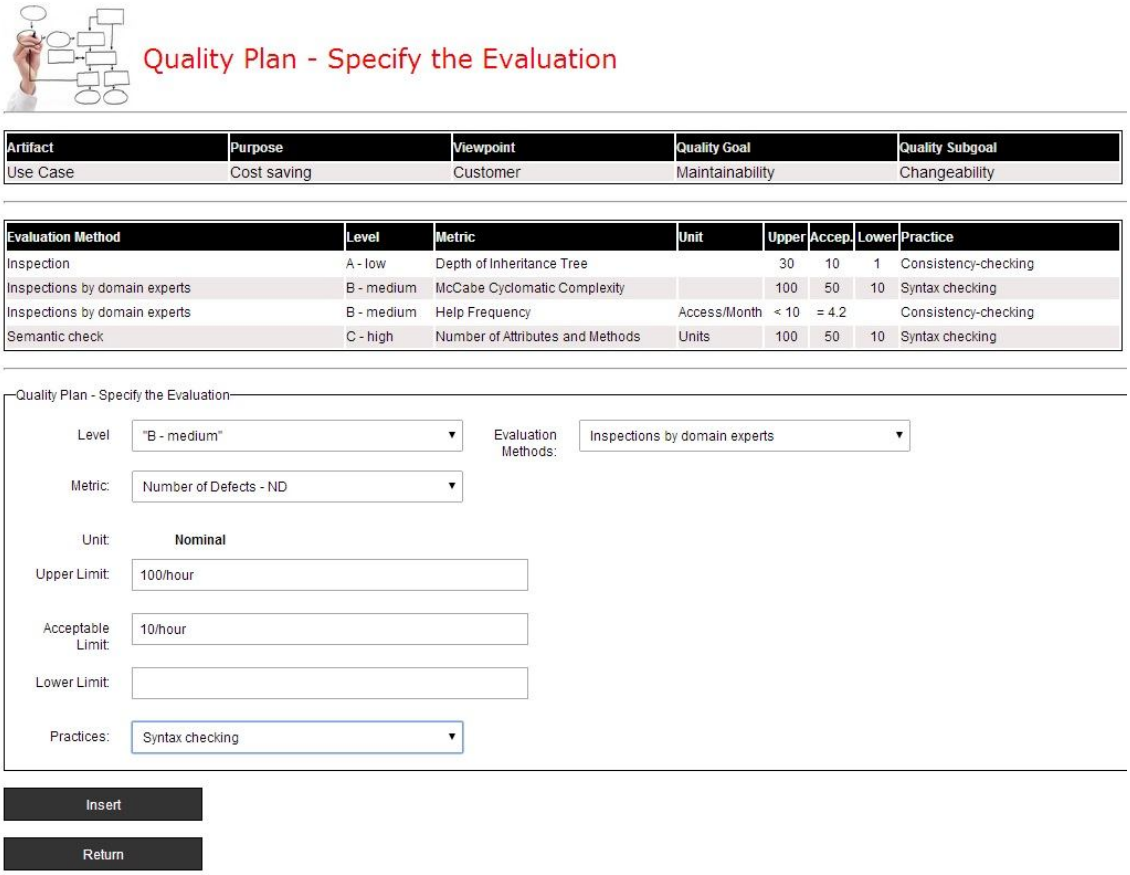

Figura 4-8 – Definição de métodos, métricas e práticas para especificar a avaliação dos artefatos.

A próxima etapa compreende a documentação dos procedimentos definidos e que serão usados pelo avaliador para definir a qualidade dos artefatos selecionados, ou seja, nesta fase é produzido o plano de qualidade. A ideia é que todas as definições dos artefatos e suas relações sejam estruturadas de forma clara e compreensível com o objetivo de, primeiramente, guiar os avaliadores na validação da qualidade. Contudo, o plano de qualidade pode ser, também, definido antes da construção do software e, então, fornecido à equipe de desenvolvimento para que as especificações de qualidade sejam incorporadas ao desenvolvimento. Por fim, o plano de qualidade ficará armazenado e poderá servir como referência para avaliações futuras, visto que os artefatos gerados durante um processo de software são semelhantes se mantidas as mesmas abordagens de desenvolvimento e os mesmos modelos de processo de software. Na Figura 4-9 é ilustrado um exemplo de um plano de avaliação da qualidade. Para facilitar a pesquisa e geração, a ferramenta segmenta o relatório dos planos por projeto e fase do ciclo de vida.

Design the Evaluation

Tutoring System - Oriented-Object Programming - RUP

Inception - RUP

Artifact			Language				Maturity
Data Model			UML				Version 1
Purpose - Type		**Viewpoint**	**Quality Goal - Subgoal**				
Comprehension - Prediction		Database Designer	Maintainability - Changeability				
Evaluation Method	**Level**	**Metric**	**Unit**	**Upper**	**Accep.**	**Lower**	**Practice**
Consistency analysis tool	B	Number of Attributes and Methods	Units	100	50	10	Statement deletion
Evaluation Method	**Level**	**Metric**	**Unit**	**Upper**	**Accep.**	**Lower**	**Practice**
Peer Review	B	Number of Associations	Units	20	10		Syntax checking
Artifact			**Language**				**Maturity**
Class Diagram			UML				Version 1
Purpose - Type		**Viewpoint**	**Quality Goal - Subgoal**				
Improve Communication - Assessment		Customer	Maintainability - Analyzability				
Evaluation Method	**Level**	**Metric**	**Unit**	**Upper**	**Accep.**	**Lower**	**Practice**
Inspection	A	Number of Associations	Units	50	25		Statement deletion
Evaluation Method	**Level**	**Metric**	**Unit**	**Upper**	**Accep.**	**Lower**	**Practice**
Consistency analysis tool	B	Number of Aggregations	Units	10	5		Statement deletion
Purpose - Type		**Viewpoint**	**Quality Goal - Subgoal**				
Implementation - Assessment		Developer	Maintainability - Changeability				
Evaluation Method	**Level**	**Metric**	**Unit**	**Upper**	**Accep.**	**Lower**	**Practice**
Consistency analysis tool	B	Number of Associations	Units	50	10		Consistency-checking

Return

Figura 4-9 – Exemplo da listagem de um plano de avaliação.

A última etapa compreende a execução da avaliação, quando os artefatos especificados no plano de avaliação são submetidos aos métodos de avaliação tendo, então, seus resultados comparados aos limites estabelecidos por cada métrica. Todavia, entende-se que essa é uma etapa mecânica, podendo ser executada tanto por pessoas quanto por aplicações especializadas e constitui um passo seguinte a esta pesquisa, portanto, fora do seu escopo.

4.2 Considerações

Este capítulo apresentou o processo de avaliação de artefatos de software que compõe o *framework* de qualidade desenvolvido neste trabalho. A ideia desse *framework* segue preceitos baseados em outros processos descritos na literatura e abordados na Seção 2.5. Todavia, todos os processos pesquisados seguem uma sequência lógica semelhante: i – a definição do que se quer avaliar; ii – a seleção de metas (ou características) de qualidade relacionadas ao objeto em avaliação; iii – a definição de métricas compatíveis com tais metas de qualidade.

A principal diferença entre os processos referenciados e o proposto é que este inclui uma etapa a mais: a relação entre as metas de qualidade e as respectivas métricas passa pela definição de métodos relacionados à avaliação. Ao mesmo tempo, o processo permite que seja definido um plano de qualidade para cada artefato com base na fase do ciclo de vida do processo em que se encontra, considerando uma série de aspectos, como: a abordagem de desenvolvimento usada no projeto, o propósito de cada artefato, a visão dos interessados e o nível de rigor da avaliação em que tal artefato deve ser submetido.

Por fim, entende-se que o armazenamento de dados referentes aos planos de qualidade definidos durante o processo de avaliação para um projeto específico possa servir como referência para guiar a avaliação em projetos futuros, independentemente da similaridade entre os projetos, visto que normalmente os artefatos produzidos em uma mesma etapa do ciclo de vida de um processo de software seguem os mesmos propósitos.

5 ESTUDO DE CASO

Como forma de validar a proposta defendida neste trabalho, foi realizado um estudo de caso envolvendo a avaliação da qualidade em três projetos de software, desenvolvidos por alunos de graduação do curso de Sistemas de Informação da Universidade Federal de Santa Maria (UFSM). Embora tenham tido propósitos acadêmicos, todos os projetos foram implementados e serão implantados em clientes reais. Ou seja, os projetos seguiram um desenvolvimento linear, iniciado pela negociação com o cliente (definição do Termo de Abertura do Projeto), projeto, codificação e implantação. Até o momento da escrita deste trabalho, os projetos se encontram em fase de testes nos clientes.

Assim, a validação foi realizada em duas etapas. A primeira delas envolveu os alunos de graduação que desenvolveram os projetos onde o objetivo foi verificar as metas de qualidade pretendidas para as aplicações projetadas e como os alunos/desenvolvedores pretendiam alcança-las. A segunda etapa, por sua vez, contou com a participação de alunos de pós-graduação em Informática da mesma universidade. Nesta segunda fase, o propósito foi confrontar os dados obtidos na primeira, sob o ponto de vista de desenvolvedores e analistas mais experientes, tanto na área de desenvolvimento quanto na área de avaliação de produtos de software.

Dessa forma, ao logo deste capítulo, primeiramente são descritos, resumidamente, cada um dos três projetos usados na validação, seus propósitos, artefatos gerados e perfis tecnológicos. Em seguida, é abordada a metodologia usada no experimento, como a avaliação da qualidade se deu para cada projeto e, por fim, uma análise e descrição dos resultados obtidos.

5.1 Projeto *CAU Mobile*

O objetivo do projeto *CAU Mobile* foi projetar e desenvolver um aplicativo móvel para *smartphones* que, conectado ao sistema do Centro de Processamento de Dados (CPD) da UFSM, auxiliasse os técnicos administrativos no atendimento às chamadas de suporte referentes à manutenção dos computadores da instituição. Atualmente esses servidores precisam se conectar à *intranet*, depois de cada atendimento, para preencher seus relatórios e

consultar os próximos atendimentos a serem feitos. Uma vez implantado, o aplicativo visa tornar esse processo mais ágil e eficiente, pois os técnicos não precisarão mais buscar um ponto de acesso ou mesmo retornar ao CPD para visualizar as próximas tarefas.

Do ponto de vista técnico, a aplicação foi desenvolvida exclusivamente para o sistema operacional *Android* e foram usadas como linguagens de programação: HTML5 para desenvolver as interfaces com o usuário e *Python* para realizar as operações envolvendo o banco de dados e o gerenciamento do sistema. Por fim, criou-se um pacote de instalação do aplicativo por meio do *framework Phonegap Build*.

Durante o desenvolvimento do projeto, além dos códigos-fonte, foram gerados os seguintes artefatos: Termo de Abertura do Projeto (documento), diagramas de casos de uso, diagrama de classes, diagramas de sequência, diagramas de pacotes, diagrama de componentes e o modelo de dados.

5.2 Sistema Gerenciador de Atividades de Tutoria

O Sistema Gerenciador de Atividades de Tutoria é um projeto que foi desenvolvido em parceria com o Instituto Federal Farroupilha com o objetivo de organizar a alocação de tutores presenciais e à distância que prestam serviços à instituição. Além disso, a aplicação é usada, pelos próprios tutores, para gerenciar as turmas pelas quais são responsáveis, e pela instituição para monitorar as atividades realizadas por esses tutores, tais como a frequência das suas atividades e o contato com os alunos.

No nível tecnológico, a aplicação foi desenvolvida para o ambiente *web* por meio da linguagem PHP e JavaScript com suporte do banco de dados relacional MySQL, e os artefatos gerados foram: modelos UML – diagramas de casos de uso, diagramas de classes, diagrama de componentes e modelo de dados, além da documentação necessária para a abertura do projeto e concordância dos requisitos entre o cliente e a equipe de desenvolvimento.

5.3 Sistema de Gerenciamento de Bolsistas

Por fim, o terceiro projeto também teve como cliente o Instituto Federal Farroupilha. Porém, o objetivo dessa aplicação foi informatizar o sistema de gerenciamento de pagamento de professores bolsistas, distribuídos entre os oito campi e a reitoria da instituição. A justificativa para o desenvolvimento do projeto, segundo os autores, está no fato de que a distância física entre os polos causa lentidão e dificulta o processo, além de inviabilizar a geração de relatórios e outras consultas relevantes aos administradores do sistema. Por isso, entre os requisitos especificados para a aplicação estão: i – a construção de um sistema que forneça acesso a múltiplos usuários; ii – a gerência de dados confidenciais; iii – a especificação de uma interface intuitiva e fácil, para que pessoas com variados níveis de conhecimento em informática possam utiliza-la sem recorrer ao suporte; e iv – que o sistema desenvolvido seja modulado e documentado em um nível satisfatório para que futuras alterações e/ou manutenções possam ser realizadas facilmente por outras equipes.

Com relação às tecnologias, foram usadas as linguagens HTML para desenvolver as interfaces com os usuários; PHP para construir a lógica do sistema; e MySQL como gerenciador de banco de dados. Os artefatos gerados, a exemplo dos projetos anteriores, foram: documentação (Termo de Abertura do Projeto); modelos UML – diagramas de casos de uso, classes, componentes e modelo de dados, além dos códigos-fonte.

5.4 Primeira Fase da Avaliação

A primeira fase do estudo de caso, como mencionado anteriormente, foi realizada com os próprios alunos que desenvolveram os projetos. Primeiramente foi oferecido um treinamento abordando os conceitos de qualidade, modelos de qualidade existentes, a importância da avaliação da qualidade de produtos de software e dos seus artefatos, bem como foi explicado o objetivo deste trabalho.

Em seguida, de posse de um material de apoio fornecido, cada grupo teve como objetivo apontar, em sua opinião, quais as metas de qualidade mais importantes para o projeto como um

todo, e para cada um dos artefatos gerados durante o projeto, bem como os papéis interessados nas metas de qualidade e quais os meios conhecidos para avaliar as metas definidas.

Para o projeto *CAU Mobile*, a equipe apontou, em ordem de prioridade, as seguintes metas de qualidade e interessados: usabilidade (usuários finais); funcionalidade (cliente); eficiência (cliente); portabilidade (usuários finais); manutenibilidade (desenvolvedores); e confiabilidade (cliente e desenvolvedores). Embora tenha sido questionada, a equipe não soube justificar quais os motivos que levaram a escolha de tais metas nem quais os métodos de avaliação ou métricas que poderiam ser usados.

No projeto de tutorias, por sua vez, a equipe apontou como metas de qualidade a serem alcançadas: manutenibilidade, usabilidade e portabilidade, conforme ilustra a Tabela 5.1.

Tabela 5.1 – Metas de qualidade apontadas para o projeto Sistema Gerenciador de Atividades de Tutoria.

Metas de Qualidade	Interessados	Motivo
Manutenibilidade	Equipe de desenvolvedores e administradores	O sistema será mantido por uma equipe diferente da equipe de desenvolvimento inicial.
Usabilidade	Usuários finais do sistema	Os usuários finais do sistema serão tutores de um curso à distância que, na sua maioria, não possuem conhecimento específico em informática.
Portabilidade	Administradores do sistema	O cliente não definiu em que plataforma o sistema irá executar, portanto, é necessário que o sistema execute nas principais plataformas existentes no mercado.

Por fim, a equipe do projeto de gerenciamento de bolsistas apontou como metas de qualidade: funcionalidade, eficiência e manutenibilidade, apresentadas na Tabela 5.2.

Embora tenha sido oferecido um treinamento, a primeira fase do estudo de caso mostrou que a maioria dos desenvolvedores possui pouca noção dos conceitos de qualidade, tão pouco quais são os meios necessários para que a qualidade pretendida seja alcançada, visto que nenhum dos grupos soube informar quais métricas ou métodos de avaliação poderiam ser usados. Além disso, um dos grupos (do Sistema de Gerenciamento de Bolsistas) apontou metas de qualidade consideradas compatíveis aos propósitos da aplicação, contudo, a justificativa para as escolhas não corresponde aos propósitos de qualidade de cada meta. Essa primeira conclusão

vai ao encontro do que argumentam alguns autores (ARRUDA, 2014; DOMINGUEZ-MAYO et al., 2010; ELBERZHAGER; MÜNCH, 2011) ao afirmarem que a avaliação da qualidade em produtos de software é, normalmente, realizada sem uma base teórica sólida e sem o auxílio de uma metodologia que guie os avaliadores. Por outro lado, tem-se que os pesquisados nesta primeira etapa são alunos com pouca experiência prática no desenvolvimento de softwares. Assim, como forma de confrontar os resultados obtidos, uma segunda fase de avaliação foi proposta e descrita a seguir.

Tabela 5.2 – Metas de qualidade apontadas para o projeto Sistema de Gerenciamento de Bolsistas.

Metas de Qualidade	Interessados	Motivo
Funcionalidade	Usuários finais	O software deve ser seguro, por ter como uma de suas funcionalidades a aprovação de pagamento das bolsas do Instituto.
Eficiência	Usuários finais	Espera-se que o software mantenha o comportamento esperado.
Manutenibilidade	Equipe de desenvolvimento	Por ser um software grande, é necessário que seja de fácil compreensão e análise, para que possíveis problemas possam ser corrigidos facilmente.

5.5 Segunda Fase da Avaliação

A segunda fase do estudo de caso contou com o apoio de alunos do curso de pós-graduação em Informática da UFSM. A escolha desse grupo deu-se pela sua experiência, tanto acadêmica quanto no mercado de desenvolvimento de software. Alguns integrantes deste grupo, inclusive, tiveram relação direta na definição de dois dos projetos executados e serão possíveis usuários desses aplicativos. Mesmo assim, os projetos foram apresentados tanto verbalmente quanto por meio de um material descritivo e, a exemplo da primeira fase do estudo de caso, os alunos tiveram um treinamento sobre conceitos de qualidade e modelos de qualidade, especialmente sobre as normas ISO/IEC 9126 e SQuaRE.

A proposta, nesta etapa, então, foi que os entrevistados apontassem, com base nos seus conhecimentos e no treinamento, quais as metas de qualidade compatíveis com os produtos de

software desenvolvidos pelos alunos de graduação, além dos papéis relacionados a cada meta e a proposta de possíveis métodos de avaliação que os integrantes do grupo conheciam.

Como resposta, houve certa compatibilidade entre as metas de qualidade escolhidas na primeira fase com as apresentadas nesta segunda fase do experimento (resumidas na Tabela 5.3). Porém, os entrevistados apontaram ter um grau elevado de dificuldades para sugerir possíveis métodos de avaliação para cada meta de qualidade. Algumas sugestões foram propostas, mas de maneira bastante informal e sem justificativas para as escolhas.

Tabela 5.3 – Visões de qualidade identificadas na segunda fase do estudo de caso.

Metas de Qualidade	Interessados	Métodos de Avaliação
Projeto – *CAU Mobile*		
Adequação Operacionalidade Adaptabilidade Compreensibilidade Segurança	Usuários finais	✓ *Checklist* e Inspeção
Testabilidade Modificabilidade	Equipe de desenvolvimento	✓ *Checklist* e Inspeção
Projeto – Sistema de Tutorias		
Funcionalidade Portabilidade Usabilidade	Usuários finais	✓ Testes funcionais de compatibilidade entre navegadores
Eficiência	Usuários finais	✓ Medidas de tempos de respostas por função
Confiabilidade	Equipe de desenvolvimento	✓ Unidade de tempo para recuperar o sistema
Manutenibilidade	Equipe de desenvolvimento	✓ Unidade de tempo para medir o processo de manutenção
Projeto – Sistema de Bolsistas		
Funcionalidade Confiabilidade Manutenibilidade Portabilidade	Clientes e usuários finais	✓ Testes funcionais ✓ Tempos de mudanças ✓ Validação e recuperação de dados ✓ Testes de *stress*
Usabilidade Eficiência	Clientes e usuários finais	✓ Número de chamadas ao suporte ✓ Tempos de desempenho

Além disso, os grupos foram questionados com relação à importância da avaliação da qualidade de produtos de software e o quanto uma ferramenta especializada poderia auxiliar as equipes de desenvolvimento na definição e na implementação da qualidade nesses produtos. Como resposta, os entrevistados convergiram suas ideias argumentando que a avaliação da qualidade pode levar à redução da incidência de erros, diminuição dos custos e do tempo de produção além de impactar positivamente na imagem da organização que os desenvolve. Ao mesmo tempo, houve consenso de que uma ferramenta de auxílio na definição e avaliação da qualidade em produtos de software é fundamental na aplicação dos conceitos de qualidade.

O uso de uma base de conhecimento, dessa forma, apoiada por uma ferramenta poderia auxiliar os avaliadores na associação de métodos de avaliação e métricas compatíveis. Um exemplo disso é o caso do projeto *CAU Mobile*, no qual os entrevistados, em ambas as etapas do estudo de caso, demonstraram ter uma noção bem definida de quais as metas de qualidade, tanto para o produto quanto para os artefatos, que melhor representariam os propósitos do projeto. Porém, não conseguiram definir possíveis métodos, métricas ou práticas para avaliar ou alcançar tais metas, e quando o fizeram, as escolhas se mostraram efêmeras e com pouco embasamento técnico, ou seja, sem justificativas.

Por ser uma aplicação cujo objetivo é facilitar o acesso às informações por meio de um dispositivo móvel (*smartphone*), a usabilidade foi apontada, pelos pesquisados, com a meta de qualidade mais importante no projeto *CAU Mobile*. Assim, sob o ponto de vista técnico, esta é uma meta que afeta a equipe de desenvolvimento como um todo, mas principalmente dos projetistas das interfaces que precisam que os requisitos sejam bem especificados nos artefatos construídos durante a fase de inicialização e análise do projeto. Em vista disso, a Figura 5-1 ilustra como o uso da abordagem – por meio da ferramenta desenvolvida – poderia ser usada para propor um plano de avaliação com o objetivo de avaliar e melhorar a usabilidade da aplicação *CAU Mobile* a partir dos artefatos gerados durante a fase de inicialização do projeto. É importante destacar que esta é apenas uma pequena parte do plano de qualidade e que para uma avaliação completa, os artefatos deveriam ser avaliados sob a ótica das outras metas de qualidade propostas e dos demais *stakeholders* envolvidos no projeto, para que possíveis sobreposições de métodos pudessem ser detectadas.

CAU Mobile - Oriented-Object Programming - RUP

Inception - RUP

Artifact			Language				Maturity
Use Case			UML				version 1.1
Purpose - Type		**Viewpoint**	**Quality Goal - Subgoal**				
Comprehension - Prediction		Developer	Usability - Operability				
Evaluation Method	**Level**	**Metric**	**Unit**	**Upper**	**Accep.**	**Lower**	**Practice**
Inspection	A	Number of Scenarios	Unit				Hiding the complexity
Artifact			**Language**				**Maturity**
Supplementary Specifications			Text				
Purpose - Type		**Viewpoint**	**Quality Goal - Subgoal**				
Comprehension - Prediction		User-interface Designer	Usability - Understandability				
Evaluation Method	**Level**	**Metric**	**Unit**	**Upper**	**Accep.**	**Lower**	**Practice**
Inspections by domain experts	C	Ease of Function Learning	Time	> 1.5min	<= 1.5min		Naming conventions
Artifact			**Language**				**Maturity**
Sequence diagram			UML				Version 1
Purpose - Type		**Viewpoint**	**Quality Goal - Subgoal**				
Improving understandability among stakeholders - Prediction		User-interface Designer	Usability - Understandability				
Evaluation Method	**Level**	**Metric**	**Unit**	**Upper**	**Accep.**	**Lower**	**Practice**
Consistency analysis tool	B	Number of Dependencies	Units				Statement insertion
Artifact			**Language**				**Maturity**
Component diagram			UML				Version 1
Purpose - Type		**Viewpoint**	**Quality Goal - Subgoal**				
Improving understandability among stakeholders - Prediction		User-interface Designer	Usability - Learnability				
Evaluation Method	**Level**	**Metric**	**Unit**	**Upper**	**Accep.**	**Lower**	**Practice**
Consistency analysis tool	B	Number of Dependencies	Units				Packaging conventions
Evaluation Method	**Level**	**Metric**	**Unit**	**Upper**	**Accep.**	**Lower**	**Practice**
Peer Review	B	Number of Dependencies	Units				Syntax checking

Figura 5-1 – Plano de Qualidade sugerido para avaliar a meta de qualidade usabilidade no projeto *CAU Mobile.*

5.6 Considerações

Por meio dos experimentos realizados como estudo de caso, percebeu-se que, embora qualidade seja um assunto bastante evidente na disciplina de Engenharia de Software e alvo de diversos estudos publicados na literatura, ainda existem dificuldades para as equipes de desenvolvimento identificarem quais são as metas de qualidade mais importantes para os produtos de software por elas desenvolvidos.

No que tange aos artefatos produzidos e a identificação de possíveis métodos e métricas de avaliação, essa dificuldade se torna ainda mais evidente. Paralelamente, mesmo que alguns desenvolvedores possuam, devido à experiência acumulada na área, certo grau de

discernimento sobre pontos a serem alcançados, esses pontos não são apoiados por uma estrutura definida, capaz de guia-los nas escolhas de como cada produto e/ou artefato deve ser avaliado com vistas à eficiência e eficácia da avaliação da qualidade.

6 TRABALHOS RELACIONADOS

A literatura dispõe de uma considerável quantidade e variedade de propostas, tanto para a avaliação de produtos de software como um todo, quanto para a avaliação de componentes ou artefatos individuais que compõem esses softwares. Boa parte dessas propostas foi influenciada, de alguma forma, pelos conceitos dos Modelos de Qualidade descritos no Capítulo 2 e, portanto, possuem elementos comuns, como a definição de *características de qualidade* e a especificação de *métricas* para medi-las. Porém, a forma como as ideias são estruturadas, a inclusão de outros elementos que complementam cada proposta e o relacionamento entre esses elementos, pode variar consideravelmente.

Assim, este capítulo destaca alguns trabalhos que compartilham algumas das definições apresentadas nesta dissertação. Com o objetivo de enfatizar as semelhanças e diferenças, as comparações são focadas nos elementos considerados mais importantes na avaliação de artefatos de software, como: a existência de um metamodelo para estruturar a proposta; os tipos de artefatos suportados; a dimensionalidade da abordagem – relacionando a avaliação dos artefatos com uma abordagem e/ou contexto de desenvolvimento, além dos interessados; e, por fim, se os trabalhos especificam algum processo de avaliação e alguma ferramenta de apoio. No final do capítulo, a Tabela 6.1 resume a abrangência desses elementos em cada trabalho.

Mohagheghi e Dehlen (2008, 2008a) descrevem um *framework* para a identificação e definição de atributos de qualidade com o objetivo de avaliar modelos usados em MDE. Tendo em vista que na abordagem *Model-Driven* os modelos são os principais artefatos e são submetidos a transformações automáticas e constantes ao longo de todo o processo de desenvolvimento, os autores argumentam que a melhoria desses modelos irá, consequentemente, melhorar a qualidade do produto final. Dessa forma, a proposta baseia-se na análise de uma série de fatores que envolvem a qualidade de modelos, tais como: seus propósitos, o processo de modelagem, as ferramentas usadas e o conhecimento dos modeladores, tanto no domínio do problema quanto no manuseio adequado das ferramentas e técnicas de modelagem.

Contudo, a identificação do *framework* descrito por Mohagheghi e Dehlen com esta dissertação está, principalmente, no fato de os autores definirem um metamodelo de qualidade cujo objetivo é relacionar a avaliação dos modelos às *metas de qualidade*. A estrutura proposta tem uma visão construtiva, ou seja, a ideia é identificar propriedades dos modelos que agreguem

qualidade e enfatizá-las por meio de práticas adequadas. Segundo os autores, é essencial avaliar os modelos para certificar-se que a qualidade desejada foi alcançada. Assim, a proposta inclui na estrutura do metamodelo a especificação de *métodos de avaliação*, compatíveis com os *propósitos* dos modelos dimensionados em relação aos usuários interessados.

Mesmo sendo uma abordagem inicialmente construída para avaliar modelos, a estrutura proposta destaca conceitos importantes da avaliação de artefatos de software, tais como a separação entre as *metas de qualidade* e os *meios* necessários para se alcança-las – tal como descreve Lindland *et al.* (1994) – além dos conceitos propostos por Dromey (1995) que busca identificar propriedades tangíveis dos artefatos, importantes para se alcançar as características de qualidade desejadas; estes dois últimos trabalhos, descritos no Capítulo 2.

Contudo, embora seja possível afirmar que o trabalho de Mohagheghi e Dehlen é flexível o suficiente para ser usado na avaliação de artefatos genéricos de software, ele especifica atributos e propriedades que são restritos à abordagem MDE. Além disso, os autores não definem explicitamente um processo de avaliação – apenas o metamodelo – nem descrevem uma possível ferramenta de auxílio.

Seguindo a mesma abordagem de desenvolvimento (MDE) Dominguez-Mayo *et al.* (DOMINGUEZ-MAYO et al., 2010) propuseram, também, um *framework* com o objetivo de avaliar artefatos de software, porém, voltado para o ambiente *web* (*Model-Driven Web Engineering* – MDWE). A argumentação dos autores para justificar a avaliação de artefatos usando a sua proposta é que, embora existam heurísticas e métricas bem conhecidas para avaliar características de qualidade nesses ambientes, como usabilidade, acessibilidade e desempenho, ainda falta rigor na definição e validação dos elementos quando combinados à abordagem MDE.

Todavia, embora destaque a necessidade de se definir um conjunto de métricas com base em condições específicas do domínio do projeto e do contexto dos artefatos, o trabalho é fortemente baseado nas ideias apresentadas pelos Modelos Hierárquicos. O metamodelo descrito se restringe a relacionar o conjunto de características e subcaracterísticas às métricas, que por sua vez, são decompostas em métricas básicas, derivadas e agregadas. Além disso, a abordagem não leva em conta a visão dos *stakeholders* na avaliação dos artefatos. No trabalho, até existe uma exemplificação de como a avaliação pode ser decomposta e quais são os passos a serem seguidos, porém, ela é restrita à usabilidade e não é descrito explicitamente um processo completo de avaliação nem uma ferramenta de apoio.

Paralelamente, em seu trabalho, Berti-Équille *et al.* (BERTI-ÉQUILLE et al., 2011) argumentam ser possível combinar e explorar uma variedade de características de qualidade por meio da quantificação de dados gerados durante um processo de software, por exemplo: número de e-mails retornados pelos clientes; quantidade de erros descobertos pelos clientes; porcentagem de dados que satisfazem uma determinada regra sintática; entre outros. Dessa forma, os autores propõem um modelo de avaliação multidimensional que intenciona capturar esses dados, armazená-los em um banco de dados, e então definir uma base de conhecimento com o objetivo de ser reusada em projetos futuros.

Assim, a estrutura do banco de dados é baseada em um metamodelo definido sob cinco perspectivas – que os autores chamaram de blocos – que incluem a especificação das características de qualidade; as dimensões relacionadas a essas características que, no caso, especificam as métricas, os atores e o contexto operacional; o conjunto de métodos que podem ser usados em cada avaliação; os objetos a serem avaliados; e, finalmente, um bloco que relaciona a avaliação da qualidade dos artefatos aos dados coletados acrescidos de informações relevantes, tais como descrição do projeto, data e período de execução.

O processo de avaliação é baseado na abordagem GQM dividido em três etapas: i – definição dos propósitos de avaliação e definição dos objetos a serem medidos, incluindo *diagnóstico de qualidade; seleção de métricas* e *correlação de métricas*; ii – definição do escopo da avaliação, abrangendo *dados recomendados* e *tarefas recomendadas* para avaliação; e, iii – predição da qualidade que objetiva computar as informações e projetar uma perspectiva de qualidade para os produtos avaliados.

Tal como esta dissertação, um dos objetivos desse trabalho é armazenar dados relativos à avaliação da qualidade de software visando a sua reutilização. Isso, ajustado a perspectiva dos usuários interessados. Contudo, o trabalho descrito por Berti-Équille *et al.* avalia os artefatos já finalizados – preferencialmente modelos de dados e código fonte, e não enquanto eles são gerados ou transformados durante o processo de software. Ou seja, o propósito da avaliação visa identificar a qualidade final do produto. Além disso, embora descrevam uma ferramenta de auxílio, não há uma definição clara de como as características de qualidade são relacionadas com as métricas e propósitos dos artefatos.

Dubielewicz *et al.* (DUBIELEWICZ et al., 2006), por sua vez, propõem um metamodelo para avaliar produtos de software com o propósito de ser aplicado em diferentes estágios do processo de desenvolvimento, sobre artefatos diversos como: código, modelos, especificações e, inclusive, sobre o produto final como um todo. A visão dos interessados, por sua vez, é

definida segundo diferentes perspectivas, incluindo usuários (perspectiva externa), desenvolvedores (perspectiva interna) e organização (perspectiva do negócio). Um ponto tangente entre esse trabalho e a proposta descrita nesta dissertação é a flexibilidade na qual as especificações de qualidade – metas de qualidade – podem ser informalmente expressas pelos usuários com base nas suas necessidades e/ou conhecimentos.

O metamodelo apresentado é dividido, basicamente, em três partes: a primeira delas relaciona as características de qualidade aos atributos dos artefatos capazes de serem medidos e as suas respectivas métricas, descrevendo o que os autores chamaram de *elementos do metamodelo de qualidade*. Uma segunda parte define os requisitos de qualidade de software (propósitos de qualidade) inerentes a cada artefato em avaliação e tem por objetivo instanciar os elementos do metamodelo para um determinado produto de software. Por fim, a terceira parte do metamodelo relaciona a estrutura definida anteriormente à ideia de avaliação dos artefatos, embora nenhum processo de avaliação tenha sido especificado no trabalho.

Um ponto importante desse trabalho está no fato de os autores enfatizarem a necessidade de reter as informações coletadas durante a avaliação e relacioná-las com os requisitos especificados durante a fase de projeto de forma que as expectativas de qualidade dos usuários com o produto final sejam atendidas. Porém, não há uma descrição de como isso é feito. Os autores não descrevem, como mencionado, nenhum processo de avaliação capaz de guiar a equipe de desenvolvimento na escolha de métricas adequadas para cada artefato. Em outras palavras, a ênfase dessa proposta está na especificação, classificação, e posterior armazenamento, de métricas para avaliar características de qualidade coletadas durante a fase de análise dos requisitos, embora a proposta indique que a avaliação dos artefatos deva ocorrer durante todo o processo de desenvolvimento.

Vanderose e Habra (VANDEROSE; HABRA, 2011), a exemplo dos trabalhos relacionados descritos até agora, também propõem um *framework* com o objetivo de avaliar artefatos de software. A diferença está no fato de que, embora esse trabalho seja voltado prioritariamente à avaliação de modelos, ele especifica uma abordagem genérica, que pode ser extensível à avaliação de artefatos gerados pelas várias metodologias de desenvolvimento de software. Segundo os autores, a abordagem é um modelo integrado de avaliação capaz de representar os vários fatores de qualidade para cada projeto, além dos relacionamentos entre esses fatores e os diversos métodos de avaliação usados na medição das entidades. Porém, em vez de determinar modelos de qualidade específicos para vários produtos de software, o objetivo é que o *framework* proposto ajude na construção de um modelo que concentre os

esforços no planejamento necessário à avaliação dos artefatos do projeto, como ilustrado na Figura 6-1. Contudo, essa não é uma tarefa trivial e pode requerer a assistência de especialistas.

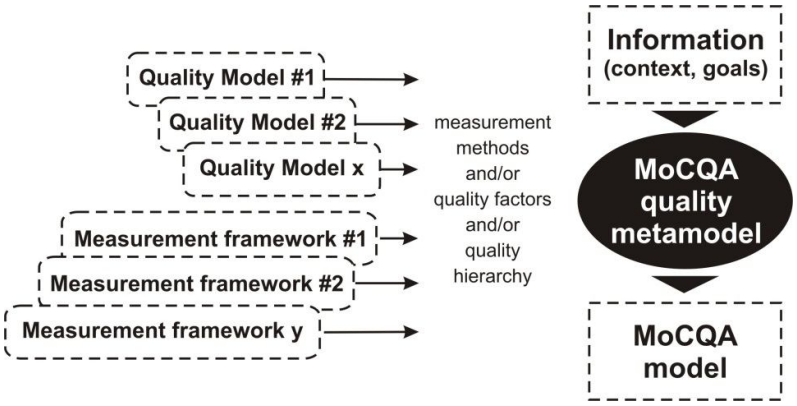

Figura 6-1 – Integração de modelos de qualidade e métodos de medição proposta por Vanderose e Habra (2011).

Para estruturar a proposta, os autores especificam um metamodelo que captura os conceitos dos modelos tradicionais como a ISO/IEC 9126 e alinham métodos de avaliação aos interesses dos vários *stakeholders* envolvidos. A metodologia de avaliação, por sua vez, é baseada nos princípios da abordagem GQM (BASILI; CALDIERA; ROMBACH, 1994) e inclui cinco passos: i – capturar as informações contextuais (*stakeholders*, motivações, tempo, orçamento); ii – inserir as informações coletadas na estrutura do metamodelo; iii – definir o plano de avaliação genérico; iv – submeter os artefatos selecionados ao plano de avaliação; e, v – interpretar os resultados.

Dentre os trabalhos relacionados, esse é o que mais se assemelha com a proposta descrita nesta dissertação. A diferença principal está no fato de que, no trabalho de Vanderose e Habra, as terminologias referentes às características de qualidade somente podem ser definidas com base em conceitos existentes. Em outras palavras, não é permitido à equipe de desenvolvimento especificar suas próprias metas de qualidade, mas sim extraí-las de padrões ou modelos já existentes.

Tabela 6.1 – Comparação dos trabalhos relacionados.

Trabalho	Abordagem	Tipos de Artefatos	Metamodelo	Processo de Avaliação	Ferramenta de Apoio
Mohagheghi e Dehlen (2008, 2008a)	MDE	Apenas Modelos	Sim, completo	Não	Não
Dominguez-Mayo *et al.* (2010)	MDWE	Apenas Modelos	Parcial	Específico, incompleto	Não
Berti-Équille *et al.* (2011)	Não descreve	Modelos de dados e códigos-fonte	Sim, completo	GQM	Sim
Dubielewicz *et al.* (2006)	Não descreve	Genérico a vários tipos de artefatos	Parcial	Não	Não
Vanderose e Habra (2011)	Genérico a várias abordagens	Prioritaria-mente Modelos	Sim, completo	GQM	Sim
Esta Dissertação	**Genérico a várias abordagens**	**Genérico a vários tipos de artefatos**	**Sim, completo**	**Baseado na ISO/IEC 14598**	**Sim**

7 CONCLUSÕES E COMENTÁRIOS FINAIS

Os benefícios de se desenvolver produtos de software com qualidade agregada são diversos, disseminados e bem conhecidos. Contudo, a importância da avaliação da qualidade desses produtos vai além de questões comerciais ou de segurança, pois tem o objetivo de fornecer resultados qualitativos e, sempre que possível, quantitativos sobre a qualidade do software produzido e, além disso, fornecer o *feedback* necessário para o aprimoramento dos processos de software. Porém, para que sejam efetivos esses resultados precisam ser compreensíveis, confiáveis e em conformidade com o ambiente que cercam as avaliações.

Em vista disso, diversos modelos têm sido apresentados ao longo do tempo com o propósito de definir conceitos e estruturar a avaliação da qualidade de produtos de software. Alguns desses modelos definem uma estrutura fixa e hierárquica na qual determinadas características de qualidade são decompostas em subcaracterísticas e propriedades mais restritas e, então, medidas por meio de métricas pré-estabelecidas. Porém, a rigidez desses modelos tende a dificultar sua adaptação a contextos organizacionais específicos, além de existirem relados sobre inconsistências e sobreposições de termos relacionados ao seu uso. Em contrapartida, outras propostas defendem a ideia de uma estrutura aberta que possa ser adaptada e aplicada conforme as necessidades requeridas por cada projeto. Todavia, a maioria dos trabalhos baseados nessa ideia aborda a avaliação com vistas a um único contexto de desenvolvimento ou são propostas com base na verificação e validação do produto gerado ao final do processo de desenvolvimento.

Assim, este trabalho propôs uma abordagem sistemática para a definição de um *framework* de qualidade, cujo propósito é fornecer uma estrutura linear aos avaliadores, engenheiros, e demais interessados, na validação de artefatos de software gerados ao longo de um processo de desenvolvimento de software. O *framework* é composto por um metamodelo, e inclui ainda um processo de avaliação e uma ferramenta de apoio.

O metamodelo, por sua vez, define os elementos conceituais – e suas relações – necessários para organizar a avaliação dos artefatos, tendo em vista os seus propósitos, os papéis interessados, as metas de qualidade correspondentes, bem como possíveis métricas e métodos compatíveis. O metamodelo resulta, ainda, de um alinhamento conceitual dos vários trabalhos pesquisados, dedicados à definição de elementos e de uma terminologia unificada para avaliação da qualidade de software.

Ao mesmo tempo, o processo de avaliação definido, institui a sequência de passos necessários para relacionar os artefatos com as metas de qualidade pretendidas e auxiliar na escolha de métodos de avaliação, métricas e práticas de melhorias de software adequadas aos propósitos de qualidade pretendidos para cada um dos artefatos e/ou para o sistema como um todo.

A validação da abordagem proposta foi realizada por meio de um estudo de caso envolvendo três projetos de software. Cada um dos projetos foi submetido à validação por dois grupos de estudantes/desenvolvedores para que questões referentes à qualidade fossem definidas e elucidadas. Durante o experimento verificou-se que, embora o tema seja recorrente na disciplina de Engenharia de Software, ainda existem dúvidas e/ou mal entendimento sobre seus conceitos e as formas de como se avaliar produtos de software.

Por fim, conclui-se que, ainda que este trabalho não seja definitivo no que tange a avaliação da qualidade de artefatos de software – e nem é o propósito desta dissertação – ele atingiu seus objetivos ao trazer à tona questões e conceitos importantes em relação à qualidade desses produtos, abordando conceitos, problemas e possíveis soluções.

7.1 Contribuições

Esta dissertação apresenta, como principal contribuição, a proposta de um mecanismo de avaliação da qualidade de artefatos de softwares. Contudo, outras contribuições incluem:

- Desenvolvimento de uma abordagem teórica para avaliação da qualidade dos artefatos gerados durante um processo de software;
- A proposta de um *framework* de qualidade, composto por um metamodelo e uma sequência lógica de passos (processo), cujo objetivo é organizar os conceitos que envolvem a definição de metas de qualidade e seus respectivos métodos e métricas de avaliação;
- O desenvolvimento de um protótipo de ferramenta com vistas a auxiliar os avaliadores na definição de um plano de qualidade dos artefatos gerados ao longo de um processo de software;

- A proposta de uma base de conhecimento inicial, como sugestão, para que ações tomadas para avaliar a qualidade em projetos passados possam ser reusadas em projetos futuros com o objetivo de armazenar o conhecimento adquirido e reusá-lo sempre que útil e/ou necessário.

7.2 Trabalhos Futuros

Uma das limitações associadas à abordagem descrita neste trabalho está no fato de a definição e, principalmente, a escolha dos elementos que compõem o plano de avaliação dos artefatos ser fortemente dependente da intervenção humana, ou seja, do conhecimento e experiência dos avaliadores. Como melhoria para trabalhos futuros, pretende-se estender este processo de forma que a abordagem seja automatizada por meio de uma heurística apropriada.

Paralelamente, pretende-se utilizar o conhecimento desenvolvido neste trabalho para integrá-lo a uma abordagem mais ampla de desenvolvimento de software. Uma das possibilidades é a sua integração com modelos de adaptação de Processos de Software. A adaptação de processos está relacionada ao ato de customizar uma definição de processo de desenvolvimento de software para atender necessidades específicas de um projeto, ajustar um processo de desenvolvimento de acordo com diferentes ambientes ou contextos (GINSBERG; QUINN, 1995). Diversas abordagens, descritas na literatura, descrevem esse tema. Entre as mais referenciadas estão as Linhas de Processos de Software (*Software Process Lines* – SPL) (RUIZ; HURTADO, 2012; SIMMONDS et al., 2011) e *Situational Method Engineering* (SME) (HENDERSON-SELLERS; RALYTÉ, 2010). Essas abordagens são baseadas na classificação e seleção de fragmentos – que podem ser atividades e/ou artefatos – descritos por um *processo padrão da organização* (PSPO). Ou seja, a qualidade dos artefatos (descrita aqui) pode ser associada à seleção desses fragmentos para compor um plano de qualidade a ser observado em processos adaptados.

7.3 Publicações

- **Simpósio Brasileiro de Qualidade de Software (SBQS):** Bertuol, G., Lorenz, W. G. e Fontoura, L. M. “Avaliação da Qualidade de Modelos em um Processo MDE” In: XII Simpósio Brasileiro de Qualidade de Software. Salvador, Brasil, 2013.

REFERÊNCIAS

ABNT. IEC 9126-1: 2003-Engenharia de software-Qualidade de produto-Parte 1: Modelo de qualidade. **Rio de Janeiro: ABNT**, p. 1–21, 2003.

ADJOYAN, S.; SERIAI, A.-D.; SHATNAWI, A. **Service Identification Based on Quality Metrics**Proceedings of the 26 International Conference on Software Engineering & Knowledge Engineering (SEKE2014). **Anais**...2014

AGGARWAL, K. et al. Application of Artificial Neural Network for Predicting Maintainability Using Object-Oriented Metrics. **International Science Indes**, v. 2, p. 990–994, 2008.

AL-KILIDAR, H.; COX, K.; KITCHENHAM, B. The Use and Usefulness of the ISO/IEC 9126 Quality Standard. **International Symposium on Empirical Software Engineering.**, p. 126–132, 2005.

AL-QUTAISH, R. E. Quality Models in Software Engineering Literature: An Analytical and Comparative Study. **Journal of American Science**, v. 6, n. 3, p. 166–175, 2010.

ANTOLIÉ, Z. Software development quality plan-Organizational assignment on projects. **Proceedings of the 33rd International Convention (MIPRO 2010)**, p. 396 – 401, 2010.

ARPAIA, P. et al. A Model-Driven Domain-Specific Scripting Language for Measurement-System Frameworks. **IEEE Transactions on Instrumentation and Measurement**, v. 60, n. 12, p. 3756–3766, dez. 2011.

ARRUDA, D. F. DE. Métricas de Software: Problema ou Solução? **Engenharia de Software Magazine Ed.67**, p. 38–42, 2014.

BACH, J. The Challenge of“ Good Enough” Software. **American Programmer Magazine**, p. 1–11, 1996.

BACH, J. Good Enough Quality : Beyond the Buzzword. **IEEE Computer Society**, p. 96–98, 1997.

BACH, J. A Framework for Good Enough Testing. **IEEE Computer Society**, p. 124–126, 1998.

BASILI, V. R. Software Modeling and Measurement: the Goal Question Metric Paradigm. **Computer Science Technical Report Series**, 1992.

BASILI, V. R. et al. GQM+Strategies: A Comprehensive Methodology for Aligning Business Strategies with Software Measurement. **In Proceedings of the DASMA Software Metric Congress - MetriKon2007**, p. 253–266, 2007.

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. The Goal Question Metric Approach. **Encyclopedia of Software Engineering**, v. 2, p. 1–10, 1994.

BERTI-ÉQUILLE, L. et al. Assessment and analysis of information quality: a multidimensional model and case studies. **International Journal of Information Quality**, v. 2, n. 4/2011, p. 300–323, 2011.

BOEHM, B. W. et al. **Characteristics of Software Quality**. [s.l.] North-Holland Pub. Co., 1978.

BOEHM, B. W.; BROWN, J. R.; LIPOW, M. **Quantitative Evaluation of Software Quality**Proceedings of the 2nd International Conference on Software Engineering. **Anais**...: ICSE '76.Los Alamitos, CA, USA: IEEE Computer Society Press, 1976

BOMMEL, P. VAN et al. **QoMo: A modeling process quality framework based on SEQUAL**Proceedings of the Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD'07), held in conjunctiun with the 19th Conference on Advanced Information Systems (CAiSE'07). **Anais**...2007

BOUWERS, E.; VAN DEURSEN, A.; VISSER, J. Evaluating Usefulness of Software Metrics: an Industrial Experience Report. **Software Engineering in Pratice track (SEIP) of International Conference on Software Engineering**, p. 921–930, maio 2013.

BREIVOLD, H. P.; CRNKOVIC, I. **Analysis of Software Evolvability in Quality Models**35th Euromicro Conference on Software Engineering and Advanced Applications, SEAA '09. **Anais**...2009

BRIAND, L. C.; WÜST, J. Empirical studies of quality models in object-oriented systems. In: ZELKOWITZ, M. V (Ed.). **Advances in Computers**. Advances in Computers. [s.l.] Elsevier, 2002. v. 56p. 97–166.

CACHERO, C.; CALERO, C.; POELS, G. Metamodeling the Quality of the Web Development Process' Intermediate Artifacts. In: BARESI, LUCIANO AND FRATERNALI, PIERO AND HOUBEN, G.-J. (Ed.). **Web Engineering**. [s.l.] Springer Berlin Heidelberg, 2007. p. 74–89.

DAVIS, F. D. Information Technology Introduction. **MIS Quarterly**, v. 13, n. 3, p. 319–340, 1989.

DE ABREU, T. C.; MOTA, L. DA S.; ARAÚJO, M. A. P. Métricas de Software: Como utilizá-las no gerenciamento de projetos de software. **Engenharia de Software Magazine Ed.21**, p. 50–55, 2010.

DEISSENBOECK, F. et al. Software Quality Models: Purposes, Usage Scenarios and Requirements. **Proceeding WOSQ'09 Proceedings of the Seventh ICSE Conference on Software Quality**, n. IEEE Computer Society Washington, DC, USA, p. 9–14, 2009.

DOMINGUEZ-MAYO, F. J. et al. A Quality Model in a Quality Evaluation Framework for MDWE methodologies. **2010 Fourth International Conference on Research Challenges in Information Science (RCIS)**, p. 495–506, maio 2010.

DROMEY, G. R. A model for software product quality. **IEEE Transactions on Software Engineering**, v. 21, n. 2, p. 146–162, 1995.

DUBIELEWICZ, I. et al. **Software Quality Metamodel for Requirement, Evaluation and Assessment**ISIM06 Conference. **Anais**...2006

ELBERZHAGER, F.; MÜNCH, J. Using Early Quality Assurance Metrics to Focus Testing Activities. **Proceedings of the International Conference on Software Process and Product Measurement (MetriKon)**, p. 29–36, 2011.

EL-KORANY, A.; NABIL, D.; ELDIN, A. S. Quality Measurement Model for KADS-Domain Knowledge. **Journal of Software Engineering**, v. 4, n. 1, p. 30–43, 2010.

FENTON, N. E.; PFLEEGER, S. L. **Software Metrics: A Rigorous and Practical Approach**. 2nd. ed. Boston, MA, USA: International Thomson Computer Press, 1996.

FREY, A. G. et al. QUIMERA: a quality metamodel to improve design rationale. **Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems - EICS'11**, p. 265–270, 2011.

FREY, A. G. et al. QUIMERA-Toward an Unifying Quality Metamodel. **INFORSID**, v. X, n. x/année, p. 1–10, 2012.

GARVIN, D. A. What Does "Product Quality" Really Mean? **Sloan management review**, v. 26, n. 1, p. 25–43, 1984.

GINSBERG, M. P.; QUINN, L. H. Process Tailoring and the Software Capability Maturity Model. **Software Engineering Institute**, 1995.

GRADY, R. B.; CASWELL, D. L. **Software metrics: establishing a company-wide program**. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1987.

GUERRA, A. C.; COLOMBO, R. M. T. **Tecnologia da Informação: Qualidade de Produto de Software**. Brasilia: PBQP Software, 2009. p. 259

HENDERSON-SELLERS, B.; RALYTÉ, J. Situational Method Engineering : State-of-the-Art Review. **Jornal of Universal Computer Science**, v. 16, n. 3, p. 424–478, 2010.

IEEE. **IEEE Standard for a Software Quality Metrics Methodology**. IEEE Std 1 ed. [s.l.] IEEE Computer Society, 2009. v. 1998p. 1061–1998

ISO/IEC. **ISO/IEC 14598 - Information Technology - Software Product Evaluation. Parts 1-5.** [s.l: s.n.].

ISO/IEC. **Software Engineering - Product Quality, {ISO/IEC} 9126-1**. [s.l: s.n.].

ISO/IEC. **Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE.** [s.l: s.n.].

KAN, S. H. **Metrics and Models in Software Quality Engineering**. Second Edi ed. [s.l.] Wesley, Addison, 2002. p. 560

KHOSHGOFTAAR, T. M. et al. Data Mining For Predictors of Software Quality. **International Journal of Software Engineering and Knowledge Engineering**, v. 09, n. 05, p. 547–563, 1999.

KHOSHGOFTAAR, T. M.; ALLEN, E. B. Logistic regression modeling of software quality. **International Journal of Reliability, Quality and Safety Engineering**, v. 06, n. 04, p. 1–15, 1999.

KLÄS, M. et al. How to Evaluate Meta-Models for Software Quality? **International Workshop on Software Measurement (IWSM) Software Metrik Kongress (MetriKon)**, 2010.

KLÄS, M.; LAMPASONA, C.; MÜNCH, J. Adapting Software Quality Models: Practical Challenges, Approach, and First Empirical Results. **37th EUROMICRO Conference on Software Enginnering and Advanced Applications**, p. 341–348, ago. 2011.

KROGSTIE, J. Evaluating UML using a generic quality framework. **UML and the Unified Process**, v. 1, 2003.

KROGSTIE, J.; LINDLAND, O. I.; SINDRE, G. **Defining quality aspects for conceptual models**Proceedings of the IFIP international working conference on Information system concepts: Towards a consolidation of views. **Anais**...London, UK, UK: Chapman & Hall, Ltd., 1995a

KROGSTIE, J.; LINDLAND, O. I.; SINDRE, G. Towards a deeper understanding of quality in requirements engineering. In: IIVARI, J.; LYYTINEN, K.; ROSSI, M. (Eds.). **Advanced Information Systems Engineering SE - 7**. Lecture Notes in Computer Science. [s.l.] Springer Berlin Heidelberg, 1995b. v. 932p. 82–95.

LANGE, C. F. J.; CHAUDRON, M. R. Managing model quality in UML-based software development. **13th IEEE International Workshop on Software Technology and Engineering Practice - STEP'05**, p. 15–23, 2005.

LINDLAND, O. I. **A Prototyping Approach to Validation of Conceptual Models in Information Systems Engineering**. [s.l.] Faculty of Electrical Enginnering and Computer Science, 1993.

LINDLAND, O. I.; SINDRE, G.; SOLVBERG, A. Understanding quality in conceptual modeling. **Software, IEEE**, p. 42–49, 1994.

LIU, X.; PANG, J. A fuzzy synthetic evaluation method for software quality. **e-Business and Information System Security (EBISS)**, n. m, p. 1–4, maio 2010.

MAES, A.; POELS, G. Evaluating quality of conceptual modelling scripts based on user perceptions. **Data & Knowledge Engineering**, v. 63, n. 3, p. 701–724, dez. 2007.

MALHOTRA, N.; PRUTHI, S. An Efficient Software Quality Models for Safety and Resilience. **International Journal of Recent Technology and Engineering - IJRTE**, v. 1, n. 3, p. 66–70, 2012.

MCCALL, J. A.; RICHARDS, P. K.; WALTERS, G. F. Factors in Software Quality. **Nat'l Tech. Information Servicel**, v. 1, 2 and 3, n. ADA049055, 1977.

MEHMOOD, K.; CHERFI, S. S.-S.; COMYN-WATTIAU, I. Data Quality through Conceptual Model Quality-Reconciling Researchers and Practitioners through a Customizable Quality Model. **14th Internation Conference on Information Quality - ICIQ**, p. 61–74, 2009.

MENS, T.; LANZA, M. A graph-based metamodel for object-oriented software metrics. **Electronic Notes in Theoretical Computer Science**, 2002.

MOHAGHEGHI, P. An Approach for Empirical Evaluation of Model-Driven Engineering in Multiple Dimensions. **From Code Centric to Model Centric: Evaluation the Effectiveness of MDD (C2M:EEMDD)**, 2010.

MOHAGHEGHI, P.; AAGEDAL, J. Evaluating quality in model-driven engineering. **International Workshop on Modeling in Software Engineering (MISE'07)**, p. 0–5, 2007.

MOHAGHEGHI, P.; DEHLEN, V. An overview of quality frameworks in model-driven engineering and observations on transformation quality. **Workshop on Quality in Modeling**, 2007.

MOHAGHEGHI, P.; DEHLEN, V. A Metamodel for Specifying Quality Models in Model-Driven Engineering. **Nordic Workshop on Model-Driven Engineering**, 2008a.

MOHAGHEGHI, P.; DEHLEN, V. Developing a quality framework for model-driven engineering. **Models in Software Engineering**, 2008b.

MOHAGHEGHI, P.; DEHLEN, V. **Existing model metrics and relations to model quality**2009 ICSE Workshop on Software Quality. **Anais**...IEEE, maio 2009Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5071555>. Acesso em: 12 dez. 2013

MOHAGHEGHI, P.; DEHLEN, V.; NEPLE, T. Towards a tool-supported quality model for model-driven engineering. **Proc. 3rd International Workshop on Quality in Modeling.**, 2008.

MOHAGHEGHI, P.; DEHLEN, V.; NEPLE, T. Definitions and approaches to model quality in model-based software development – A review of literature. **Information and Software Technology**, v. 51, n. 12, p. 1646–1669, dez. 2009.

MOODY, D. L. et al. Evaluating the quality of information models: empirical testing of a conceptual model quality framework. **Proceedings of the 25th International Conference on Software Engineering (ICSE'03)**, v. 6, 2003.

MOODY, D. L. Measuring the quality of data models: an empirical evaluation of the use of quality metrics in practice. **Proceedings of the 11th European Conference on Information Systems, ECIS 2003, Naples, Italy**, 2003.

MOODY, D. L.; SHANKS, G. G. What makes a good data model? Evaluating the quality of entity relationship models. In: LOUCOPOULOS, P. (Ed.). **Entity-Relationship Approach — ER '94 Business Modelling and Re-Engineering SE - 7**. Lecture Notes in Computer Science. [s.l.] Springer Berlin Heidelberg, 1994. v. 881p. 94–111.

MORRIS, C. W. **Foundations of the theory of signs**. University ed. [s.l.] University of Chicago Press, 1938. p. 59

NELSON, H. J. et al. A conceptual modeling quality framework. **Software Quality Journal**, v. 20, n. 1, p. 201–228, 19 abr. 2011.

PMI. **A Guide to the Project Management Body of Knowledge**. 5th. ed. Newtown Square, PA: Project Management Institute, 2013.

PRESSMAN, R. S. **Software Engineering: A Practitioner's Approach**. 7. ed. New York, NY, USA: McGraw-Hill, Inc., 2010. p. 930

PUNTER, T. et al. The W-Process for Software Product Evaluation: A Method for Goal-Oriented Implementation of the ISO 14598 Standard. **Software Quality Journal**, v. 12, n. 2, p. 137–158, jun. 2004.

PUNTER, T.; SOLINGEN, R. VAN; TRIENEKENS, J. Software Product Evaluation - Current Status and Future Needs for Customers and Industry -. **4th IT Evaluation Conference - EVIT97**, 1997.

ROSENBERG, L. H.; SHEPPARD, S. B. Metrics in software process assessment, quality assurance and risk assessment. **Proceedings of 1994 IEEE 2nd International Software Metrics Symposium**, p. 10–16, 1994.

RUIZ, P. H.; HURTADO, J. A. A software process line based on the Unified Process. **2012 7th Colombian Computing Congress (CCC)**, p. 1–6, out. 2012.

SAMADHIYA, D.; WANG, S.-H.; CHEN, D. Quality Models: Role and Value in Software Engineering. **2nd International Conference on Software Technology and Engineering**, v. 1, p. 320–324, out. 2010.

SCHMIDT, D. Model-Driven Engineering. **Computer Society - IEEE**, n. February, p. 25–31, 2006.

SEFFAH, A. et al. Usability measurement and metrics: A consolidated model. **Software Quality Journal**, v. 14, n. 2, p. 159–178, jun. 2006.

SHANKS, G. G.; DARKE, P. Quality in Conceptual Modeling - Linking Theory and Practice.pdf. **Pacific Asia Conference on Information Systems - PACIS**, p. 805–814, 1997.

SIMMONDS, J. et al. Modeling Variability in Software Process Models ∗. **Technical Report TR/DCC-2011-10, Universidad de Chile, Departamento de Ciencias de la Computacion**, 2011.

SINGH, B.; KANNOJIA, S. P. **A Review on Software Quality Models**2013 International Conference on Communication Systems and Network Technologies. **Anais**...IEEE Comput. Soc, abr. 2013Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6524514>. Acesso em: 22 ago. 2013

SINGH, M.; CHAWLA, E. R. Identification of the Most Adaptable Quality Meta-Model. **International Journal of Electronics and Computer Science Engineering (IJECSE)**, v. 3, n. 1, p. 231–232, 2012.

SOLHEIM, I.; NEPLE, T. **Model Quality in the Context of Model-Driven Development.**Model-Driven Enterprise Information Systems - MDEIS. **Anais**...2006

SURYN, W.; ABRAN, A. ISO/IEC SQuaRE: The Second Generation of Standards for Software Product Quality. **International Association of Science and Technology for Development - IASTED**, p. 1–11, 2003.

TERNITÉ, T. Process Lines: A Product Line Approach Designed for Process Model Development. **2009 35th Euromicro Conference on Software Engineering and Advanced Applications**, p. 173–180, 2009.

THAKURTA, R. A framework for prioritization of quality requirements for inclusion in a software project. **Software Quality Journal**, v. 21, n. 4, p. 573–597, 25 out. 2012.

TRENDOWICZ, A.; PUNTER, T. Quality Modelling for Software Product Lines. **7th ECOOP Workshop on Quantitative Approach in Object-Oriented Software Engineering - QAOOSE'03**, 2003.

USMAN, M. et al. A Survey of Consistency Checking Techniques for UML Models. **2008 Advanced Software Engineering and Its Applications**, p. 57–62, dez. 2008.

VANDEROSE, B. Supporting a model-driven and iterative quality assessment methodology: The MoCQA framework. 2012.

VANDEROSE, B.; HABRA, N. Tool-Support for a Model-Centric Quality Assessment: QuaTALOG. **2011 Joint Conference of the 21st International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement**, p. 263–268, nov. 2011.

WAGNER, S. Using Economics as Basis for Modelling and Evaluating Software Quality. **Proceedings of the 29th International Conference on Software Engineering Workshops - ICSEW'07 (2007) IEEE Computer Society Washington, DC, USA**, p. 2–5, 2007.

WAGNER, S.; DEISSENBOECK, F. An integrated approach to quality modelling. **Fifth International Workshop on Software Quality In: Proc. of ICSE'07, 6p. (2007)**, 2007.

ANEXO A – EXEMPLOS DE DADOS

Este anexo apresenta alguns exemplos de dados que podem ser usados para compor a base de conhecimento, definida pelo metamodelo de qualidade descrito no Capítulo 3. Cada metaclasse é apresentada e identificada nas tabelas a seguir e seus atributos organizados nas colunas das respectivas tabelas. Contudo, a relação é meramente ilustrativa, sem a pretensão de ser completa ou definitiva, apenas com o propósito de exemplificar possíveis dados que podem ser usados para povoar a base de conhecimento.

Tabela A.1 – Exemplificação de dados para a metaclasse *Approaches*.

Approaches	
Name	**Description**
Model-Driven Engineering (MDE)	MDE is a software development methodology which focuses on creating and exploiting domain models (that is, abstract represetnations of the knowledge and activities that govern a particular application domain), rather than on the computing concepts.
Oriented-Object Programming	It is a programming paradigm that represents the concept of "objects" that have data fields (attributes that describe the object) and associated procedures known as methods. Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs.
Software Product Line	It refers to software engineering methods, tools and techniques for creating a collection of similar software systems from a shared set of software assets using a common means of production.

Tabela A.2 – Exemplificação de dados para a metaclasse *Artifacts*.

Artifacts				
Name	**Life Cycle Phase**	**Description**	**Language**	**Maturity**
Use Cases	Inception (RUP)	Use case for login.	UML	Version 1.1
Requirements Attributes	Inception (RUP)	A repository of project requirements attributes and dependencies to assist managing change from a requirements perspective.	UML	Complete
Supplementary Specifications	Inception (RUP)	This artifact capture system requirements that are not readily captured in behavioral requirements artifacts such as use-case specifications.	Text	Complete
Change Request	Transition (RUP)	These artifacts are used to document and track requests for a change to the product.	Text	Complete
Testability Class	Construction (RUP)	A specialized class in the design model that represents test-specific behavior that the software will support.	jUnit	Version 1.1

Tabela A.3 – Exemplificação de dados para a metaclasse *Purpose*.

Purposes		
Name	**Description**	**Type**
Communication	The artifact enables efficient communication about the elements of the systems, their behavior and the design decisions. Communication includes communication during the development phase with different stakeholders and documentation for understanding the system in later phases such as maintenance. (Lange & Chaudron, 2005).	Prediction
Documentation	The artifact is used as a form of system documentation for future reference and agreements.	Prediction
Implementation	The artifact is used as basis for (manual) implementation of the source code of the system.	Assessment
Analysis	The purpose of the artifact is to explore and analyze the problem domain including its key concepts and making some early design decisions. (Lange & Chaudron, 2005).	Assessment

Tabela A.4 – Exemplificação de dados para a metaclasse *Viewpoint*.

Viewpoint	
Name	**Description**
Project Manager	This role plans, manages and allocate resources, shapes priorities, coordinates interactions with customers and user, and keeps the project team focused (RUP).
Customer	It defines the project and sets its goals. The more accurate his/her work and the more frequent his/her involvement, the greater the chances the project will succeed (RUP).
Development Team	Who are responsible for interpreting the models and customers' requirements and transform them into executable source code.
Database Designer	He is responsible for designing the persistent data storage to be used by the system, description for most application development projects, the technology used for persisting data in a database (RUP).
Users	It is who will use the applications developed.

Tabela A.5 – Exemplificação de dados para a metaclasse *QualityType*.

Quality Type		
Name	**Description**	**Source**
Internal Quality	Static measures of the code.	ISO/IEC 9126
External Quality	Measuring the behavior of the code when executed.	ISO/IEC 9126
Quality in Use	Users view of quality.	ISO/IEC 9126
Syntactic Quality	Describes the relation among language constructs without considering their meaning. Syntactic quality is how well the artifact corresponds to the language.	Lindland *et al.* (1994).
Semantic Quality	Is how well an artifact corresponds to the domain or the knowledge of people from the domain.	Lindland *et al.* (1994).
Pragmatics Quality	Relates the artifact to the interpretation fo the audience.	Lindland *et al.* (1994).

Tabela A.6 – Exemplificação de dados para a metaclasse *QualityGoal*.

Quality Goal			
Name	**Description**	**Quality Type**	**Source**
Functionality	The capability of the software to provide functions which meet the stated and implied needs of users under specified conditions of usage.	External Quality	ISO/IEC 9126
Reliability	The capability of the software product to maintain its level of performance under stated conditions for a stated period of time.	External Quality	ISO/IEC 9126
Usability	The capability of the software product to be understood, learned, used and provide visual appeal, under specified conditions of usage.	External Quality	ISO/IEC 9126
Efficiency	The capability of the software product to provide desired performance, relative to the amount of resources used, under stated conditions.	External Quality	ISO/IEC 9126

Tabela A.7 – Exemplificação de dados para a metaclasse *QualityGoal* (*Subgoals*).

Quality (Sub) Goal		
Name	**Description**	**Quality Goal**
Suitability	The suitability sub-characteristic allows drawing conclusions about how suitable software is for a particular purpose.	Functionality
Maturity	The capability of the software product to maintain its level of performance under stated conditions for a stated period of time.	Reliability
Learnability	The capability of the software product to be understood, learned, used and provide visual appeal, under specified conditions of usage.	Usability
Resource Utilization	The capability of the software product to provide desired performance, relative to the amount of resources used, under stated conditions.	Efficiency

Tabela A.8 – Exemplificação de dados para a metaclasse *EvaluationMethod*.

Evaluation Method			
Name	**Description**	**Reference**	**Level**
Inspection	It refers to review of any work product by trained individuals who look for defects using a well-defined process.		A – low
Simulation	It is based on the process of modeling a real phenomenon with a set of mathematical formulas. It is, essentially, a program that allows the user to observe an operation through simulation without actually performing that operation.	Conceptual Models	D – very high
Semantic check	It can be used to check if an artifact is adhering to its requirements. The check can be made by experts or specialized tools.	Conceptual Models	C – high
Code review	It can be done as a special kind of inspection in which the team examines a sample of code and fixes any defects in it. It can be by an inspector or by tools.		B – medium

Tabela A.9 – Exemplificação de dados para a metaclasse *Practice*.

Practice	
Name	**Description**
Syntax checking	It is a way to verify if an artifact conforms to its syntax.
Consistency checking	Check if the artifact has consistency.
Statement insertion	Include a statement to solving a conflict.
Statement deletion	Remove a statement to solving a conflict.

Tabela A.10 – Exemplificação de dados para a metaclasse *Metric.*

Metric			
Name	**Acronym**	**Description**	**Unit**
Number of Local Methods	NOM	It measures the number of methods locally declared in a class. Inherited methods are not considered. It is the size of the interface of a class and allows conclusions on its complexity.	Units
Number of Defects	ND	It is introduced by programmers/hour. This metric depends of the language and the experience of development team..	Nominal
Help Frequency	HF	Number of an user (or a set of users) call for help.	Access / Month
Number of Associations	NA	Total number of associations in a model.	Units
Number of Dependencies	ND	This metrics is used to calculate the total number of dependency relationships within the class diagram.	Units
Number of Aggregations	NA	It calculates the number of aggregation relationships within a class diagram.	Units

ANEXO B – DIAGRAMA DE DADOS

Este anexo apresenta o diagrama de dados que compõe a base de dados construída a partir do metamodelo descrito no Capítulo 3 e que suporta a ferramenta de apoio desenvolvida como parte deste trabalho.

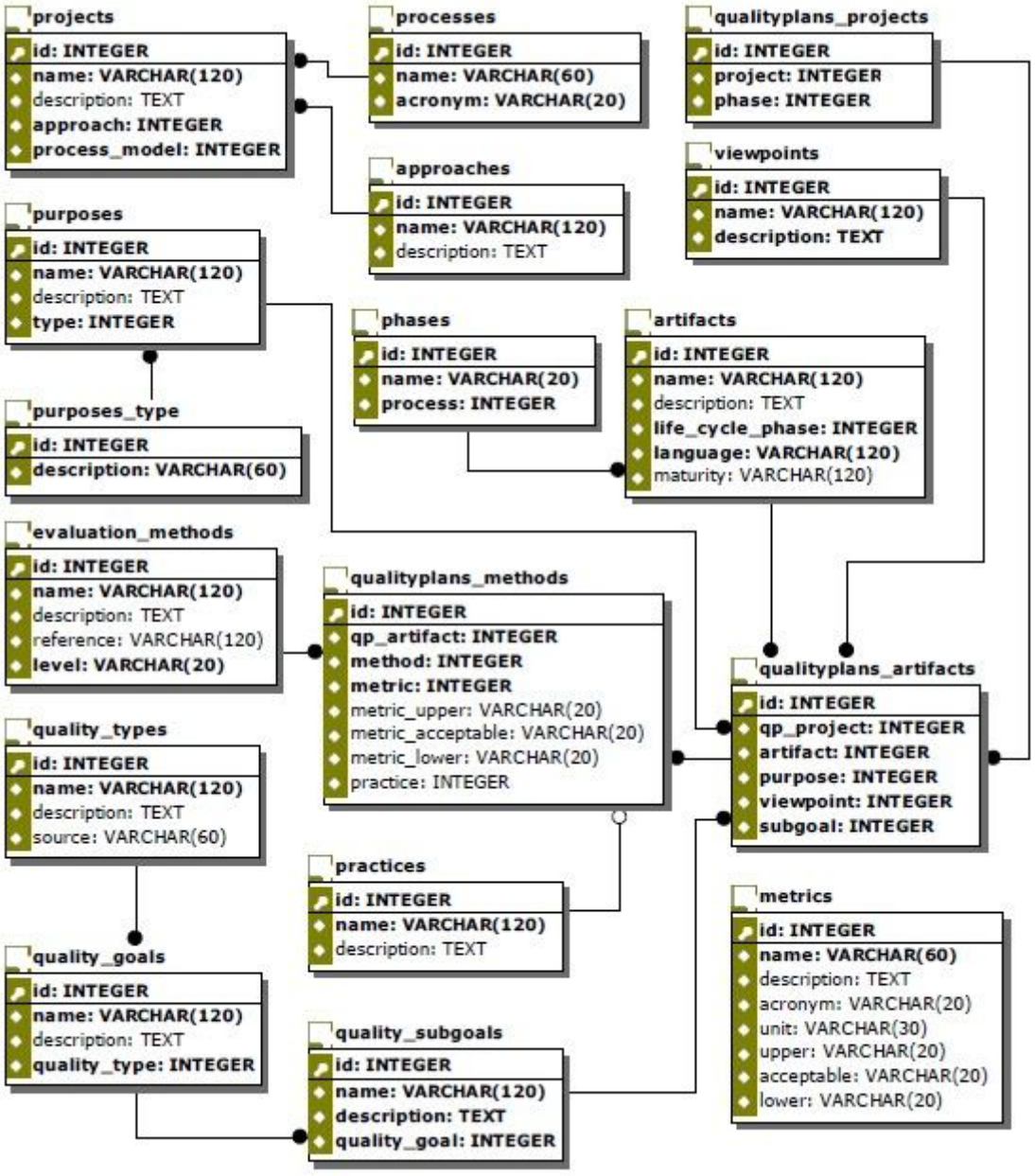

Figura B-1 – Diagrama de dados da ferramenta desenvolvida para avaliação de artefatos de software.