Avaliação da qualidade de modelos em um processo MDE

Gelson Bertuol, Wagner Gadêa Lorenz, Lisandra Manzoni Fontoura

Departamento de Eletrônica e Computação – Univ. Federal de Santa Maria (UFSM).
Prédio Anexo B/CT, 3º andar – Sala 378 – 97.105-900 Santa Maria – RS – Brasil

{gelson.bertuol, wagnerglorenz, lisandramf}@gmail.com

Abstract. *Model-Driven Engineering (MDE) is an approach whose models assume, instead of the code, the role main artifacts in software development process. This paradigm shift creates a need for tailoring practices for assessment of models quality within the processes. This paper proposes an approach for assessment the quality of models in MDE processes which is able to lead the designers in the search for quality goals considering the purpose of the models within the processes lifecycle. The quality is defined and measured from a tailored quality framework to the needs MDE by this work. To validate this proposal, a MDE process is tailored.*

Resumo. *Model-Driven Engineering (MDE) é uma abordagem cujos modelos assumem, no lugar do código, o papel de principais artefatos no processo de software. Esta alteração de paradigma cria a necessidade da adaptação de práticas de avaliação da qualidade de modelos dentro dos processos. Este artigo propõe uma abordagem para avaliação da qualidade de modelos em processos MDE capaz de guiar os projetistas na busca por metas de qualidade considerando o propósito dos modelos dentro do ciclo de vida dos processos. A qualidade é definida e avaliada a partir de um framework de qualidade adaptado às necessidades MDE por este trabalho. Para validar a proposta é realizada uma adaptação em um processo MDE.*

1.Introdução

Model-Driven Engineering (MDE) é uma abordagem que enfatiza o uso de modelos como os principais artefatos em um processo de desenvolvimento de softwares, graças à separação clara entre modelos que descrevem o domínio do negócio e modelos que carregam atributos específicos das plataformas alvo [Schmidt 2006]. O ponto chave dessa abordagem são as transformações (semi) automáticas entre esses modelos que prometem, entre outras vantagens, melhorar a interoperabilidade, portabilidade, reduzir custos e o tempo de desenvolvimento.

Consequentemente, por ser uma abordagem voltada à manipulação de modelos, a avaliação da qualidade ganha importância adicional em relação aos processos centrados no código. Ou seja, a avaliação da qualidade deve ter como foco principal os modelos. Isso acontece porque em MDE as transformações ocorrem necessariamente sobre os modelos, sendo que a geração de código, quando executada, ocorre somente no final do processo. Outro ponto importante é o fato de que os modelos que descrevem o domínio da aplicação tendem a serem reusados em várias transformações e a sua validação ajuda a aumentar a confiabilidade sobre os artefatos gerados. Portanto, a qualidade dos

modelos tem consequência direta na qualidade do produto final [Mohagheghi e Dehlen 2008].

Paralelamente, por serem os primeiros artefatos criados, outras vantagens da avaliação de modelos, independente da abordagem de desenvolvimento usada, caracterizam-se pelo fato de que defeitos tendem a ter um custo, financeiro e de tempo, exponencialmente menor quanto mais cedo forem detectados e corrigidos, além de proporcionar aos *stakeholders* maior segurança e compreensão sobre o comportamento semântico do sistema.

Entretanto, o conceito de qualidade pode assumir uma perspectiva subjetiva e dependente da visão de quem os manipula, dificultando uma avaliação quantitativa. Assim, uma das formas de se definir e alcançar a qualidade conceitual dos modelos é por meio de *frameworks* de qualidade. Estes são usados para definir características de qualidades inerentes a cada tipo de modelo e, em seguida, relacioná-las às metas de qualidade desejadas [Wagner e Deissenboeck 2007, Mohagheghi e Dehlen 2008]. Esses *frameworks* não são exclusivos para processos MDE, mas sim para guiar a busca por metas de qualidade de modelos em geral. Porém, como dito anteriormente, a abordagem MDE tem seu foco no desenvolvimento dirigido por modelos, então esses *frameworks* tornam-se ferramentas importantes para que projetistas consigam estabelecer práticas de avaliação da qualidade que resultarão em um produto de software mais confiável, correto e de acordo com as expectativas dos usuários.

Este trabalho propõe o uso de um *framework* de qualidade, instanciado a partir de um metamodelo proposto por Mohagheghi e Dehlen (2008), para definir as características, práticas e métodos de avaliação de modelos usados no desenvolvimento de softwares com atividades ajustadas para MDE. Este *framework* objetiva guiar projetistas e desenvolvedores na escolha de como avaliar um determinado modelo considerando o seu propósito, a meta de qualidade pretendida e a fase do ciclo de vida do desenvolvimento em que o modelo está inserido. Assim, espera-se criar uma estrutura que sirva como guia para processos MDE, e consequentemente produtos de software, ajustados aos fatores de qualidade desejados.

O restante desse artigo está organizado da seguinte forma. Na Seção 2 são descritos os conceitos sobre modelos e modelos de qualidade relacionados a este trabalho. Na Seção 3 são abordados alguns trabalhos relacionados a *frameworks* de qualidade referenciados na literatura e que possuem relevância para essa pesquisa. Na Seção 4 é descrito o processo adaptado para MDE usado para exemplificar a proposta. Na Seção 5 é proposta a integração do metamodelo de qualidade com o processo de desenvolvimento MDE. Por fim, na Seção 6 são descritas as considerações finais sobre o trabalho.

2. Modelos de Qualidade

O conceito de qualidade é naturalmente vago, pois depende basicamente da interpretação humana. Dessa forma, muitos trabalhos têm sido propostos na literatura ao longo do tempo para definir, de forma mais precisa, o termo qualidade. Segundo a IEEE, atributos de qualidade em engenharia de software são: i) o grau em que um sistema, componente ou processo está de acordo com os requisitos especificados; ii) o grau em que um sistema, componente ou processo está de acordo com as expectativas e necessidades dos

usuários. Ao mesmo tempo, Claxton e McDougal (2000) definem que a avaliação da qualidade pode ser realizada sob duas óticas: i) medir a coisa certa, da forma certa e com as medidas certas; ii) julgar algo baseado nas suas pretensões e propósitos funcionais.

Com relação aos modelos, Kühne (2006) classifica-os como *descritivos* (capturam algum conhecimento, requisitos ou análise de domínio) ou *prescritivos* (usados como estruturas para guiar a implementação de um sistema). Ou seja, modelos podem ser usados para descrever o domínio de uma aplicação e/ou descrever o software desse domínio. Paralelamente, diferentes tipos de modelos são usados para especificar visões diferentes de um mesmo sistema. Por exemplo, a UML, que é considerada atualmente como um padrão *de-facto* pela indústria para modelar softwares orientados a objetos, possui diagramas especializados para diferentes visões [Usman *et al.* 2008]. Entre os mais utilizados, pode-se destacar o diagrama de caso de uso para capturar e validar os requisitos, o diagrama de classes para projetar e guiar a estrutura e os diagramas de máquina de estados e de atividades para projetar a visão comportamental do sistema em desenvolvimento. Portanto, diferentes tipos de modelos possuem diferentes propósitos e, consequentemente, as visões de qualidade devem considerar a relação entre o objetivo do modelo e a meta de qualidade a ser alcançada. Essa relação pode ser estabelecida mediante práticas conhecidas, como por exemplo, para um modelo de caso de uso, cuja qualidade desejada é melhorar a comunicação com o cliente, pode-se aumentar a abstração do domínio e/ou envolver especialistas do domínio na construção do modelo [Mohagheghi e Dehlen 2008].

Na abordagem *Model-Driven*, os modelos assumem uma estrutura mais formal e são divididos em três visões diferentes. A primeira delas é a *Computation Independent Model* (CIM) em que os modelos possuem uma visão independente da computação, descrevendo apenas os aspectos lógicos do domínio. A visão seguinte é a *Platform Independent Model* (PIM) em que os modelos são definidos em um alto grau de abstração, independentes de qualquer tecnologia ou arquitetura. Por fim, os modelos *Platform Specific Models* (PSM) são descritos com toda a completude e detalhes específicos da plataforma a que aderem. Essa terminologia é comumente usada na abordagem *Model-Driven Arquitecture* (MDA) [OMG 2001] que descreve a abordagem MDE em um nível conceitual mais concreto, ou seja, implementa as transformações entre modelos e a geração de código.

Na busca por mecanismos para definir e avaliar o conceito de qualidade, muitos trabalhos têm sido realizados. Alguns deles abordam modelos hierarquicamente estruturados que focam na descrição de conjuntos de características/subcaracterísticas e a relação entre eles para especificar e avaliar os atributos de qualidade. Essa abordagem é chamada de *fixed-model* [Trendowicz e Punter 2003] e tenta identificar características de qualidade universais que possam ser usadas em processos de desenvolvimento distintos. Contudo, essa abordagem tende a ser rígida, permitindo poucas adaptações entre os processos. Além disso, há uma falta de transparência por impor a arquitetura do modelo sem fornecer a lógica por trás e sem descrever como as características de alto nível são decompostas para as subcaracterísticas e métricas [Deissenboeck *et al.* 2009]. Alguns trabalhos que caracterizam essa abordagem são: Cavano e McCall (1978), Boehm *et al.* (1978), Dromey (1995) e o padrão ISO/IEC 9126 (2001).

Em contrapartida, na abordagem *define-your-own-model* não há um conjunto de características de qualidade pré-definido, mas sim, um modelo de qualidade que, com a cooperação dos usuários, guia a busca por metas de qualidade que se deseja alcançar. Essa é a abordagem adotada para guiar este trabalho, visto que ela oferece a flexibilidade necessária para que o *framework* de qualidade seja aderente às necessidades de cada fase do ciclo de vida do processo, além de considerar as necessidades e desejos dos projetistas para as metas de qualidade dos modelos sob propósitos específicos.

Contudo, para evitar a dispersão de conceitos e abordagens de qualidade e a criação excessiva de *frameworks* extremamente específicos a um único contexto e sem relação entre eles, este trabalho adere à ideia de Wagner e Deissenboeck (2007) que propõem que modelos de qualidade sejam criados a partir de metamodelos genéricos. Os metamodelos de qualidade permitem capturar e descrever características de qualidade que são comuns a múltiplos contextos, mas extensíveis, por intermédio dos *frameworks*, a propósitos específicos. Além disso: i) fornecem diferentes visões de qualidade, ii) habilitam a busca por metas de qualidade variadas, iii) fornecem apoio a múltiplas técnicas de garantia da qualidade aplicadas em suas respectivas fases de desenvolvimento, iv) permitem a identificação de inconsistências e redundâncias causadas, ou não visíveis, quando há a aplicação dos métodos isoladamente, e v) facilitam o reuso do contexto de qualidade.

3. Trabalhos Relacionados à Modelagem da Qualidade

Com relação a metas de qualidade, Lindland *et al.* (1994) desenvolveram um dos primeiros trabalhos para identificar as propriedades de qualidade por meio de uma estrutura sistemática capaz de separar metas (*o que* se está tentando alcançar em uma modelagem conceitual) dos meios para se alcançar essas metas (*como* chegar a um modelo de qualidade). Dessa forma, esse *framework*, ilustrado na Figura 1, é separado em três conceitos básicos de qualidade:

- Qualidade Sintática: relaciona o modelo à linguagem de modelagem usada, verificando o quão bem um determinado modelo está de acordo com a sintaxe dessa linguagem. A única meta de qualidade a ser alcançada, segundo esse *framework*, é a correção sintática, por meio de atividades de checagens formais da sintaxe com o auxílio de ferramentas;
- Qualidade Semântica: relaciona o modelo ao domínio, ou seja, verifica o quão bem um determinado modelo corresponde ao domínio que ele descreve. A principal atividade usada para alcançar essa meta é a checagem da consistência entre os modelos que descrevem o mesmo domínio; porém, caso haja alguma inconsistência, como por exemplo, duas regras conflitantes, o *framework* estabelece como práticas a exclusão de uma delas ou a inserção de uma terceira 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).

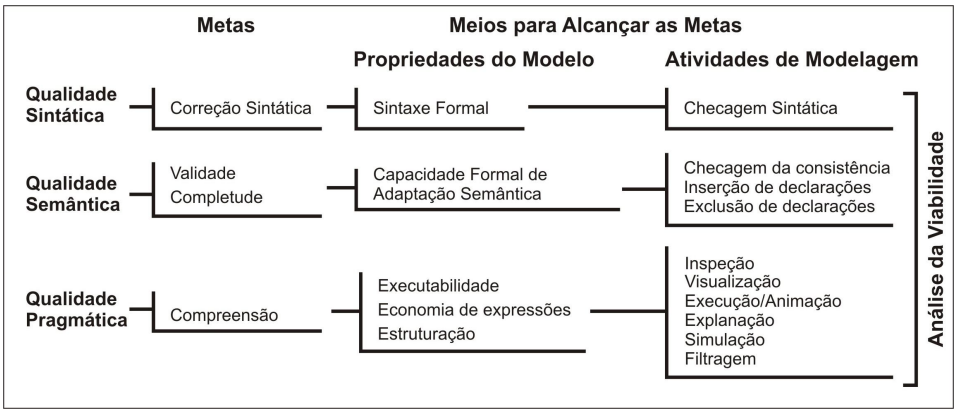

Figura 1. *Framework* adaptado de Lindland *et al.* (1994) para distinguir metas de qualidade e meios para alcança-las.

Porém, conseguir que um único modelo atinja todas as metas de qualidade na sua totalidade é algo extremamente difícil, talvez impossível, tanto que em seu trabalho, Lindland *et al.* (1994) incluíram a notação de *Análise da Viabilidade* que propõe que uma meta de qualidade seja buscada somente enquanto a relação custo/benefício for vantajosa. Essa mesma ideia de viabilidade também foi descrita por Bach em uma série de três artigos [Bach 1995, Bach 1997, Bach 1998] que afirmam que um modelo não precisa ser necessariamente livre de erros, mas deve atingir um grau de confiabilidade que satisfaça às necessidades dos usuários e do sistema. Contudo, essa noção de viabilidade, segundo os autores, depende exclusivamente da interpretação humana.

O *framework* de Lindland *et al.* (1994) foi posteriormente estendido para outros *frameworks*, entre eles o descrito por Trendowicz e Punter (2003) que apesar de ter sido desenvolvido para o paradigma de *Software Product Lines* (SPL), adere à proposta defendida neste artigo por ser um método orientado a metas, capaz de empregar abordagens tanto qualitativas quanto quantitativas para o controle da qualidade. Em seu trabalho, Trendowicz e Punter (2003) definem três requisitos não-funcionais que são imprescindíveis para um *framework* de qualidade:

- Flexibilidade – um *framework* de qualidade deve ser flexível ao contexto de dependência da qualidade, adaptado ao contexto da organização, do projeto e do processo. Organizações possuem projetos diferentes e os próprios projetos possuem diferenças entre si. Como um dos objetivos de um *framework* de qualidade é o reuso, ele deve ser flexível o suficiente para poder ser ajustado a projetos e/ou processos distintos, além de ser flexível aos contextos individuais dos usuários que, por sua vez, possuem visões e níveis de entendimento diferentes para modelos distintos que abordam um mesmo propósito;
- Reusabilidade – para assumir o desenvolvimento de produtos de alta qualidade, um *framework* de qualidade deve ser reusável, ou seja, conforme introduzido no item anterior, o *framework* deve reaproveitar a experiência adquirida no uso de modelos de qualidade em projetos anteriores;

- Transparência – um modelo de qualidade deve fornecer uma análise racional de como as características de qualidade são identificadas, se relacionam e são medidas, de forma clara e sem ambiguidades.

Outro conceito importante descrito no trabalho de Trendowicz e Punter (2003) refere-se ao fato de que um *framework* deve relacionar as metas de qualidade a serem alcançadas às fases do ciclo de vida do processo de desenvolvimento do software em que o modelo está inserido. Assim, quanto mais cedo a avaliação puder ser realizada dentro do processo e, ao mesmo tempo, mais fases essa avaliação puder abranger, mais efetivo será o desempenho do *framework* na avaliação dos modelos.

Santos e Pretz (2010) apresentam um *framework* para viabilização da especificação de modelos e metodologias reconhecidas (RRBT – *Risk & Requirement Based Testing*, GQM – *Goal Question Metric*, e as normas ISO/IEC 9126-1 e ISO/IEC 14598). Esse modelo de qualidade apoia a definição dos atributos de qualidade específicos para o domínio da aplicação, baseado nos riscos e requisitos do produto. O processo de medição da qualidade contempla as métricas e técnicas para mensuração das evidências dos atributos de qualidade. Sua relação com esta pesquisa está no fato de que o *framework* apresentado baseia-se nas normas que relacionam características de qualidade às características do produto, mesmo que a abordagem não atue diretamente sobre modelos.

Paralelamente, Mohagheghi e Dehlen (2008) afirmam que embora exista uma noção bem definida de qualidade sintática, semântica e pragmática, é necessária a definição de metas de qualidade para modelos baseada na experiência dos desenvolvedores, que são os que efetivamente conhecem as propriedades de consistência e correção semântica. Dessa forma, os autores relacionaram seis propriedades básicas para identificar a qualidade dos modelos: correção, completude, consistência, compreensão, limitação e adaptabilidade.

Com base nessas propriedades, Mohagheghi e Dehlen (2008) construíram um metamodelo, exibido na Figura 2, que reúne os artefatos comuns que devem ser considerados para guiar os desenvolvedores na avaliação da qualidade dos modelos. Os principais elementos que compõem esse metamodelo são:

- *Framework* de Qualidade (*QualityFramework*) que refere-se à coleção de entidades e relacionamentos que compõe a abordagem para verificar a qualidade dos modelos;
- Metas de Qualidade (*QualityGoal*) definem quais as metas de qualidade são pretendidas, considerando os anseios de determinados *stakeholders*, tais como usuários, desenvolvedores ou gerentes. Essas metas podem ter submetas organizadas hierarquicamente;
- Propriedades que Agregam Qualidade (*QualityCarryingProperty*) definem quais propriedades de um artefato ou atividades são necessárias para alcançar uma determinada meta de qualidade;
- Alvo (*Target*) é um artefato ou atividade que possui a propriedade requerida para se alcançar uma meta de qualidade;

- Ponto de Vista (*Viewpoint*) indica o papel do usuário interessado na meta de qualidade que será avaliada;
- Propósito (*Purpose*) descreve qual o propósito do *stakeholder* em atingir aquela meta de qualidade;
- Prática (*Practice*) é o meio para se alcançar uma determinada propriedade de qualidade;
- Método de Avaliação (*EvaluationMethod*) define o método, qualitativo ou quantitativo, para avaliar uma determinada propriedade.

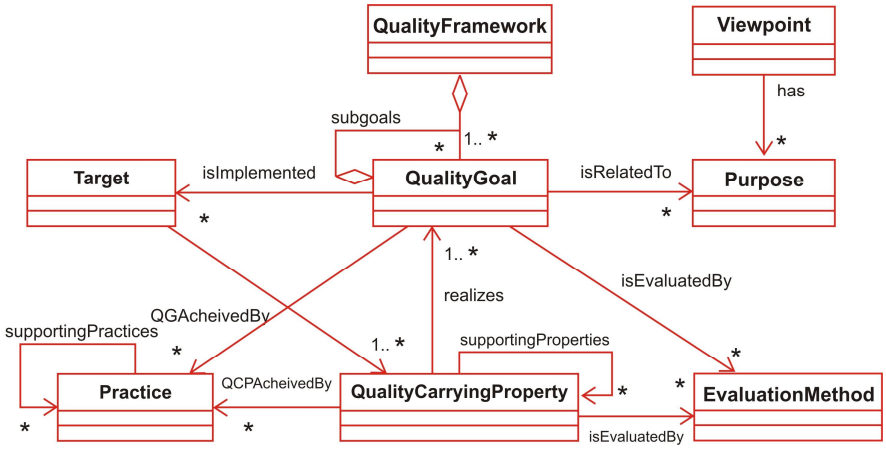

Figura 2. Metamodelo de Qualidade. Extraído de [Mohagheghi e Dehlen 2008]

O metamodelo proposto por Mohagheghi e Dehlen (2008) e os modelos de qualidade descritos nesta seção formam a base para guiar o desenvolvimento deste trabalho na avaliação de modelos inseridos em um processo de desenvolvimento de software adaptado do RUP para conter atividades específicas à abordagem MDE. Esse processo foi proposto por Copetti (2012) e é apresentado na Seção 4.

4. Um Processo Integrado para MDE

Copetti (2012) propôs um *framework* de processo de desenvolvimento de software para o paradigma MDE. A escolha desse trabalho como exemplo para a aplicação da avaliação da qualidade dos modelos proposta aqui ocorreu devido: i) ser baseado no *Rational Unified Process* (RUP), que é um *framework* amplamente conhecido e documentado; ii) por apresentar, de forma clara, as modificações que ocorreram entre os dois processos; iii) por definir, dentro das disciplinas, atividades específicas para verificação e validação de modelos que puderam, assim, servir de base para a avaliação da qualidade de toda a iteração da disciplina; iv) por apresentar uma visão holística do desenvolvimento MDE, considerando tanto os modelos do domínio quanto as transformações.

Sendo MDE uma abordagem guiada essencialmente por modelos, as principais alterações ocorrem nas fases de *elaboração* e *construção* [Copetti 2012]. A fase de *elaboração*, no processo adaptado, deve trabalhar com modelos mais abstratos e independentes da arquitetura. Assim, a construção e o detalhamento da arquitetura são

feitos na fase de *construção*, juntamente com as transformações dos modelos. A implementação deixa de existir, pois nessa abordagem, isso é realizado por ferramentas nas transformações de modelos.

Com relação às disciplinas, as que sofreram alterações mais significativas foram: i) *Análise e Projeto*, que passa a ser o ponto central do desenvolvimento, pois é responsável por construir e gerenciar os modelos, além de criar as arquiteturas mais específicas; ii) *Implementação*, que é substituída pela disciplina de *Gerenciamento de Transformações*; iii) *Testes* que deve englobar agora métodos de avaliação e validação de modelos e não mais de código; e iv) *Ambiente* que envolve a estruturação do processo que integra a transformação com o ciclo de desenvolvimento do software.

Com o objetivo de tornar este artigo mais compreensível e por restrições de espaço, serão exemplificados apenas os modelos do domínio e a disciplina de *Análise e Projeto* do processo. Porém, a ideia é semelhante para as outras disciplinas que são descritas na íntegra por Copetti (2012). A Figura 3 apresenta o diagrama de atividades da disciplina de *Análise e Projeto* do processo de desenvolvimento adaptado para MDE.

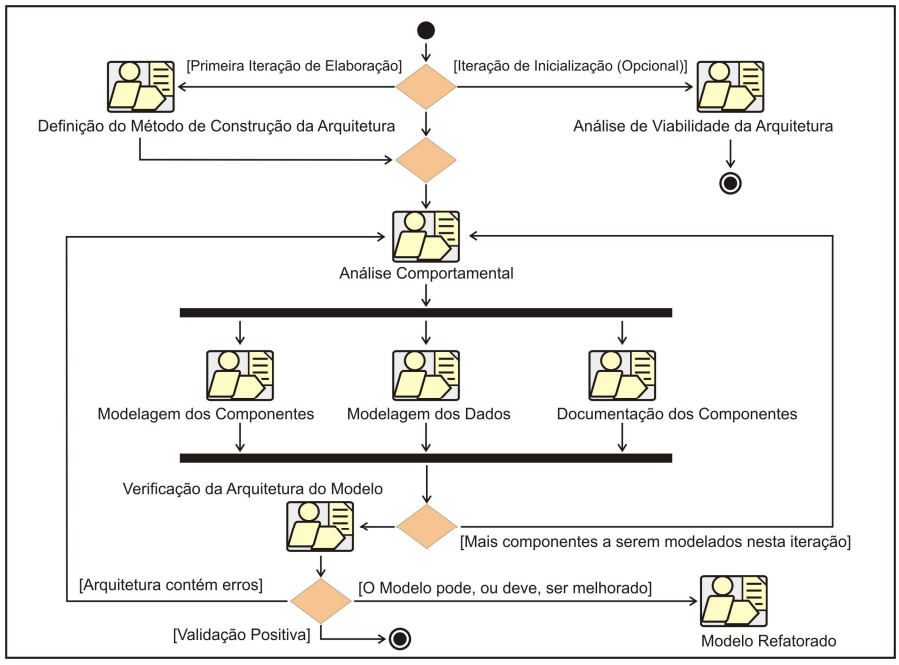

Figura 3. Disciplina de *Análise e Projeto*, adaptado de Copetti (2012)

A disciplina de *Análise e Projeto* foi alterada nesse processo MDE para abranger duas fases do ciclo de desenvolvimento [Copetti 2012]. A primeira é a fase de *concepção* que deve verificar se o sistema é viável. A outra é a fase de *elaboração* que avalia as soluções que podem ser utilizadas, elabora uma arquitetura e inicia as atividades relacionadas ao sistema. A fase de elaboração trabalha apenas com os artefatos de transformação.

A atividade da Análise de Viabilidade da Arquitetura, na fase de concepção, envolve a análise dos requisitos e das necessidades do sistema, construindo um conceito

da arquitetura para solução a ser implementada. Na atividade de Definição do Método de Construção da Arquitetura, os arquitetos analisam os requisitos e a descrição do sistema, decidem a linguagem de modelagem a ser usada, especificam como o sistema será modelado, quais os artefatos serão avaliados e quais os propósitos dos modelos no processo, além de realizarem um planejamento preliminar das iterações que compõe o processo de desenvolvimento da arquitetura.

Em seguida, tem-se a modelagem da iteração pela Análise Comportamental, que verifica os requisitos elucidados e sua aderência ao modelo que irá descrevê-lo. Dessa forma, o principal objetivo dessa atividade é, a cada iteração, garantir que os projetistas e arquitetos tenham a compreensão necessária do comportamento dos modelos e, se esse comportamento está em conformidade com os requisitos e, consequentemente, com o domínio do negócio. A iteração, nesse ponto, se divide em três atividades paralelas:

- Modelagem dos Componentes – atividade em que ocorrem, de fato, a composição e o sequenciamento dos modelos que serão submetidos às transformações, definindo-se as informações necessárias para as transformações tanto entre modelos (chamadas Transformações M2M) como para geração de código compilável (Transformações M2T), quando necessário;
- Modelagem de Dados – atividade responsável pela criação da estrutura relacional do banco de dados, se necessário, e a descrição e especificação dos dados que serão utilizados pelo sistema;
- Documentação dos Componentes – atividade responsável por criar a documentação necessária para cada componente gerado.

A iteração continua até que todos os componentes tenham sido modelados, especificados, construídos e documentados. Em MDE, a arquitetura deve atingir certo grau de robustez antes de seguir com as próximas iterações, visto que, sem correção e completude, as transformações parciais do sistema não poderão acontecer [Mohagheghi e Dehlen 2008].

A próxima atividade corresponde à Verificação da Arquitetura do Modelo. Seu objetivo é realizar a validação do modelo como um todo após todas as iterações estarem completas. Apesar de Copetti (2012) definir uma atividade específica para validação dos modelos gerados, não há uma especificação clara de quais as metas de qualidade que devem ser alcançadas, nem quais deveriam ser as práticas ou métodos que melhor se adaptam ao modelo segundo seu propósito. Assim, baseado nessas deficiências, a Seção 5 define como as atividades de verificação e validação de modelos devem ser estruturadas para permitir a análise da qualidade de modelos segundo os *frameworks* descritos na Seção 3.

5. Avaliação da qualidade no processo baseado em MDE

Tecnicamente, em um processo MDE, são duas as visões mais importantes dos modelos: a visão dos usuários (especialmente dos desenvolvedores e projetistas) que esperam que os modelos sejam livres de erros, porém, sobretudo compreensíveis; e a visão das ferramentas de transformação que precisam que tanto os modelos do domínio quanto os mecanismos de transformação – que nesta abordagem também são considerados como modelos [Schmidt 2006] – sejam válidos, completos e consistentes. Ou seja, tenham as

qualidades pragmáticas, semânticas e sintáticas, tal como descreve o *framework* de Lindland *et al.* (1994) discutido na Seção 3.

Dessa forma, esta seção descreve, como um estudo de caso, a estrutura que pode ser usada para alcançar as metas de qualidade almejadas para os modelos seguindo o processo MDE apresentado na Seção 4. Essa estrutura serve para guiar os projetistas na definição de quais são essas metas e como elas devem ser buscadas para cada modelo com base em quatro fatores: i) o propósito do modelo; ii) as características do modelo; iii) a visão do usuário interessado no propósito do modelo; e iv) a fase do ciclo de vida do processo em que o modelo está inserido.

Como exemplo, tem-se a atividade Análise Comportamental – na fase de *concepção* do ciclo de vida do processo. Para esta atividade, uma das metas de qualidade é a *melhoria da comunicação* entre os envolvidos no projeto (projetistas, arquitetos e clientes), em que os propósitos são a compreensão dos requisitos e a verificação destes requisitos em relação ao modelo desenvolvido. Sendo assim, valendo-se da abordagem proposta, é possível avaliar as propriedades dos modelos mais relevantes à meta de qualidade desejada, que neste caso podem ser a economia de expressões, a estruturação do modelo e a abstração do problema, conforme sugere o *framework* proposto por Lindland *et al.* (1994). As práticas necessárias para alcançar a meta de qualidade neste caso são: o uso de uma linguagem de modelagem padrão (UML) e a ocultação dos detalhes do modelo [Alves *et al.* 2011]. Considerando a fase do desenvolvimento, e por ser um processo voltado à abordagem *Model-Driven*, a sugestão é que a avaliação seja realizada sobre o modelo CIM.

Paralelamente, na atividade Modelagem dos Componentes, os modelos podem assumir outras metas de qualidade. Esta atividade trabalha com modelos mais completos e coesos (modelos independentes de plataforma – PIM) a fim de compor o seu sequenciamento e realizar as transformações. Assim, uma das metas avaliadas é a *correção sintática*, cujos propósitos são a adequação das declarações ao domínio e as transformações entre modelos (M2M) na visão das ferramentas de transformação. As propriedades que levam a esta meta são correção e completude mediante práticas como: uso de DSLs *(Domain Specific Language)*, sintaxe formal e detecção e correção de erros [Mohagheghi e Dehlen 2008], por exemplo. A Figura 4 ilustra uma possível composição do *framework* proposto por este trabalho com as metas de qualidade das atividades descritas acima. Do lado esquerdo (a) tem-se a estrutura para a atividade Análise Comportamental, enquanto do lado direito (b), para a atividade Modelagem de Componentes.

Em seu processo, Copetti (2012) especificou uma atividade exclusiva para a verificação dos modelos depois que todas as iterações tivessem completas (Verificação da Arquitetura do Modelo). A abordagem proposta aqui defende a definição de metas de qualidade distribuídas por todas as atividades da disciplina. Contudo, uma atividade exclusiva para verificação é importante, pois segundo o diagrama de atividades do processo adaptado, ela é responsável por validar a semântica e a aderência ao domínio de todos os modelos já integrados. Assim, as metas de qualidade que se propõe para essa atividade são: a *correção sintática*, pois o modelo deve estar completo e aderente à linguagem de modelagem escolhida; e a *correção semântica* que deve avaliar se o

modelo atingiu o objetivo de capturar todas as perspectivas do domínio corretamente, sem declarações desnecessárias, ambíguas ou contraditórias.

atividade: Análise Comportamental	
QualityGoal	Melhoria da Comunicação
ViewPoint	Projetistas, Arquitetos e Clientes
Purpose	Compreensão e Validação dos Requisitos Aderência dos Requisitos ao Modelo
Target	*Computation Independent Model* (CIM)
QualityCarryingProperty	Economia de Expressões Estruturação do Modelo Abstração do Problema
Practice	Uso de Linguagem Padrão Ocultação de Detalhes
EvaluationMethod	Inspeção por Especialistas do Domínio Simulação Execução/Animação

(a)

atividade: Modelagem dos Componentes	
QualityGoal	Correção Sintática
ViewPoint	Ferramenta de Transformação
Purpose	Adequação das Declarações Transformações M2M
Target	*Platform Independent Model* (PIM)
QualityCarryingProperty	Correção Completude
Practice	Uso de Linguagens Específicas do Domínio Sintaxe Formal Detecção e Correção de Erros
EvaluationMethod	Checagem Semântica Checagem Sintática

(b)

Figura 4. Composição da estrutura do *framework* para duas atividades do processo MDE.

Segundo a abordagem de Lindland *et al.* (1994), a meta de correção sintática deve ser alcançada mediante a aderência do modelo à formalidade da linguagem usando práticas de sintaxe formal por meio de checagem, geralmente com o uso de ferramentas. Ou seja, esta meta de qualidade, sob o ponto de vista das ferramentas de transformação, tem o propósito de refinamento dos modelos (transformações M2M) e geração de código compilável (transformações M2T), sendo alcançada por meio de práticas como sintaxe formal, detecção e correção de erros.

Entretanto, a meta de correção semântica precisa da interferência humana para sua interpretação e as propriedades a serem alcançadas são a *correção* e a *completude*. Desta forma, a correção semântica tem os propósitos de comunicação e implementação do ponto de vista de clientes e desenvolvedores, respectivamente, e as propriedades dos modelos podem ser avaliadas por práticas como semântica formal e restrições para a propriedade de correção do modelo.

A flexibilidade proporcionada pelo metamodelo permite que a estrutura usada para alcançar uma meta de qualidade possa ser reusada em outros modelos com os mesmos propósitos e visões, ao mesmo tempo em que permite especificar práticas e métodos de avaliação diferentes para o mesmo modelo, porém, com outras visões e propósitos, levando a uma arquitetura coesa, consistente, completa e sem ambiguidades.

Vale ressaltar que as práticas e os métodos de avaliação dependem das características e dos propósitos dos modelos, contudo, são independentes entre si e das características do projeto como um todo, podendo ser aplicados a qualquer padrão de processo ajustado para MDE. A ideia de se incluir um mecanismo de avaliação da qualidade para modelos no processo de desenvolvimento é que cada atividade tenha suas metas de qualidade associada como atributos de entrada. Essas metas devem ser definidas, pelos projetistas, considerando a opinião dos usuários finais como sugerem Trendowicz e Punter (2003), e levando em consideração o propósito da atividade e a visão de quem irá executá-la.

Os outros *constructos* do metamodelo de Mohagheghi e Dehlen (2008) podem ser alcançados por meio de práticas e métodos de avaliação já consolidados na indústria e na literatura, ou de uma possível base de conhecimento mantida pela organização. Algumas dessas práticas e métodos são descritos por Usman *et al.* (2008) para a avaliação da consistência entre modelo, ou Alves *et al.* (2011) que sugerem práticas formais para validação de requisitos em sistemas confiáveis.

A Figura 5 ilustra uma visão geral das relações entre os atributos e as metas de qualidade no processo de avaliação dos modelos para as duas metas de qualidade descritas anteriormente, porém, serve como uma visão geral da proposta apresentada.

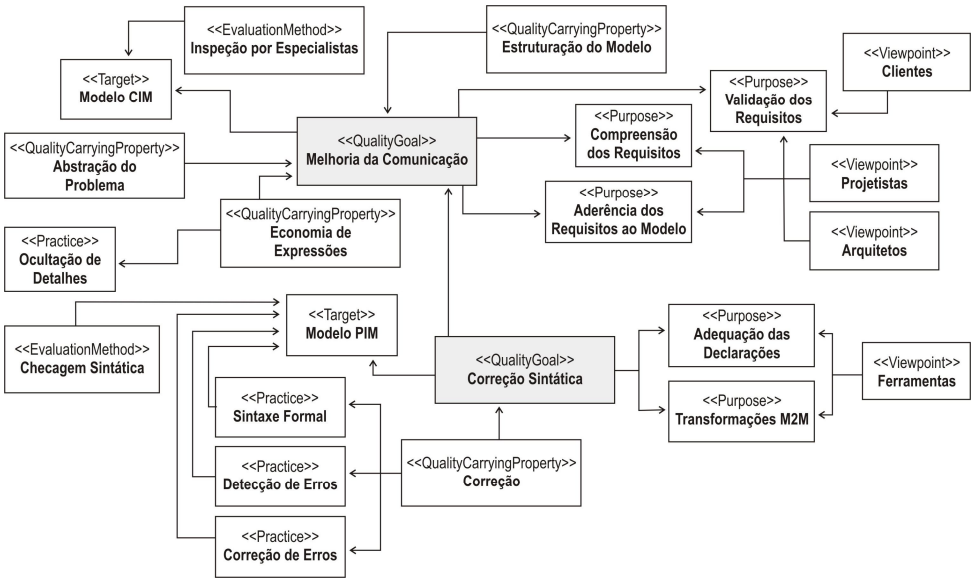

Figura 5. Exemplo da composição estrutural para avaliação de Metas de Qualidade.

Como forma de relacionar a abordagem proposta neste artigo àquelas pesquisadas para composição deste trabalho, a Tabela 1 apresenta uma comparação destacando os principais fatores e contribuições de cada um. As contribuições comparadas entre as abordagens foram:

1. Quais as principais contribuições do trabalho?
2. Quais são as visões de qualidades adotadas pela proposta?
3. O *framework* suporta a avaliação de modelos e/ou a abordagem MDE?
4. O *framework* é instanciado a partir de um metamodelo?
5. O *framework* é flexível para que os *stakeholders* definam as metas de qualidade desejadas?
6. O *framework* considera a fase do ciclo de vida do processo para a avaliação das metas de qualidade?
7. O *framework* suporta as análises quantitativas e/ou qualitativas da qualidade?

Tabela 1. Comparação entre os trabalhos relacionados

Fatores	Lindland *et al.* (1994)	Trendowicz e Punter (2003)	Mohagheghi e Dehlen (2008)	Santos e Pretz (2010)	Abordagem Proposta
Principais Contribuições	Separar as metas de qualidade dos meios para alcança-las.	Definir requisitos não-funcionais necessários à avaliação da qualidade.	Definir um metamodelo para estruturar aspectos de qualidade para MDE.	Relacionar a qualidade a normas técnicas conhecidas.	Definir a avaliação da qualidade para modelos em um processo MDE.
Visões de qualidade adotada.	Qualidade Sintática Qualidade Semântica Qualidade Pragmática	Definida pela medição das características do produto por um processo matemático (BBN *Bayesian Belief Net*).	Aquelas definidas pelos os interessados no modelo (projetistas, desenvolvedores, clientes).	A visão de qualidade é aderente a normas e padrões (RRBT, ISO 9126, ISO 14598).	Aquelas definidas pelos os interessados no modelo (projetistas, desenvolvedores, clientes)
Framework suporta a avaliação de modelos e/ou a abordagem MDE.	Considera modelos e produtos, porém, não é aderente a MDE.	Não. Desenvolvido para Linha de Produtos de Software.	Desenvolvido para avaliação de modelos considerando a abordagem MDE.	Não. Desenvolvido somente para modelos de processos.	Desenvolvido para avaliação de modelos considerando a abordagem MDE.
O *framework* é instanciado por um metamodelo.	Não	Não	Sim	Não	Sim
Flexível à definição de qualidade dos *stakeholders.*	Sim	Não	Sim	Não	Sim
Considera a fase do ciclo de vida do processo na avaliação.	Não	Sim	Não	Não	Sim
Tipo de Análise	Quantitativa e Qualitativa	Somente Quantitativa	Quantitativa e Qualitativa	Somente Qualitativa	Quantitativa e Qualitativa

Portanto, percebe-se que a abordagem proposta por este artigo se baseou nas principais contribuições dos trabalhos pesquisados. Entre elas, a distinção entre metas de qualidade e meios para alcança-las [Lindland *et al.* 1994]; o relacionamento da qualidade pretendia à fase do ciclo de vida do processo [Trendowicz e Punter 2003]; o uso de um metamodelo para guiar a construção do *framework* de qualidade [Mohagheghi e Dehlen 2008]; e a aderência das práticas de avaliação dos modelos com padrões e normas técnicas, tais como a ISO 9126 [Santos e Pretz 2010]. A avaliação da qualidade de modelos é relativamente menos comum na literatura e na prática que a avaliação de processos e produtos, especialmente em modelos usados em processos de software guiados pela abordagem MDE o que justifica a proposta apresentada.

8. Conclusão

Este trabalho apresentou uma abordagem para a avaliação da qualidade dos modelos usados em um processo de desenvolvimento adaptado ao conceito MDE. A abordagem estabelece que os modelos sejam avaliados por um conjunto de fatores característicos que incluem: o propósito do modelo, a fase do ciclo de vida do processo em que o modelo está inserido e a visão (ou anseio) de quem almeja a avaliação (projetistas, desenvolvedores, ou mesmo uma ferramenta de transformação de modelos). Estabelecida a meta de qualidade, o *framework* visa guiar o interessado na escolha das práticas e métodos de avaliação mais apropriados ao propósito, sendo esses métodos e práticas extraídos da literatura ou de uma base de conhecimento mantida pela organização.

A base teórica do trabalho foi extraída de uma seleção de trabalhos, descritos na literatura, que apresentam conceitos relacionados tanto à avaliação da qualidade de produtos como de processos, desde que apresentem uma estrutura lógica e conceitos claros para essa avaliação. Assim, usou-se o metamodelo proposto por Mohagheghi e Dehlen (2008) como base para estabelecer a relação entre os conceitos de qualidade apresentados.

Apesar de o foco ter sido um processo MDE, devido à importância dos modelos para essa abordagem, a estrutura proposta por este artigo pode ser usada para qualquer tipo de desenvolvimento, visto que a modelagem é uma etapa comum aos processos de desenvolvimento de softwares. Ao mesmo tempo, a detecção de erros e/ou inconsistências, semânticas ou sintáticas, já nas fases iniciais da modelagem, tende a tornar o desenvolvimento do produto de software mais rápido, barato e com qualidade agregada.

Como trabalhos futuros, pretende-se relacionar o metamodelo de qualidade apresentado às metas e práticas de qualidade relacionadas a atividades do processo de desenvolvimento e manutenção extraídos de um modelo de melhoria da qualidade e aplicados a produtos e serviços, além de realizar uma avaliação prática para a extração de dados empíricos em um processo MDE em uso.

Agradecimentos

Agradece-se a CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pelo suporte financeiro a este trabalho.

Referências

Alves, M. C. B., Drusinsky D. e Shing M. T. (2011) “A Practical Formal Approach for Requirements Validation and Verification of Dependable System”, In: *Fifth Latin-American Symposium on Dependable Computing Workshops*, pages 47-51. IEEE Computer Society.

Bach, J. (1995) “The Challenger of Good Enough Software”, In: *American ProgrammerMagazine*, http://www.satisfice.com/articles/gooden2.pdf, March, 2013.

Bach, J. (1997) “Framework Good Enough Quality”, In: IEEE Computer Society.

Bach, J. (1998) “A Framework for Good Enough Testing”, In: IEEE Computer Society.

Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., MacLeod, G. J. e Merritt, M. J. (1978) "Characteristics of Software Quality", In: *North Holland Publishing Company*.

Cavano, J. P. e McCall, J. A. (1978) "A Framework for the Measurement of Software Quality", In: *Proceedings of the software quality assurance workshop on Functional and performance issues*, pages 133-139.

Claxton, J. e McDougall, P. A. (2000) "Measuring the Quality of Models", In: *The Data Administration Newsletter (TDAN.com)*, http://www.tdan.com/view-articles/4877/, December, 2012.

Copetti, M. A. (2012) "Um Processo Integrado para Qualidade em *Model-Driven Engineering*", In: *Dissertação de Mestrado*, Universidade Federal de Santa Maria (UFSM), Santa Maria, RS, Brasil.

Deissenboeck, F., Juergens, E., Lochmann, K. e Wagner, S. (2009) "Software Quality Models: Purposes, Usage Scenarios and Requirements", In: *Workshop on Software Quality, WOSQ'09. ICSE Workshop on*, pages 9-14.

Dromey, R. G. (1995) "A Model for Software Product Quality", In: *IEEE Transaction on Software Engineering*, 21, pages 146-162.

Lindland, O. I., Sindre, G. e Solvberg, A. (1994) "Understanding Quality in Conceptual Modeling", In: *IEEE Software*, pages 42-49.

ISO (2001) "International Organization for Standardization: ISO/IEC 9126-1:2001, Software Engineering – Product Quality, Part 1: Quality model".

Kühne, T. (2006) "Matters of (Meta-) Modeling", In: *Journal on Software and System Modeling*, Vol. 5, Number 4, pages 396-385.

Mohagheghi, P. e Dehlen, V. (2008) "A Metamodel for Specifying Quality Framework in Model-Driven Engineering", In: *Proceedings of the Nordic Workshop on Model Driven Engineering, Engineering Research Institute, University of Iceland*.

OMG (2001) "MDA Guide – Object Management Group", In: http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf March, 2013.

Santos, L. B. e Pretz, E. (2010) "*Framework* para Especialização de Modelos de Qualidade de Produtos de Software", In: IX Simpósio Brasileiro de Qualidade de Software, pages 57-71.

Schmidt, D. C. (2006) "Model-Driven Engineering", In: IEEE Computer, (Vol. 39, No 2), pages 25-31.

Trendowicz, A. e Punter, T. (2003) "Quality Modelling for Software Product Lines", In: *7th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineeging – QAOOSE'03*.

Usman, M., Nadeem, A., Kim, T. e Cho, Eun-Suk (2008) "A Survey of Consistency Checking Techniques for UML Models", In: *Advanced Software Engineering & Its Applications*, pages 57-62.

Wagner, S. e Deissenboeck, F. (2007) "An Integrated Approach to Quality Modeling", In: Fifth International Workshop on Software Quality – ICSE'07.