MoPE – Modelo de Programação baseado em Exemplos

Uedson S. Reis[1]

uedsonreis@gmail.com

Marcelo A. Moret[1]

mamoret@gmail.com

Eduardo M. F. Jorge[1 2]

emjorge1974@gmail.com

Hernane B. de Barros Pereira[1]

hbbpereira@gmail.com

Programa de Pós-graduação em Modelagem Computacional e Tecnologia Industrial (Doutorado) – PPG-MCTI

Modelagem de Sistemas Cognitivos

Resumo –

Após a proposta do modelo de programação orientado a objetos, os modelos de programação pouco evoluíram. Este artigo visa contribuir com a discussão sugerindo novas abordagens para um modelo de programação que solucione alguns problemas recorrentes da temática, além de reduzir a abstração necessária ao desenvolvimento de sistemas.

Palavras-Chave:

Paradigmas de Programação, Modelo de Programação, Programação baseada em Exemplos, Modelagem baseada em Cenário.

Introdução

Os paradigmas de programação são conceitos usados por muitos desenvolvedores e influenciam bastante a maturidade da área de Software. Eles surgiram para converter problemas reais em soluções computacionais em uma linguagem de programação. Os principais paradigmas são: Imperativo, Funcional, Lógico e Orientado a Objetos. Esses paradigmas têm suas especificidades, porém alguns dos seus conceitos e estruturas são absorvidos por outros, por exemplo, o paradigma Orientado a Objetos pode ser considerado uma extensão do Imperativo (HURLIMANN, 1998). Após a proposta do modelo orientado a objetos, que teve sua origem em meados dos anos de 1960 no centro Norueguês de Computação, poucas mudanças foram realizadas sobre a temática dos paradigmas de programação. Na atualidade algumas linguagens apresentam características diferenciadas, como a tipagem dinâmica (e.g.: Python, Perl e Ruby) ou a chamada de métodos de forma verbosa (Objetive-C) (Apple Inc., 2012). Outras linguagens foram feitas para serem multiparadigmas, podendo conter chamadas de script ou programação funcional junto com o paradigma orientado a objetos. Estes novos recursos não alteram questões paradigmáticas sobre a filosofia de como desenvolver software. Por exemplo, houve poucas mudanças relacionadas à maior proximidade das estruturas com o mundo real ou melhora dos aspectos de organização semântica de código.

Neste contexto, a legibilidade é um importante critério de avaliação utilizado para definir a facilidade de leitura e entendimento de uma linguagem de programação. Outra característica importante é o tempo de aprendizagem de uma linguagem de programação (SEBESTA, 2010). Plataformas como Visual Studio e Xcode, buscaram reduzir o tempo de desenvolvimento e suavizar a complexidade do processo de desenvolvimento através de editores com ferramentas e recursos de apoio ao desenvolvedor, como auto-completar código, reuso de biblioteca e componentes, engenharia reversa, construção visual de telas, etc. Ferramentas com essas característica são denominadas de Rapid Application Development (RAD) como mencionado o objetivo é tornar a construção de software uma tarefa menos árdua. Apesar dos benefícios das ferramentas RAD, existem problemas históricos gerados nos projeto, tais como: forte acoplamento, fraca organização semântica, geração de código desnecessário, etc.

Para minorar estes problemas, a Engenharia de Software fornece técnicas para garantir ao máximo a coesão de software, objetivando uma manutenção evolutiva e corretiva adequada. Importantes técnicas que potencializaram Programação Orientada a Objetos (POO) vêm sendo incorporadas no processo de desenvolvimento. A Unified Modeling Language (UML),

Padrões de Projeto, Framework e Refatoramento atuam sobre a abstração, nível de acoplamento, modularização, complexidade e reuso.

Dentre os diversos aspectos associados à codificação de software este artigo está centrado na problemática de como evoluir regras de negócio com um menor impacto no código e como atualizar a documentação de projeto automaticamente. Serão exploradas algumas das técnicas atuais como Model Driven Architecture (MDA) nas linguagens de programação atuais e as diferenças em relação a esta proposta de um novo paradigma de programação. O MDA define uma especificação para a construção de modelos. Essa especificação define elementos como o Platform-Independent Model (PIM), onde é feita a modelagem genérica de um sistema, e o Platform-Specific Model (PSM), que é o modelo de uma plataforma específica gerado a partir do PIM (OMG, 2003).

Um modelo que pode ser utilizado como PIM em uma plataforma baseado no MDA é o MOBI (JORGE, 2012). O MOBI é um modelo centrado em cenários e na independência da instância para com uma classe, e voltado para o processo de Modelagem Conceitual.

A proposta desse artigo é uma extensão do MOBI que aborde também a implementação de software. Essa extensão visa à criação de um novo Modelo de Programação baseado em Exemplos (MoPE), juntamente com uma metalinguagem, voltado para minimizar questões supracitadas na Engenharia de Software. A metalinguagem citada deve permitir a escrita de códigos de programação que complementem algumas partes do modelo criado no MoPE, como a escrita de métodos e lógicas de negócio. A ideia é que o código escrito com essa metalinguagem seja convertido em uma linguagem específica de domínio junto com o restante do modelo, que define as classes e suas relações, e seus atributos. Nesta fase de concepção do modelo, questões de otimização e desempenho de código não são alvo da pesquisa.

A arquitetura desse novo modelo de programação será pautada pelo MDA, onde os PIMs serão construídos a partir do MOBI e da metalinguagem, e os PSMs construirão o código gerado, para uma plataforma específica, como a linguagem Java por exemplo.

Este artigo está dividido em quatro seções, a primeira sendo esta introdução. A seção 2 trata de problemas inerentes a programação de software. O MOBI será apresentado na seção 3. Na seção 4 o modelo de programação proposto será especificado e discutido. E na Seção 5 são feitas as considerações finais.

Metodologia

Na Figura 1 temos uma ilustração da sequência de atividades previstas na metodologia.

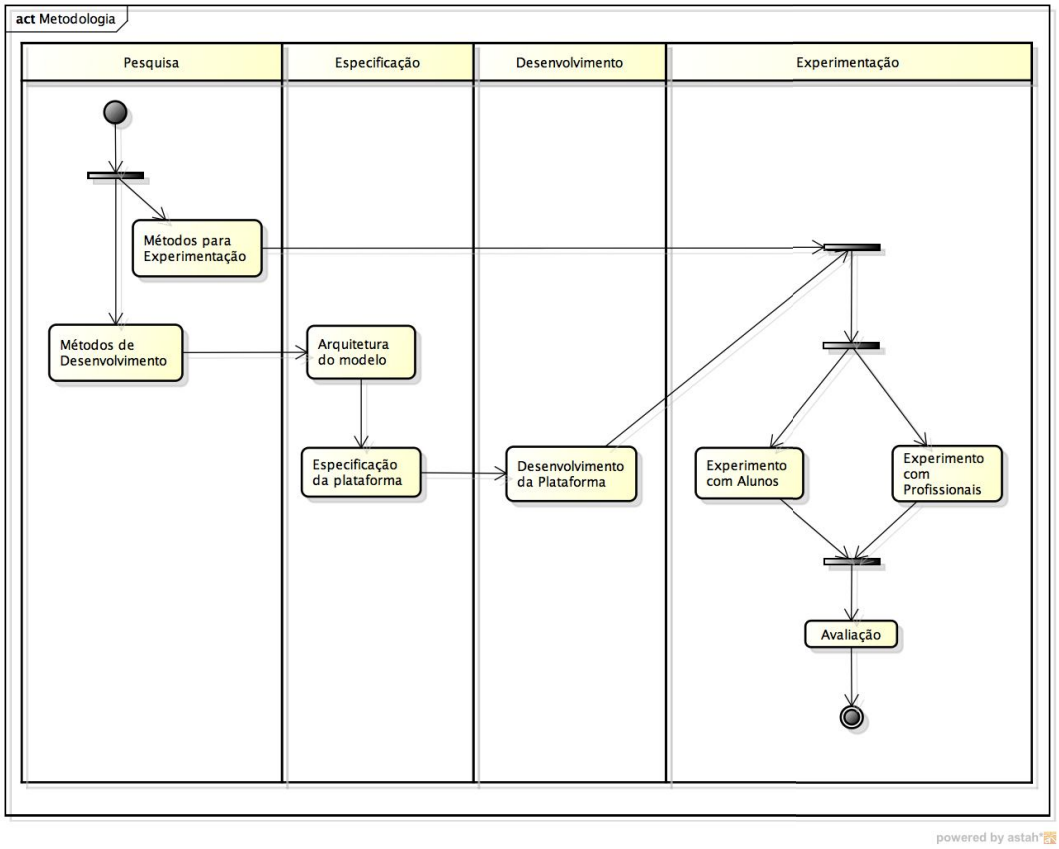

Figure 1 - Sequência das Atividades do Projeto.

Pesquisa sobre novos paradigmas, métodos e modelagem de programação de sistemas. Umas dessas técnicas é o MDA, que propõe uma arquitetura de desenvolvimento baseada em modelos.

Arquitetar uma plataforma de programação que integre modelagem e implementação, de forma a reutilizar os recursos projetados no ambiente de modelagem no ambiente de implementação. O objetivo é prover o modelo e a lógica do sistema para que, baseado no MDA, a plataforma consiga gerar bibliotecas, componentes ou até sistemas completos.

Experimentos com estudantes e profissionais serão desenvolvido para verificar ou validar o novo modelo. Para isso pesquisas sobre técnicas de avaliação ou verificação também serão realizadas.

Resultados e discussões

Desenvolvimento de uma metalinguagem de programação orientada a exemplos, que seja convertida em outras linguagens orientadas a objeto. Essa metalinguagem será utilizada de forma a complementar a modelagem realizada com o MOBI. E a especificação, arquitetura e desenvolvimento de uma plataforma de modelagem e programação de sistemas, baseado no MDA e na Estratégia *Bottom-Up*.

Com isso, visa-se a redução dos problemas apontados na Crise do Software. Contribuindo para a evolução dos modelos de programação. Além de permitir a construção de sistemas a partir de sua modelagem, através da geração de código em linguagem específica de plataforma descrita no MDA. Ao longo desse artigo serão discutidos alguns desses problemas e citados alguns exemplos de como o novo modelo pode melhor o processo de desenvolvimento de software.

Considerações Finais

Este artigo apresentou alguns problemas ainda não solucionados plenamente no âmbito do desenvolvimento de softwares. Também foram apresentadas algumas abordagens que podem auxiliar na resolução, ou amenizar, esses problemas. A consolidação dessas abordagens é proposta na forma de um novo modelo de programação baseado em exemplos. Centrada nas instâncias e em exemplos do mundo real para construção de software, o objetivo do modelo é tornar a programação mais dinâmica e adaptativa às mudanças que podem ocorrer em um projeto de software, o que ameniza alguns dos problemas causadores da crise do software.

Além disso, este novo modelo de programação tratará as instâncias do ambiente computacional como as instâncias do mundo real, independente de uma classe e livre para se vincular a quantas classes forem necessárias, já que no mundo real um indivíduo pode ser classificado em diversos papéis, o que melhora a organização semântica do código.

O próximo passo dessa pesquisa é a especificação detalhada do modelo de programação proposto e a implementação de uma plataforma de desenvolvimento que atenda as especificações do modelo. Uma metalinguagem de programação, que seja compatível com o MOBI, também deverá ser desenvolvida e utilizada na nova plataforma, para que os elementos modelados com o MOBI estejam disponíveis para serem utilizados no modo de implementação.

Referências

APPLE Inc. (13 de 12 de 2012). ***Programming with Objective-C***. disponível em Mac Developer Library: https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html#//apple_ref/doc/uid/TP40011210-CH1-SW1. Acesso em 29 de Out. de 2013,

GAMMA, E., JOHNSON, R., & HELM Richard, V. J. Design Patterns: elements of reusable object-oriented software. ***Addison-Wesley***, 1995

HURLIMANN, T. ***Modeling Languages: A new Paradigm of Programming***. 1998.

JORGE, E. **MOBI - Um Modelagem de Ontologia Baseado em Instância**. Salvador, Bahia, Brasil, 2012.

Naur, P. ***Techniques, Software Engineering: Concepts and Techniques.*** (Petrocelli/Charter, Ed.). (10 de 1969).

OMG. MDA ***Guide Version*** 1.0.1. Fonte: omg.org. (12 de 06 de 2003).

SEBESTA, R. Concepts of Programming Languages (9ª ed.). ***Addison-Wesley***., 2010

SOMMERVILLE, I. **Engenharia de Software** (Vol. 9). São Paulo, São Paulo, Brasil: Pearson Prentice Hall, 2011.

WAMPLER, D., & Clark, T. Multiparadigm Programming. ***IEEE Computer Society***, 2010.