Uma abordagem distribuída para o Desenvolvimento Orientado a Modelos

David Fernandes Neto[1], **Patrício Carneiro da Frota**[1], **Renata Pontin de Mattos Fortes**[1]

[1]Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo (ICMC-USP)
Caixa Postal – São Carlos – SP – Brazil

{david,renata}@icmc.usp.br, patricio@grad.usp.br

Resumo. *O Desenvolvimento Orientado a Modelos (Model Driven Development - MDD) é uma abordagem que tem ganhado cada vez mais espaço na indústria e na academia, trazendo grandes benefícios, como o aumento de produtividade, por exemplo. Contudo, ferramentas que auxiliem o desenvolvimento orientado a modelos de maneira colaborativa são praticamente inexistentes. Isso faz com que o MDD seja limitado ao desenvolvimento isolado e stand alone, dificultando que sistemas complexos e abrangentes sejam desenvolvidos com esse paradigma. Assim, este trabalho apresenta uma abordagem colaborativa orientada a modelos (CoMDD), a qual pretende evidenciar que a colaboração pode trazer benefícios ao MDD e permitir que desenvolvedores de MDD possam trabalhar colaborativamente e distribuídas, aumentando ainda mais a produtividade, possibilitando a troca de experiências e conhecimentos, e permitindo o desenvolvimento de sistemas complexos com o uso de MDD.*

Abstract. *The Model Driven Development (MDD) is an approach that has gained more space in industry and academia, bringing great benefits such as increased productivity, for example. However, tools to aid the development-oriented models in a collaborative manner is practically nonexistent. This makes the MDD limited to isolated development, making it difficult for complex and comprehensive systems be developed with this paradigm. Thus, this paper presents a model-driven collaborative approach (CoMDD), which aims to show that collaboration can bring benefits to the MDD and allow MDD developers to work collaboratively, further increasing productivity, enabling the exchange of experiences and knowledge and allowing the development of complex systems with the use of MDD.*

1. Introdução

O Desenvolvimento Orientado a Modelos (MDD - *Model-Driven Development*) é uma abordagem que se concentra na concepção e transformação de modelos. Os modelos são o foco central dos artefatos, que por meio de transformações podem gerar código-fonte [Teppola et al. 2009]. Seu objetivo é reduzir a distância semântica que existe entre o domínio do problema e o domínio da implementação/solução, utilizando modelos mais abstratos que protegem os desenvolvedores de software das complexidades inerentes à plataforma de implementação [France and Rumpe 2007].

Um dos principais benefícios desta abordagem é a reutilização. Modelos condensam conhecimento produzido durante atividades de análise, projeto e implementação,

de uma forma (idealmente) independente da plataforma de implementação. Assim, reutilizando-se um modelo, este conhecimento pode ser mais facilmente reaproveitado em outros contextos [Sherif et al. 2006].

A reutilização (ou reuso) de software é uma atividade altamente colaborativa em sua essência, pois normalmente o reutilizador de um artefato não é a mesma pessoa (e possivelmente não precisa pertencer à mesma equipe) que o produtor do artefato [Almeida et al. 2005].

Modelos são bons artefatos para comunicação e interação entre os *stakeholders* e para troca de experiências. Isso porque modelos geralmente são mais fáceis de entender do que código-fonte, de modo que não só os engenheiros de software participam da modelagem, mas também os especialistas de domínio e até os *stakeholders* podem participar. Assim, um modelo pode ser construído por diferentes pessoas ao mesmo tempo e distribuídas. Contudo, ferramentas que permitam a modelagem distribuída são desconhecidas.

Além disso, o desenvolvimento de software está mudando da programação manual para o MDD, como uma maneira de reduzir a crescente complexidade dos sistemas [Liggesmeyer and Trapp 2009, Aho et al. 2009] e o desenvolvimento de software é um processo intensamente colaborativo onde o sucesso depende da habilidade de criar, compartilhar e integrar informação [Lanubile 2009]. Desta forma, acredita-se que uma abordagem que junte os benefícios da colaboração ao MDD, possa trazer essencialmente o aumento de produtividade.

Embora o MDD tenha se mostrado uma abordagem promissora, emergindo ao longo das últimas décadas [Liggesmeyer and Trapp 2009] e apresentado ganhos de produtividade no desenvolvimento de sistemas [Aho et al. 2009] as soluções existentes ainda são incipientes em termos de ferramentas e processos necessários para auxiliar os desenvolvedores em seus trabalhos [Bendix and Emanuelsson 2009]. Isso se torna mais evidente quando observamos as ferramentas e processos que auxiliam o MDD e às que auxiliam o desenvolvimento tradicional (sem ser MDD).

No desenvolvimento tradicional, ferramentas permitem à equipe de desenvolvedores trabalharem e coordenarem atividades, combinarem código e dividirem tarefas, entre outros. Contudo, pela falta das mesmas ferramentas de apoio no contexto de MDD e pela falta do uso de modelos para criar conhecimentos compartilhados, as equipes de desenvolvimento trabalham de maneira mais restrita [Bendix and Emanuelsson 2009], [Lanubile 2009]. Assim, o MDD, que é uma abordagem que promove alta troca de experiências, não é usado em colaboração e nem de maneira distribuída, ao contrário do desenvolvimento tradicional. Parte deste trabalho está na observação do desenvolvimento tradicional e como pode-se agregar suas vantagens ao MDD.

Ainda no contexto de suporte a trabalhos colaborativos, percebe-se que, atualmente, há um movimento de migração de sistemas desktop para aplicações web. Observando o histórico desses sistemas acredita-se que brevemente, ter-se-á o desenvolvimento de softwares via web com muito mais vigor.

Assim as vantagens do MDD, o seu potencial para o desenvolvimento distribuído e a carência de ferramentas/processos para o MDD nos levou a identificarmos características do desenvolvimento tradicional com o intuito de tentar aplicar essas carac-

terísticas ao MDD para que ele seja aplicado de maneira distribuída.

Este trabalho apresenta uma abordagem colaborativa de desenvolvimento orientada a modelos (Colaborative Model Driven Development - CoMDD) que possibilita o desenvolvimento distribuído de software a fim de que desenvolvedores tradicionais possam adotar o MDD e receber seus benefícios e de que os atuais desenvolvedores que usam MDD ganhem com os benefícios da colaboração.

Na Seção 2 são apresentados os trabalhos relacionados, na Seção 3 é apresentado o conceito de MDD e suas vantagens, na Seção 4 é definido o CoMDD e na Seção 5 é apresentada a conclusão e o estágio atual do trabalho.

2. Trabalhos Relacionados

Levytskyy et al. (2009) [Levytskyy et al. 2009] foca em aplicações M&S (*Modeling & Simulation*) desenvolvidas na web. Os autores atentam para o fato de que a modelagem de artefatos requer o envolvimento de várias pessoas e que por isso o desenvolvimento tradicional da forma individual, usando ferramentas M&S, deverá ser substituído ou complementado com ferramentas mais poderosas que suportem a prática colaborativa dos usuários. O trabalho propõe um ambiente colaborativo baseado na web que seja capaz de se adaptar facilmente às necessidades de modelagem e simulação de um número de domínios de aplicações M&S arbitrário. Para isto os autores sugerem um *framework collaborative Modeling and online Simulation* (cMoS) como suporte para aplicações M&S desenvolvidas em plataforma web.

Outro trabalho relativo a colaboração em modelagens é o de Abeti et al. (2009) [Abeti et al. 2009], no qual é desenvolvido sob uma plataforma wiki dentro do contexto de gerenciamento de requisitos para *Business Process Reengineering* (BPR). Eles propõem um *framework* BPR que formalize o conhecimento empresarial por meio de um conjunto de modelos conectados com os requisitos do software e que também possibilite uma automação parcial da implementação do sistema. Esse *framework* provê uma wiki, denominada de *Wiki for Requirements (WikiReq)*, baseada na plataforma *Semantic MediaWiki*.

Um trabalho que contribuiu para o CoMDD foi o trabalho de Bendix e Emanuelsson (2009) [Bendix and Emanuelsson 2009] que identifica um problema quanto às limitações dos desenvolvedores em MDD, principalmente relacionadas a carência de ambientes que auxiliem o desenvolvimento orientado a modelos de maneira colaborativa. Assim, por meio de casos de uso, os autores realizam um levantamento dos requisitos que um ambiente colaborativo (em particular orientado a modelos) precisa satisfazer, de tal forma que possibilite um time de desenvolvedores trabalhar eficientemente neste ambiente.

Existem na literatura trabalhos que propõem técnicas de *merge* e de comparação entre modelos, como: [Sriplakich et al. 2006, Wenzel 2008, Kelter et al. 2005], entre outros. Tais trabalhos são importantes para o CoMDD devido a importância dessas técnicas para o desenvolvimento colaborativo. Existe também *workshops* sobre comparação e versionamento de modelos [CVS 2008] mostrando a importância da área.

Existem muitas ferramentas que possibilitam o MDD como o Eclipse, Matlab/Simulink, WebRatio, Labview, AndroMDA, mas nenhuma delas possibilita edição si-

multânea por exemplo. Já ferramentas como ThoughtSlinger, Notapipe, Zoho, Moonedit e o Google Docs, permitem edição simultânea, mas não desenvolvimento de sistemas, visto que são ferramentas de escritório, ao contrário deste trabalho que tenta juntar o desenvolvimento de sistemas e a edição simultânea, por exemplo.

No CoMDD, a idéia é que todo o processo possa ser realizado pela web, sem a necessidade de programas instalados no computador e que permita o desenvolvimento distribuído de software.

3. Model Driven Development

O Desenvolvimento Orientado a Modelos (*Model-Driven Development* - MDD) é uma abordagem da Engenharia de Software que consiste na aplicação de modelos para elevar o nível de abstração, no qual os desenvolvedores criam e evoluem o software. Sua intenção é tanto simplificar (tornar mais fácil) quanto formalizar (padronizando, de forma que a automação seja possível) as várias atividades e tarefas que formam o ciclo de vida do software [Hailpern and Tarr 2006]; ou numa definição mais singela: o MDD é a simples noção de construir um modelo de um sistema e depois transformá-lo em algo real [Mellor et al. 2003].

O que caracteriza o MDD são os modelos como foco primário do desenvolvimento de software, ao invés das linguagens de programação. Os modelos são usados para descrever vários aspectos do software e para automatizar a geração de código. A principal vantagem disto é poder expressar modelos usando conceitos menos vinculados a detalhes de implementação, além do fato de modelos serem mais próximos do domínio do problema. Isto torna os modelos mais fáceis de se especificar, entender e manter, do que abordagens que não usam modelos. E, em alguns casos, ainda é possível os especialistas do domínio produzirem os sistemas ao invés dos especialistas da tecnologia de implementação [Selic 2003, Sriplakich et al. 2006]

Portanto, o modelo no MDD é um artefato primordial para o desenvolvimento e dessa forma se contrapõe ao desenvolvimento convencional de software, no qual os modelos são utilizados para representar o problema e servem *apenas* para auxiliar o desenvolvimento. No MDD, a idéia é que esses modelos possam gerar o código (e outros artefatos) e não fiquem limitados apenas como um auxílio ao desenvolvedor.

Vantagens do MDD

Segundo [Kleppe et al. 2003, Lucredio 2009] o MDD apresenta grandes benefícios como:

- Produtividade: fatores como redução de atividades repetitivas e manuais e o aumento da possibilidade de reúso, podem contribuir para o aumento da produtividade no processo de desenvolvimento;
- Portabilidade e Interoperabilidade: como o modelo é independente de plataforma, um mesmo modelo pode ser transformado em código para diferentes plataformas;
- Corretude: o MDD evita que os desenvolvedores exerçam atividades repetitivas e manuais para gerar código; dessa maneira, evita-se também alguns erros como: geradores de código não introduzem erros acidentais, como o de digitação, por exemplo;
- Manutenção: alterações no código respectivas à manutenção podem requerer o mesmo esforço produzido durante o desenvolvimento;

- Documentação: modelos são o artefato principal do processo de desenvolvimento e por isso não se desatualizam, pois o código é gerado a partir deles e com isso a documentação se mantém também sempre atualizada;
- Comunicação: modelos são mais fáceis de entender do que código-fonte, assim isso facilita a comunicação entre os desenvolvedores, os *stakeholders* e demais envolvidos.

Analisando tais definições de MDD e suas vantagens apresentadas, pode-se concluir que MDD é um conceito bem definido, pois diferentes autores apresentam as mesmas idéias de que MDD é uma elevação no nível de abstração do desenvolvimento de software, no qual o modelo passa de um artefato auxiliar para uma peça fundamental no processo de desenvolvimento. Em virtude dessa elevação na abstração, muitos problemas relativos a portabilidade, interoperabilidade, corretude, manutenção e documentação, são reduzidos nesta abordagem. Contudo, é fato que mesmo diante de tantos benefícios, inclusive o aumento de produtividade, o MDD ainda está longe de ser a solução para todos os problemas relativos ao desenvolvimento de software.

Principais elementos do MDD

Lucrédio [Lucredio 2009] define os principais elementos do MDD como sendo:

- *Ferramenta de modelagem*: através dela o engenheiro de software produz modelos que descrevem conceitos do domínio, segundo uma linguagem específica de domínio (*Domain Specific Language* - DSL). Para que os modelos sejam capazes de gerarem código de maneira automática, eles devem ser semanticamente completos e corretos;
- *Ferramenta para definir as transformações*: os modelos serão transformados em outros modelos ou em código fonte; portanto é necessário uma ferramenta para definir transformações na qual o engenheiro de software constrói regras de mapeamento de modelo para modelo ou de modelo para código;
- *Mecanismo que aplique as transformações*: com o modelo e as transformações definidas, resta um mecanismo que aplique as transformações nos modelos. É importante que além de aplicar as transformações, esse mecanismo mantenha informações de rastreabilidade, possibilitando saber a origem de cada elemento gerado.

4. Definição do CoMDD

O CoMDD consiste de uma ferramenta de modelagem e de um mecanismo que aplica as transformações (ambas funcionalidades estão na wiki), permitindo a colaboração de várias pessoas. Assim, esta seção define os três pontos principais do CoMDD (seções 4.1, 4.2 e 4.3) e suas principais vantagens (seção 4.4).

4.1. Edição simultânea, Merge e Divisão de Tarefas

No desenvolvimento tradicional e distribuído, geralmente, os desenvolvedores fazem uma cópia do servidor para sua máquina local, alteram essa cópia e depois submetem uma versão modificada para o servidor, havendo o *merge* de documentos neste momento. Quando duas pessoas baixam o mesmo arquivo do servidor, o alteram e depois o submetem novamente, haverá conflito de dados. Uma forma de solucionar esse problema é

através da edição simultânea. Desta forma o CoMDD usa a edição simultânea e define um processo para que desenvolvedores trabalhem no mesmo artefato e façam *merge* só quando necessário.

A abordagem analisa 4 pontos:

1. **Artefatos diferentes e funcionalidades diferentes** É uma tarefa auto-contida e independente, ou seja, neste caso o desenvolvedor pode criar, editar e compilar um artefato, de modo que outro desenvolvedor esteja trabalhando em outra tarefa auto-contida e independente. Assim o desenvolvimento de uma tarefa não interfere na outra.
2. **Mesmo artefato e mesma funcionalidade** Quando dois desenvolvedores estão trabalhando juntos no mesmo artefato eles têm suporte à edição simultânea dele e tem suporte de chat e conferência de voz.
3. **Mesmo artefato e funcionalidades diferentes** Aqui tem um problema, porque caso o desenvolvedor A queira compilar enquanto o desenvolvedor B estiver editando o artefato haverá erros de compilação, por isso, para contornar o problema, o servidor cria uma cópia do artefato que está sendo editado para cada desenvolvedor, como se fosse uma cópia local, e depois os desenvolvedores fazem o *submit* e o servidor faz o *merge* dos artefatos. É apenas neste caso em que haverá o *merge*.
4. **Artefatos diferentes e mesma funcionalidade** Essa atividade é evitada pelo CoMDD porque gera conflitos e não promove discussões. Assim, ou um artefato é editado simultaneamente por várias pessoas ou apenas uma pessoa pode editá-lo.

Com a edição simultânea promovemos mais discussão, pois todos envolvidos participam da edição do artefato. Para a edição simultânea não gerar conflitos, define-se duas regras:

- Duas ou mais pessoas não podem editar, ao mesmo tempo, a mesma linha de código. Para isso elas devem editar partes separadas do código (aplicada para todos os quatro casos).
- Caso alguém queira compilar, todas devem estar de acordo porque se não dificilmente vai compilar, dado que uma pode estar digitando enquanto a outra quer compilar (somente para o segundo caso).

Uma das vantagens da edição simultânea na web é a possibilidade de que pessoas geograficamente distantes possam trabalhar colaborativamente em um mesmo modelo.

Uma outra possibilidade para o uso da abordagem, além do desenvolvimento de sistemas, é em treinamentos de programação, por exemplo; de modo que professor e aluno possam programar juntos.

4.2. Transformações e Validações

Um modelo, objeto editado colaborativamente ou individualmente, pode ser transformado em código-fonte ou em outros modelos, mas para isso acontecer transformações devem ser definidas. O CoMDD precisa de uma linguagem específica de domínio com transformações definidas para poder gerar código-fonte de um metamodelo, de forma que o modelo seja validado, ou seja, um modelo só tem significado e pode gerar código-fonte caso ele siga um metamodelo.

4.3. Comunicação

Para que o CoMDD permita o desenvolvimento distribuído de software é necessário um suporte a comunicação. Assim, o CoMDD define o uso de uma interface de comunicação textual ou por voz e que possibilite a contribuição de todos para termos modelos mais completos.

4.4. Benefícios

Os benefícios dessa abordagem são:

- Aumento da discussão e participação entre os envolvidos
- Possibilidade de Pair Programming por pessoas distribuídas geograficamente
- Registro das decisões tomadas durante o desenvolvimento (Design Rational) a partir da interface de comunicação
- Independência de tecnologia e de programas instalados localmente
- Aumento do reuso e produtividade, uma vez que a abordagem usa MDD

5. Conclusão

A partir da abordagem definida espera-se que o desenvolvimento orientado a modelos seja usado por mais desenvolvedores e que os desenvolvedores que usam MDD possam se beneficiar das vantagens de trabalhar-se colaborativamente. Atualmente o CoMDD está sendo desenvolvido pelo ICMC-USP e sua próxima fase será a avaliação em um estudo de caso.

References

(2008). *CVSM '08: Proceedings of the 2008 international workshop on Comparison and versioning of software models*, New York, NY, USA. ACM. Program Chair-Ebert, Jürgen and Program Chair-Kelter, Udo and Program Chair-Systä, Tarja.

Abeti, L., Ciancarini, P., and Moretti, R. (2009). Wiki-based requirements management for business process reengineering. In *Wikis for Software Engineering, 2009. WIKIS4SE '09. ICSE Workshop on*, pages 14–24.

Aho, P., Merilinna, J., and Ovaska, E. (2009). Model-driven open source software development - the open models approach. In *4th International Conference on Software Engineering Advances, ICSEA 2009*, pages 185–190, Porto, Portugal.

Almeida, E. S. d., Alvaro, A., Lucredio, D., Garcia, V. C., and Meira, S. R. d. L. (2005). Towards an Effective Software Reuse Process. In *The 17th Conference on Advanced Information Systems Engineering (CAiSE'05)*, Porto, Portugal. Springer-Verlag LNCS.

Bendix, L. and Emanuelsson, P. (2009). Collaborative work with software models - industrial experience and requirements. In *Model-Based Systems Engineering, 2009. MBSE '09. International Conference on*, pages 60–68.

France, R. and Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. In *29th International Conference on Software Engineering 2007 - Future of Software Engineering*, pages 37–54, Minneapolis, MN, USA. IEEE Computer Society.

Hailpern, B. and Tarr, P. (2006). Model-driven development: the good, the bad, and the ugly. *IBM Syst. J.*, 45(3):451–461.

Kelter, U., Wehren, J., and Niere, J. (2005). A generic difference algorithm for uml models. In Liggesmeyer, P., Pohl, K., and Goedicke, M., editors, *Software Engineering*, volume 64 of *LNI*, pages 105–116, Ronneby, Sweden. GI.

Kleppe, A. G., Warmer, J., and Bast, W. (2003). *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Lanubile, F. (2009). Collaboration in distributed software development. In *Software Engineering, International Summer Schools, ISSSE*, pages 174–193, Berlin, Heidelberg. Springer-Verlag.

Levytskyy, A., Vangheluwe, H., Rothkrantz, L. J., and Koppelaar, H. (2009). Mde and customization of modeling and simulation web applications. *Simulation Modelling Practice and Theory*, 17(2):408 – 429.

Liggesmeyer, P. and Trapp, M. (2009). Trends in embedded software engineering. *IEEE Software*, 26:19–25.

Lucredio, D. (2009). *Uma Abordagem Orientada a Modelos para Reutilizacão de Software*. PhD thesis, Universidade de São Paulo, Instituto de Ciências Matemáticas e de Computacão.

Mellor, S. J., Clark, A. N., and Futagami, T. (2003). Guest editors' introduction: Model-driven development. *IEEE Software*, 20:14–18.

Selic, B. (2003). The pragmatics of model-driven development. *IEEE Softw.*, 20(5):19–25.

Sherif, K., Appan, R., and Lin, Z. (2006). Resources and incentives for the adoption of systematic software reuse. *International Journal of Information Management*, 26(1):70–80.

Sriplakich, P., Blanc, X., and Gervais, M.-P. (2006). Supporting collaborative development in an open mda environment. In *Software Maintenance, 2006. ICSM '06. 22nd IEEE International Conference on*, pages 244–253.

Teppola, S., Parviainen, P., and Takalo, J. (2009). Challenges in the deployment of model driven development. In *4th International Conference on Software Engineering Advances, ICSEA 2009*, pages 15–20, Porto. cited By (since 1996) 0; Conference of 4th International Conference on Software Engineering Advances, ICSEA 2009, Includes SEDES 2009: Simposio para Estudantes de Doutoramento em Engenharia de Software; Conference Date: 20 September 2009 through 25 September 2009; Conference Code: 78651.

Wenzel, S. (2008). Scalable visualization of model differences. In *CVSM '08: Proceedings of the 2008 international workshop on Comparison and versioning of software models*, pages 41–46, New York, NY, USA. ACM.