CoMDD: uma abordagem colaborativa para auxiliar o desenvolvimento orientado a modelos

David Fernandes Neto

CoMDD: uma abordagem colaborativa para auxiliar o desenvolvimento orientado a modelos

David Fernandes Neto

Orientadora: ***Profa. Dra. Renata Pontin de Mattos Fortes***

Monografia apresentada ao Instituto de Ciências Matemáticas e de Computação — ICMC/USP, para o exame de Qualificação, como parte dos requisitos para obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

USP - São Carlos
Janeiro/2012

Resumo

O

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

API *Application Programming Interface*

CDE *Collaborative Development Environment*

CAPÍTULO

1

Introdução

Este capítulo descreve a área na qual o trabalho está inserido, apresenta o problema de pesquisa e a justificativa para sua escolha, define a hipótese e o objetivo do trabalho. O método, resultados esperados e limitações também são apresentadas aqui.

1.1 Tema - descrição geral da área, o Problema/Justificativa

O desenvolvimento orientado a modelos (*Model Driven Development* - MDD) é um método de desenvolvimento de software que foca na criação de modelos, ou de abstrações, como principais artefatos de desenvolvimento de software e nas transformações desses modelos para gerar código-fonte. 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 (Teppola et al., 2009; France e Rumpe, 2007).

A principal vantagem do MDD 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. E, em alguns casos, ainda é possível os especialistas do domínio produzirem os sistemas ao invés dos especialistas da tecnologia de implementação (**??**)

E ainda são vários os benefícios que o MDD possui, como: produtividade, interoperabilidade, reuso, ... cite???. São diversos trabalhos (cite???) que propõe e que usam o MDD como abordagem

para desenvolvimento de software. Ainda, a indústria de sistemas embarcados usa o MDD antes do surgimento da UML ou da MDA[1] (Model Driven Architecture), entre outras áreas da indústria que também usam o MDD.

Entretanto, em relação à modelos textuais[2], como equipes de desenvolvimento de software trabalham juntas?

Apesar de alguns trabalhos cite??? dizerem que as ferramentas para MDD são quase inexistentes, quando trabalha-se com modelos textuais, as ferramentas para MDD são as mesmas que existem para as abordagens tradicionais como por exemplo Java, C, C++ e etc. Ou seja, assim como pode-se usar o Eclipse[3] como IDE de desenvolvimento e o Subversion[4] para compartilhamento de código, pode-se usar ambas ferramentas no MDD. Os modelos textuais são textos da mesma forma que um código-fonte Java é texto, assim não há razão técnica para não usar-se o uma IDE que suporte MDD e um sistema de controle de versão qualquer.

Assim, uma possibilidade para trabalhar colaborativamente com o MDD é o uso de uma ide associada à um sistema de controle de versão[5], entretanto, os sistemas de controle de versão podem causar conflitos quando dois ou mais usuários modificam o mesmo documento e submetem ao servidor[6].

Neste momento, desenvolvedores terão que dedicar certo tempo para resolver os conflitos, e às vezes esses conflitos só podem ser resolvidos manualmente. Além disso, para os desenvolvedores saberem como resolver o conflito corretamente eles precisam estar em contato entre si e para eles decidirem como realizar o merge eles ainda precisam estar disponíveis no momento (PrudÃªncio et al., 2012).

Ainda há um outro ponto ao se usar as IDEs e os SCV: a necessidade de instalação desses software, ou seja, para as equipes usarem uma IDE e um sistema de controle de versão é preciso que cada desenvolvedor tenha instalado esses softwares em seus computadores [7]. O que significa dizer que se um *stakeholder* estiver interessado em acompanhar o desenvolvimento de um software ele irá precisar instalar em seu computador essas ferramentas [8], o que pode-se dizer ser uma tarefa difícil e trabalhosa. Isso sem contar que o *stakeholder* deverá saber como funciona essas ferramentas. Se ele quiser participar da edição dos modelos[9] deverá ainda conhecer como funciona a geração de códigos da IDE, como fazer check-in (), check-out (), update () e etc. De forma que, embora para

[1] A MDA (Model Driven Architecture) é um pradão da OMG como abordagem do MDA, com uso da UML para a modelagem

[2] Ver capítulo de DSLs

[3] http://eclipse.org/

[4] ???

[5] Este trabalho está focando somente no uso de versionadores, pois o estudo de sistemas de gerenciamente de configuração tornaria este trabalho muito extenso.

[6] Isso no caso de uma política de versionamento *otimistic*, a qual permite o desenvolvimento concorrente.

[7] Será discutido no Capítulo XXX ferramentas que podem ser usadas na web, entretanto a maioria dessas ferramentas são incipientes e não são usadas nas empresas/academia.

[8] No experimento deste trabalho (descrito no capítulo XXX, dois profissionais experimentes levaram cerca de quarenta minutos para instalar e configurar o Eclipse e o plugin do svn

[9] Este trabalho parte do uso de MDD como abordagem de desenvolvimento e por isso está-se falando de MDD

um usuário não desenvolvedor trabalhar com modelos seja mais fácil que código-fonte cite?? essa vantagem é perdida quando há a necessidade do uso dessas ferramentas por parte deste.

Além do que, todo esse processo de instalação e de check-in, check-out, update e etc, é relativamente demorado, de forma que é uma outra desvantagem além da necessidade de instalação e conhecimento desses conceitos. Isto fica claro no trabalho de Jones (Jones e Scaffidi, 2011) quando ele afirma que cientistas raramente usam controle de versão devido ao alto custo inicial para configuração desses sistemas.

Mesmo que o usuário ainda receba um treinamento, cada ferramenta é diferente da outra, por exemplo, o svn é diferente do git que é diferente do da microsoft, sendo não só necessário o conhecimento da instalação, uso, como também da lógica de funcionamento dessas ferramentos, pois elas tem diferente abordagens, principalmente o git que possui o paradigma mais diferente desses.

Por outro lado, as wikis são ferramentas conhecidas por serem simples de usar e ágeis para colaboração cite??. De forma que dependendo do problema o uso de uma wiki ao invés de uma IDE e de um SVN, no MDD, pode resolver todos os problemas descritos até aqui, pois elas funcionam em um navegador, possuem um controle de versões com histórico, possibilidade para comentários e, no caso da Xwiki[10] não tem problemas de conflito pois podem travar a edição de um mesmo documento para um segundo usuário, bastando apenas implementar na wiki um suporte a modelagem e geração de código-fonte.

Ainda, observando o histórico de aplicações como músicas, vídeos, armazenamento de arquivos, jogos e etc, têm convergido para a web, como exemplo disso tem-se o Grooveshark®[11] (Reprodutor de músicas on-line), o Youtube® [12](reprodutor de vídeos on-line), o Dropbox®[13](Armazenador de arquivos on-line) e até mesmo IDEs estão convergindo para a web, como é o caso do c9(). De modo que ter uma ferramenta possível de editar, compartilhar e versionar modelos, gerar código-fonte e ainda permitir a entrada de comentários pode beneficiar desenvolvedores e stakeholders.

Contudo, seria possível uma wiki de ser usada como uma ferramenta de desenvolvimento e compartilhamento de modelos, da mesma forma que uma ide e svn?

1.2 Hipótese e objetivo

A partir do cenário levantado na seção anterior, definiu-se a seguinte hipótese que norteia este trabalho:

[10] xwiki site ...
[11]
[12]
[13]

Para desenvolver um sistema colaborativamente usando MDD[14], é possível usar uma wiki ao invés do processo tradicional[15] de desenvolvimento.

O objetivo principal deste trabalho é tentar evidenciar que a hipótese de pesquisa é verdadeira para um caso específico (a ser tratado no estudo de caso) e incentivar o uso da abordagem desenvolvida neste trabalho para outros casos também (ver sessão a seguir: método).

1.3 Metodo

Para tentarmos evidenciar que nossa hipótese é possível/verdadeira faremos uma avaliação. Será pedido para duas equipes diferentes a mesma tarefa, sendo que uma delas usara o eclipse+svn e a outra uma wiki apenas. Ambas usarão dsls e geração de código, mas nao terao os recursos tradicionais de uma ide como import, highlight sugestao ... Para isso será implementado a dsl no eclipse e numa wiki usando o xtext/xpand e o antlr respectivamente. // Foi desenvolvido um sistema (implementação do comdd) para se realizar um estudo de caso a fim de evidenciar que a hipótese do trabalho é verdadeira ou não// mais detalhes no capitulo de desenvolvimento

Para isso será mostrado as potencialidades da abordagem com o intuito dos usuarios poderem avaliar se a abordagem pode ser interessante para uso ou nao. Com isso acredito que teríamos mais produtividade e a aproximação maior de todos os envolvidos no processo, e com isso softwares mais próximos dos que o cliente precisa.

1.4 Escopo/Limitações

Nao é foco deste trabalho comparar o desenvolvimento de codigo com o de mdd uma vez que ha uma vasta literatura sobre isso. Este trabalho parte do principio de que o mdd é vantajoso para o desenvolvimento mais que o desenvolvimento tradicional (segundo ....)

O foco deste trabalho não está no estudo da comunicação dos integrantes, contudo entende-se que para um trabalho colaborativo a comunicação é excencial.

1.5 Resultados Esperados

Espera-se que com este trabalho o uso de dsls seja cada vez mais empregado no desenvolvimento de sistemas e de que as wikis evoluam como uma ide para que um dia tenhamos o uso de wikis associadas a dsls para sistemas grandes e complexos.

[14] No caso estamos nos referindo a dsls como abordagem de mdd. No capítulo de revisão bibliográfica isso ficará mais claro

[15] Definimos processo tradicional de desenvolvimento como sendo o uso de uma IDE e de um sistema de controle de versão *otimistic*

conclusão da introdução: assim, com o uso de mdd resolvemos vários problemas como ... e com o uso de wikis resolvemos outros problemas como ..., por isso o objetivo deste trabalho é ...

No capitulo 2 ...

CAPÍTULO

2

Trabalhos Relacionados

Artigos

Programas semelhantes já em uso: sugestao do michetti, aquele que o chiquito mandou na lista, ...

Ferramentas de MDD: eclipse, MPS, ...

Ferramentas que auxiliam o trabalho colaborativo (sharepoint)...

CAPÍTULO

3

Revisão Bibliográfica

3.1 Model Driven Development

Definição ...

Pq MDD? Vantagens do uso de Modelos (ainda é possível checar se um modelo é satisfatório ou nao, mas um código nao é possível) e do MDD, A importancia do MDD

Justificativa de MDD na abordagem: Modelos sao colaborativos e mais faceis de entender por nao desenvolvedores,

Uma criança pode programar usando uma wiki mas nao usando eclipse e svn, mesmo que no eclipse seja modelos. A crianca é do dominio e o desenvolvedor o especialista// nivel da solução: java, c .... nivel do problema: UML, p ex //

1-Código-fonte é mais complicado de ser desenvolvido por mais de uma pessoa, logo é também menos reutilizável. Só o especialista consegue fazer e nao o cara do dominio. -> falar mal do desenvolvimento tradicional // 2- Modelos são mais colaborativos, simples, genéricos e reusáveis (nao falar em reuso pra nao criar uma expectativa errada pro leitor) e permitem que não especialistas possam desenvolver/acompanhar; alem de poder-se checar por conformidade os modelos

3.2 Domain Specific Languages

Uma abordagem de MDD...

3.2.1 Templates

3.3 Colaboração

o que é colaboração e quais seus benefícios?

Como as pessoas colaboram no desenvolvimento de software, Falar de como é feito o desenvolvimento tradicional colaborativo -> Arquitetura do desenvolvimento tradicional

3.4 Wiki

3- Artefatos online (wikis, gdocs) são mais colaborativos que controles de versao, bug tracks, ou outras ferramentas de gestao de configuração por terem interfaces mais amigáveis -> Assim a Web aumenta ainda mais colaboração: justificativa para wikis

Colaboracao e MDD: SVN

Colaboração na wiki

CAPÍTULO

4

Desenvolvimento

Definição do CoMDD ...

Funcionalidades: Ferramenta de Modelagem // Transformações // Suporte a dois tipos de comentarios, o da pagina principal e o das alterações // Suporte a Controle de Versões: histórico // Controle de Acesso // merge: plugin ?? // pode-se criar um bate-papo na wiki // wiki como ide: highlight na wiki

Deficiencia do CoMDD: nao compila, entao subentende-se que o codigo que ela gera esta sintaticamente correto, mas ela pode retornar erros // como integrar testes?

Possibilidade: é possível alterar a propria gramatica, bem como as transformações ainda pela propria wiki, mas isso seria um script

Vantagens do CoMDD: sao as vantagens da wiki associadas as do mdd DDS Produtividade Comunicação Pair Programming??

4.1 Arquitetura do CoMDD

Mostrar a arquitetura

Comparar o desenvolvimento tradicional com ou sem MDD com a nossa abordagem numa figura

Assegurar a integridade e consistência de artefatos versionados num ambiente que suporte acesso concorrente é um problema difícil (Zhang e Ray, 2007), portanto para evitar os conflitos de edição e merge adotou-se uma abordagem *pessimistic*, ou seja, uma política em que o artefato é bloqueado quando já está sendo editado por outra pessoa. Aplicando essa política, conflitos de

edição são evitados e com isso desenvolvedores não gastarão tempo ou esforços resolvendo conflitos. Ainda, a política *pessimistic* pode ajudar na comunicação, pois os desenvolvedores saberiam quando outra pessoa estaria modificando o mesmo artefato, incentivando a comunicação entre os desenvolvedores com os mesmos interesses (PrudÃªncio et al., 2012; Sarma et al., 2003). Uma outra possível solução seria a edição simultânea estilo google docs citeMeu trabalho!...

4.2 Método

: Colocar a ordem do desenvolvimento dos procedimentos para vibializar a execução na wiki. Mostrar os passos do desenvolvimento.

4.2.1 Estudo de Caso

ESTUDO DE CASO: CONCLUSAO DO ESTUDO, COLOCAR AQUI o anexo com as perguntas

Escopo do trabalho/ou experimento: Este trabalho foca principalmente em sistemas de pequeno porte, no caso foi testado para uma classe e duas pessoas, mas acredita-se que para algo em torno disso a abordagem funcione também.

Colocar os modelos transformados -> Detalhes no anexo

falar da edição de subartefatos: numa wiki é muito simples voce criar uma pagina dentro de outra [[]]. por exemplo, acho que outras wikis sustentam melhor o merge, ver a wikipedia

Limitações do trabalho: avaliar em que casos pode-se usar uma wiki no lugar de um svn nao é escopo deste trabalho // A abordagem pessimistica possui suas desvantagens como "In contrast, the pessimistic policy has some disadvantages. First, locks can cause an unnecessary serialization over the development process. Sometimes different developers can independently modify different parts of the same CI, but using a lock-based approach that is not possible. In addition, locks can cause a false sense of security (Collins-Sussman et al., 2004). Developers may think that their work will not be affected by the work of other developers due to the locks. However, sometimes CIs rely on other CIs that can be modified by other developers. This situation leads to indirect conflicts (Sarma et al., 2003). (PrudÃªncio et al., 2012) Assim, um trabalho futuro seria no estudo de estratégias otimísticas ou até mesmo na adoção de ambas como propõe o trabalho de Prudencio et al. (PrudÃªncio et al., 2012).

mas qual a diferença da minha dsl para jogar tudo em métodos de poo? vc nao trabalha com modelos e sempre fica os detalhes de plataforma juntos com a logica. com mdd vc joga tudo nas transformações e trabalha em alto nivel.

4.2.2 Resultados do Estudo de Caso

conclusão: 100% dos usuarios, quando questionados se para a dada tarefa que abordagem usariam, responderam o CoMDD.

CAPÍTULO

5

Conclusão

5.1 Considerações finais

O CoMDD é uma abordagem que prega mdd colaboartivamente. No caso foi usado uma wiki, mais especificamente a xwiki, mas poderia ser qualquer outra wiki ou plataforma web de simples uso e que promovesse velocidade. Com as atuais tecnologias a wiki é que melhor se encaixa nesse perfil. Em relação ao domínio também, espera-se que para qualquer domínio o CoMDD possa servir, entao no caso a dsl foi para robos autonomos moveis, mas também poderia ser para qualquer outro domínio // O comdd nao exige uma wiki necessariamanete, mas ela é tecnologia atual mais proxima da necessaria. Entretanto, muito ainda deve ser desenvolvido em cima das wikis para que elas possam de fato serem usadas como ferramentas de desenvolvimento de codigo, ao inves de apenas documentação ou afins.

Este trabalho está longe de criar um ambiente de desenvolvimento tão robusto e avançado quanto hoje são as ides, bug trackers, svns... até porque essas ferramentas já tem anos de maturidade. Este trabalho objetiva sim mostrar que o alinhamente de duas tecnologias aparentemente nao interligáveis é capaz de desempenhar papel semelhante e motivar novos estudos nesse caminho. O ideal seria transformar a wiki numa ide de fato, criar um merge de modelos, permitir o desenvolvimento simultaneo e a edição em tempo real que hoje tem o google. Mas a ideia de que quando voce edita um arquivo esta editando diretamente no servidor e portanto a cópia final simplifica (talvez agiliza tbm) o trabalho evitando conflitos de mege (afirmação forte). As wikis nunca foram pensadas para esse fim e por isso podem melhorar para este propósito.

É interessante para projetos menores, que tem a interedependencia entre suas classes nao muito grande, com poucas classes e que estas dependem pouco entre si, mas que necessitam fortemente

da colaboração de várias pessoas, e se torna mais interessante quando essa pessoas nao sao desenvolvedoras.

Ainda pode ser util para elaboração de algoritmos, mas que nao foi testado, ou até mesmo para código-fonte no caso da edição em tempo real.

concluir a hipotese da seguinte forma, mais ou menos: MDD é bom por causa ...

E Wikis sao boas por causa ...

E é possível integrar mdd e wiki para desenvolvimento (implementação e estudo de caso)

Portanto, ... afirmar a hipótese: 4-MDD na Wiki permite mais colaboração(wiki) e produtividade(MDD) (porque o comdd é mto mais acessível e pela sua simplicidade as pessoas podem colaborar mais facilmente), no desenvolvimento de software -> isso que tenho q avaliar , do que traria com eclipse e svn (ideal de avaliar)

Premissa I: o MDD implica em benefícios de produtividade para o desenvolvimento, modelos possibilitam mais comunicação principalmente por nao especialistas Premissa II: a wiki é excelente para comunicação, troca de conhecimento, rápida e dinâmica. Conclusão: um mdd na wiki traria beneficios associado ao uso de MDD com ao uso de uma wiki no desenvolvimento de sistemas

5.2 Contribuições

Será que o CoMDD traz inovação qnd comparado ao uso do eclipse+svn?

Quais suas contribuições?????

Que beneficios em termos de colaboração o comdd traz qnd comparado a abordagens eclipse+svn?

Que outros benefícios posso trazer?

As minhas contribuições estao totalmente relacionadas as contribuições do mdd e da wiki, por ex: por que as pessoas nao usam o drupal ou o joomla ou fazem um site para documentação? Da mesma forma. Nao estamos propondo o fim dos svns ou (mas sim das ferramentas locais, pois tudo pode ser web hoje, vide o c9 - ide na nuvem), mas estamos dizendo: olha, com uma wiki e uma dsl vc pode fazer mta coisa e colaborar mto mais.

O principal diferencial deste resultado em relação a outros trabalhos existentes é o aspecto colaborativo, permitindo a edição de modelos e a transformação de software por diferentes desenvolvedores sem a necessidade de ferramentas/ ambientes individuais instalados nas estações de trabalho dos desenvolvedores.

que beneficios o comdd traz quando comparado ao eclipse e svn? nao instalar programas vc ganha agilidade, maior participação, mais pessoas tem acesso a edição de maneira facilitada, ou seja, quando vc facilita o acesso e a edicao vc permite que mais pessoas possam colaborar. É só pensar no contrario para ver que é verdadeiro, pois se o desenvolvedor precisar do dobro de tempo para instalar um svn/ide e os mesmos forem 3 vezes mais dificeis de serem usados temos com isso menos pessoas nao desenvolvedoras participando. Talvez para o desenvolvedor nao importe

o quanto seja dificil usar um svn e o quanto isso va demandar tempo no desenvovlimento de software (na realidade isso importa pq vai atingir diretamente na produtividade), mas para o nao desenvolvedor isso importa muito -> posso colocar uma pergunta assim no questionario dos videos.

5.3 Trabalhos Futuros

Usar uma wiki para densenvolvimento que nao seja mdd talvez nao seja tao interessante porque o nível de colaboração de um mesmo artefato é menor que este artefato como modelo. Mas talvez essa seja uma pergunta interessante (para que casos a colaboração numa wiki é eficaz?) a qual este trabalho não responde.

eu tenho que dizer que o estudo de caso foi o suficiente para provar ... (ser bem especifico), mas que para (aqui posso ser mais abrangente) é necessário melhorar a ferramenta (deixar ela mais proxima de uma ide) e partir para problemas mais reais/complexos.

implementações futuras: Edição das transformações e da gramática Retorno de erros (validação do modelo) Edição de subartefatos e que compoem um artefato que ira gerar o codigo Ed em tempo real Como testar o código gerado ou o modelo?

Como validar modelos?

Hightlight, auto sugestão e linkagem das palavras -> questoes tecnologicas de implementação

Definir a gramática e as transformações-> selado!

Retorno de erro-> aparentemente nao é complicado

Analisar para que casos é interessante usar o comdd ao inves do uso de uma ide+svn. Pois o estudo de caso mostrou que o comdd adequa as necessidades, contudo, não podemos generalizar tal afirmação. Ainda pode-se a partir da evolução das wikis para esse propósito verificar se realmente há a necessidade do uso dessas ferramentas locais, de merges ou dos modelos de versionamente atual.

5.4 Considerações Finais

5.5

Referências Bibliográficas

FRANCE, R.; RUMPE, B. Model-driven development of complex software: A research roadmap. In: *29th International Conference on Software Engineering 2007 - Future of Software Engineering*, Minneapolis, MN, USA: IEEE Computer Society, 2007, p. 37–54.

JONES, M.; SCAFFIDI, C. Obstacles and opportunities with using visual and domain-specific languages in scientific programming. In: *Visual Languages and Human-Centric Computing (VL/HCC), 2011 IEEE Symposium on*, 2011, p. 9 –16.

PRUDÃªNCIO, J. G.; MURTA, L.; WERNER, C.; CEPÃªDA, R. To lock, or not to lock: That is the question. *Journal of Systems and Software*, v. 85, n. 2, p. 277 – 289, <ce:title>Special issue with selected papers from the 23rd Brazilian Symposium on Software Engineering</ce:title>, 2012.
Disponível em: `http://www.sciencedirect.com/science/article/pii/S0164121211001063`

SARMA, A.; NOROOZI, Z.; HOEK, A. Palantir: raising awareness among configuration management workspaces. In: *Software Engineering, 2003. Proceedings. 25th International Conference on*, 2003, p. 444 – 454.

TEPPOLA, S.; PARVIAINEN, P.; TAKALO, J. Challenges in deployment of model driven development. In: *Software Engineering Advances, 2009. ICSEA '09. Fourth International Conference on*, 2009, p. 15 –20.

ZHANG, J.; RAY, I. Towards secure multi-sited transactional revision control systems. 2007, p. 365 – 375.
Disponível em: `http://www.sciencedirect.com/science/article/pii/S0920548906000742`

CAPÍTULO

6

Apêndice A

Antlr: conceitos e explicação da nossa gramática

CAPÍTULO

7

Apêndice B

CoMDD: como implantar uma dsl em uma wiki?

CAPÍTULO

8

Apêndice C

Eclipse: xtext/xpand