Pontifícia Universidade Católica de Goiás



Felipe Fernandes Almeida Manso

**MDD (Model-Driven Development) e o MDA (Model Driven Architecture)**

Goiânia, 07 de dezembro de 2018.

**Introdução**

A MDD e o MDA enfatizam a função dos modelos como elementos fundadores para implementação, na expectativa de que eles serão mais do que projetos com base em quais desenvolvedores humanos gravarão os códigos, mas eles mesmos são ativáveis ou executáveis em um grau que depende do recurso do conjunto de ferramentas de suporte. Isso segue uma tendência, começada há muito tempo, de aumentar o nível de abstração em que o desenvolvedor humano trabalha. Isso desvia a atenção do código como conhecemos para os modelos expressos em uma linguagem ainda mais alta, talvez gráfica. O RUP, ao identificar determinados artefatos como modelos em vez de documentos (capturando requisitos e design, por exemplo) que contém fotos de modelos, implicitamente suporta essa possibilidade.

**MDD (Model-Driven Development)**

O desenvolvimento orientado a modelos, ou model-driven development (MDD), é um estilo de desenvolvimento de software em que os principais artefactos de software são modelos, a partir dos quais são gerados o código e outros artefactos de acordo com as boas práticas. Um modelo é uma descrição de um sistema a partir de uma determinada perspectiva, omitindo detalhes irrelevantes para que as características de interesse sejam vistas de forma mais clara. No MDD, introduz-se o critério adicional de que um modelo tem que ser compreensível para a máquina. Por exemplo, temos de ser capazes de aceder ao conteúdo do modelo

No MDD, introduz-se o critério adicional de que um modelo tem que ser compreensível para a máquina. Por exemplo, temos de ser capazes de aceder ao conteúdo do modelo de forma automatizada. A possibilidade de compreensão dos modelos pela máquina é um pré-requisito para sermos capazes de gerar artefactos. Um digrama num quadro branco pode satisfazer os outros critérios de um modelo. No entanto, enquanto não o capturarmos de uma forma legível para a máquina, não o poderemos utilizar numa cadeia de ferramentas MDD.

Os modelos de software são tipicamente expressos em UML (Unified Modeling Language). Esta é uma linguagem para a especificação, visualização e documentação de sistemas de software. A UML fornece uma notação visual e semânticas subjacentes para os modelos de software. De igual modo, tem um formato standard de serialização legível para a máquina, permitindo assim a automação.

Os modelos de software escondem os detalhes das implementações técnicas, de modo a que um sistema possa ser desenhado utilizando conceitos a partir do domínio da aplicação. O desenho de aplicações é realizado normalmente utilizando uma ferramenta de modelação UML (como o IBM Rational Software Architect), utilizando conceitos relevantes para o domínio da aplicação.

Por exemplo, quando se trabalha no domínio da integração empresarial, devemos começar por modelar o desenho da aplicação utilizando conceitos como mensagem, proxy e adaptador. Numa fase posterior, podemos refinar o modelo de software e desenhar os detalhes dos seus componentes.

**Vantagens do MDD**

O MDD tem o potencial de melhorar grandemente as práticas correntes de desenvolvimento de software. O quadro que se segue mostra as vantagens de uma abordagem MDD.

|  |  |
| --- | --- |
| Vantagem | Explicação |
| Maior produtividade | Custos reduzidos de desenvolvimento de software, através da geração de código e de artefactos a partir de modelos, aumentando também a produtividade de quem desenvolve. Temos de considerar o custo de desenvolver (ou comprar) transformações, mas um planeamento cuidado ajuda a assegurar que existe uma redução global dos custos. |
| Capacidade de manutenção | Muitos componentes de soluções são implementados utilizando tecnologias de plataformas existentes, relativamente às quais a organização já não possui know-how. O MDD conduz a uma arquitectura fácil de manter, onde as alterações são efectuadas de forma rápida e consistente, permitindo uma migração mais eficiente dos componentes para novas tecnologias.  Os modelos de alto nível são mantidos livre de detalhes de implementação irrelevantes, o que facilita as alterações na tecnologia da plataforma subjacente e na sua arquitectura técnica.Altera-se a arquitectura técnica da implementação através da actualização de uma transformação. A transformação é reaplicada aos modelos originais para produzir artefactos de implementação seguindo a nova abordagem.  Podemos experimentar diferentes ideias antes de tomarmos uma decisão final. As más decisões são facilmente alteradas. Os projectos de software são frequentemente afectados por decisões erradas, mas que são demasiado caras de resolver. |
| Reutilização do existente | Modelação consistente em UML de plataformas existentes. Se existem muitos componentes implementados na mesma plataforma existente, podemos desenvolver transformações inversas a partir dos componentes para a UML. Podemos migrar os componentes para uma nova plataforma, ou criar wrappers para permitir o acesso ao componente existente através de tecnologias de integração como os serviços Web. |
| Adaptabilidade | A adição ou modificação de uma função de negócio é linear, uma vez que já foi feito o investimento em automação. Quando se pretendem adicionar novas funções de negócio, desenvolvemos apenas o comportamento específico dessa capacidade. A restante informação necessária para gerar artefactos de implementação foi capturada nas transformações. |
| Consistência | A aplicação manual de práticas de codificação e de decisões arquitecturais é propícia a erros. O MDD assegura que os artefactos são gerados de forma consistente. |
| Repetibilidade | O MDD é especialmente poderoso quando aplicado a nível de um programa ou de uma organização. O retorno do investimento a partir do desenvolvimento de transformações aumenta de cada vez que são reutilizadas. A utilização de transformação experimentadas e testadas também aumenta a previsibilidade do desenvolvimento de novas funções, além de reduzir o risco, uma vez que os problemas arquitecturais e técnicos já foram resolvidos. |
| Melhor comunicação com os stakeholders | Os modelos omitem detalhes de implementação que não são relevantes para a compreensão do comportamento lógico de um sistema. Estão muito mais próximos do domínio do problema, reduzindo assim o gap semântico entre os conceitos que são compreendidos pelos stakeholders e a linguagem que a solução é expressa. Facilita a disponibilização de soluções que estão melhor alinhadas com os objectivos de negócio. |
| Melhor comunicação do desenho | Os modelos ajudam-nos a compreender os sistemas ao nível do desenho, conduzindo a uma melhoria da discussão e da comunicação sobre o sistema. Uma vez que os modelos fazem parte da definição do sistema, e não da documentação, nunca ficam desactualizados e são fiáveis. |
| Captura do conhecimento | Os projectos ou as organizações dependem frequentemente de especialistas chave que tomam repetidamente as melhores decisões em termos de prática. O seu know-how é capturado em padrões e transformações, pelo que esses especialistas chave não precisam de estar presentes para outros membros do projecto. Além disso, se as transformações forem acompanhadas de documentação suficiente, o conhecimento de uma organização é mantido nos padrões e transformações, mesmo quando os especialistas chave vão embora. |
| Os modelos como activos de longo prazo | Os modelos são activos importantes que capturam aquilo que os sistemas de TI fazem. Os modelos de alto nível são flexíveis face a mudanças ao nível da plataforma estado da arte. Só mudam quando mudam os requisitos de negócio. |
| Capacidade de prorrogar as decisões tecnológicas | O desenvolvimento inicial de aplicações é focalizado nas actividades de modelação. Podemos prorrogar as escolhas de uma plataforma ou produto tecnológico específico até termos disponível mais informação. Em áreas com ciclos de desenvolvimento extremamente longos (como os sistemas de controlo de tráfego aéreo), este aspecto é crucial. As plataformas alvo até podem ainda não existir quando se inicia o desenvolvimento. |

**MDA (Model Driven Architecture) Início da página**

MDA é uma iniciativa do OMG, um consórcio do segmento de mercado com cerca de 800 membros com a missão de estabelecer diretrizes e especificações padrão para fornecer uma estrutura neutra do fornecedor para o desenvolvimento do aplicativo e encorajar a interoperabilidade nas principais plataformas de infra-estrutura de hardware e software. A Unified Modeling Language é um produto do OMG. Agora, o OMG promove a MDA como uma especificação de carro-chefe e com a posição do OMG como o promulgador de padrões práticos com a intenção de serem suportados pelo IC do segmento de mercado, prática, produtos e ferramentas e, observando o sucesso da UML, vale a pena estudar a MDA. Há uma abundância de informações sobre o MDA, incluindo o Guia do MDA mais recente [OMG03], no Web site da OMG. Há também vários livros disponíveis como [FRA03], [KLE03] e [MEL04] e muitos artigos, tais como "An MDA Manifesto" de Grady Booch, Alan Brown, Sridhar Iyengar, James Rumbaugh e Bran Selic no The MDA Journal, maio de 2004.

A MDA (Model Driven Architecture - Arquitetura Orientado por Modelos) propõe separar a lógica de negócio, plataforma e tecnologia, de tal forma que as modificações na plataforma não afetem as aplicações existentes e a lógica de negócio evolua independente da tecnologia. Com isso, os benefícios são evidentes, tais como a redução de custo do ciclo de vida do projeto, a redução do tempo de desenvolvimento para novas aplicações, o aumento da qualidade da aplicação e o aumento do retorno no investimento em tecnologias.

Para obter tais benefícios, a MDA faz uso de modelos, os quais podem ser independentes ou específicos de uma plataforma. Na MDA, o Modelo Independente de Plataforma (PIM - Platform Independent Model) será submetido a processos de transformações a fim de se obter como destino um ou mais Modelos Específicos de Plataforma (PSM - Platform Specific Model). Ao final do processo, os PSM poderão ser transformados em código fonte.

**Idéias Centrais do MDA**

A MDA apresenta alguns conceitos específicos e terminologia que a distingue das abordagens mais gerais para MDD e isso é definido e discutido nas seguintes seções.

Construindo nos Padrões Existentes Início da página

A MDA está apoiada pelos padrões OMG existentes, incluindo:

O MOF (Meta-Object Facility) - Além de definir uma linguagem para a construção dos metamodelos, por exemplo, UML e CWM, o MOF define uma estrutura para implementar os repositórios para os modelos e metamodelos, permitindo uma abordagem consistente para a manipulação desses modelos ao utilizar o MDA. O MOF é uma tecnologia essencial para MDA.

O UML (Unified Modeling Language) - Os PIMs, PMs e PSM s são definidos no UML, o que é a notação fundamental para o MDA.

XMI (XML Metadata Interchange) - XMI define um formato de intercâmbio do modelo UML com base em XML.

O CWM (Common Warehouse Metamodel) - Como o UML é para a modelagem do aplicativo, o CWM é para modelagem de dados.

Independência da Plataforma Inícíio da página

Uma noção intuitiva de uma Definição de termo: plataforma é que é que ela suporta uma camada arquitetural mais alta por meio de provisão de serviços com um conjunto bem definido de interfaces que ocultam os detalhes da implementação. A definição do OMG (no Guia de MDA) da plataforma é:

"Um conjunto de subsistemas/tecnologias que fornece um conjunto coerente de funcionalidade por meio de interfaces e padrões de uso especificado que qualquer subsistema que dependa da plataforma pode utilizar sem se preocupar com os detalhes de como a funcionalidade funcionalidade fornecida pela plataforma é implementada."

Isso é parecido como o conceito de uma Definição de termo: camada utilizando no RUP.

A idéia de independência de plataforma é levemente incerta: é uma qualidade ou característica de um modelo, por exemplo, pode-se dizer que um modelo é independente de uma plataforma específica quando ele não contém qualquer referência aos serviços ou ao recurso fornecidos por essa plataforma. No entanto, a instrução é relativa porque é difícil contemplar uma forma absoluta de independência da plataforma. O Guia de MDA reconhece isso e também admite a possibilidade de graus de independência da plataforma em relação a uma plataforma específica em que, por exemplo, um modelo utiliza uma forma generalizada de um recurso em uma plataforma específica.

A realização da independência da plataforma é assistida pela evolução das plataformas como J2EE, .NET e WebSphere, para aumentar os níveis de abstração em termos do que é exposto para aplicativos. Isso torna a identificação das construções neutras da plataforma muito mais tratáveis e as transformações específicas da plataforma que as convertem muito mais simples e fáceis de gravar.

**Benefícios Putativos**

Portabilidade e Interoperabilidade Início da página

MDA oferece a esperança de controlar o gasto e a perturbação da separação de tecnologia, permitindo um conjunto relativamente estável de PIMs a ser redirecionado para qualquer nova tecnologia que seja exigida. A expectativa é de que, com a crescente aceitação de MDA, os desenvolvedores de nova tecnologia também entregará mapeamentos para que as transformações possam ser feitas rapidamente.

Estendendo os mapeamentos PIM-PSM para duas plataformas, a MDA sugere que 'pontes' possam ser construídas entre os dois PSMs e finalmente entre as duas implementações para que um PIM possa ser distribuído entre as plataformas. A maioria das empresas enfrenta a realidade de desenvolvimento para ambientes heterogêneos com uma combinação de tecnologias novas e antigas, então essa habilidade de realizar interoperabilidade é potencialmente de grande benefício.

Custo de Ciclo de Vida Reduzido Início da página

Produtividade Início da página

O foco do desenvolvimento com MDA se torna o PIM mais abstrato. Com um poderoso conjunto de ferramentas para gerar um PSM (ou um código), como um ambiente deve ser mais produtivo do mesmo modo que trabalhar em uma linguagem de alto nível é mais produtivo que trabalhar no montador, considerando especificamente que um PIM, ou algo assim, é freqüentemente desenvolvido mesmo assim, por exemplo, servindo como o Modelo de Design no RUP. O ganho de produtividade depende criticamente de quanta intervenção manual é necessária no processo de transformação.

Qualidade Início da página

Idealmente, em MDA, o destino de manutenção é o PIM. Conseqüentemente, com a condição de que o PIM seja bem arquitetado, deve haver menos defeitos na vida do produto porque há menos oportunidades para um humano injetá-los e os defeitos que são descobertos deveriam ser mais baratos para corrigir, por meio do benefício da transformação automatizada. A concentração no PIM também está mais alinhada com as preocupações do domínio, então deveria haver uma probabilidade maior de satisfazer as necessidades dos usuários.

**Principais Benefícios**

Produtividade – embora o uso da arquitetura MDA implique na geração das regras de transformação entre os modelos PIM, PSM e de código, essa geração deve ser realizada apenas uma vez, para cada plataforma. Uma vez definidas as regras, as transformações se tornam automáticas, e apenas o modelo conceitual (PIM) deveria ser atualizado, enquanto que os demais modelos, inclusive a implementação, seriam automaticamente atualizados, reduzindo o tempo de desenvolvimento e aumentando a qualidade do artefato final.

Portabilidade – uma vez que o PIM é, por definição, independente de plataforma, o mesmo PIM pode ser transformado automaticamente para vários PSMs, específicos para as diferentes plataformas.

Interoperabilidade – a interoperabilidade em MDA é alcançada através de bridges, que realizam a comunicação entre PSMs e códigos de diferentes plataformas. Para obter interoperabilidade as ferramentas de transformação devem gerar não só os PSMs, mas também as bridges entre eles.

Manutenção e Documentação – todo o trabalho de manutenção não é mais feito no código, mas sim no nível mais alto de abstração (o modelo PIM). Desta forma, a manutenção é facilitada e a documentação constantemente atualizada. E, no caso da manutenção ser realizada diretamente no PSM, deve ser possível atualizar automaticamente o PIM e manter os modelos consistentes.

Apesar de MDA vislumbrar tais benefícios ela ainda não se encontra totalmente madura. Para obter benefícios na sua totalidade, todas as transformações entre modelos devem ser automatizadas, o que o atual estado da tecnologia ainda não permite plenamente. Ainda há muitas questões a serem resolvidas e as ferramentas disponíveis atualmente não suportam toda a especificação MDA. Entretanto, MDA é flexível a ponto de permitir que uma ferramenta transforme um PIM diretamente em código, sem produzir um PSM.

Bibliografia

https://pt.scribd.com/document/297640823/Apresentacao-Desenvolvimento-Dirigido-Em-Modelos

https://homepages.dcc.ufmg.br/~figueiredo/disciplinas/2012a/reuso/reuso-aula20.pdf

http://walderson.com/IBM/RUP7/LargeProjects/tech.rsa/guidances/concepts/mdd\_and\_mda\_8F9B3685.html#Top

http://www.sinfic.pt/SinficNewsletter//sinfic/Newsletter54/Dossier4.html

http://www.sinfic.pt/SinficNewsletter/sinfic/Newsletter54/index54.html

<https://www.devmedia.com.br/artigo-engenharia-de-software-9-mda-arquitetura-orientada-por-modelos/11604>

<https://www.dimap.ufrn.br/~thais/Arquitetura/MDA.pdf>