

# MDA: Fazendo a UML Valer a Pena

*Por Walter Itamar Mourão*

Há mais de vinte anos acompanho a luta da documentação técnica de sistemas, ou melhor: a luta pela boa documentação técnica de sistemas. De um lado, consultores e acadêmicos mostram por "a+b" o valor da documentação no processo de desenvolvimento de software. Do outro, gerentes de projeto e desenvolvedores que geralmente não têm tempo para criar a documentação técnica nos padrões adequados e no tempo certo.

Considero que os principais motivos que impediam a ampla adoção da documentação como parte essencial de projeto de software eram a falta de um formato padronizado para a criação dessa documentação e uma grande resistência por parte da equipe técnica em acoplar ao seu projeto uma atividade que não traria benefícios visíveis e diretos ao produto final ou ao processo de desenvolvimento.

## Documentação técnica

Quando falo em documentação técnica estou me referindo a todo e qualquer artefato que seja produzido durante o processo de desenvolvimento que tenha um caráter informativo ou de comunicação entre as partes envolvidas no projeto. Especificações, definições de caso de uso, documentação de fontes (javadoc), diagramas são exemplos de documentação técnica.

## A UML

A UML define formalmente um conjunto de elementos para a descrição de um sistema computacional e veio preencher uma lacuna de padronização e formalidade na descrição de um programa de computador, rapidamente se tornando o padrão de fato na descrição em alto nível de sistemas computacionais de qualquer complexidade. Atualmente a UML aborda inclusive as áreas periféricas ao software propriamente dito, como definição de requisitos, ambiente de implantação e fluxo de processos genéricos.

Apesar dessa adoção ampla, pode-se dizer que ela ocorreu "de cima para baixo" uma vez que um grande número de desenvolvedores e gerentes de projeto ainda resiste a UML, ou melhor: resistem à inclusão do processo de documentação no seu processo de desenvolvimento.

De fato ainda é muito comum vermos equipes cuja documentação dos sistemas se resume aos artefatos de "baixo nível" tais como códigos fonte, scripts de banco e às vezes diagramas de entidade e relacionamento, inteligíveis somente aos participantes da equipe técnica.

Entre as equipes que desenvolvem uma documentação adequada também é comum a criação de documentos "por força de contrato" em caso de software houses ou mesmo do padrão de trabalho da empresa. Nesses casos raramente a documentação é disponibilizada como fonte de referência para o suporte operacional ou consultada e atualizada adequadamente durante uma manutenção.

Apesar de ser um avanço frente às equipes que simplesmente não documentam, o ideal é que a documentação técnica seja parte viva do ciclo do software e que os artefatos da documentação sejam um elemento ativo e facilitador na comunicação interna e externa da equipe, vindo a ser tão valorizado quanto o código fonte do programa em desenvolvimento.

Especificamente em relação à UML resiste ainda o argumento, relativamente justificado, do custo de adoção de uma nova tecnologia "só para melhorar a documentação técnica". A MDA reverte essa situação colocando os modelos no centro do processo de desenvolvimento, agregando à

documentação técnica o papel de participante ativo na fase de programação, derrubando por mérito as resistências da equipe técnica.

## **Modelos**

Modelos são reproduções simplificadas e contextualizadas, embora fiéis, do mundo real. Em uma comparação entre engenharias, pode-se dizer que o desenho arquitetônico de um prédio é um modelo, assim como seu projeto hidráulico e elétrico também o são. Da mesma forma, diagramas de entidades e relacionamentos, fluxogramas e diagramas de classes são modelos de um sistema informatizado utilizados largamente pela engenharia de software.

A UML prevê uma série de diagramas e elementos para a modelagem de sistemas informatizados. Define ainda extensões e especializações, expondo um caráter flexível e adaptável sem a perda de formalização. Esse caráter adaptável do UML foi, e às vezes ainda é, motivo de muita confusão por parte de profissionais que imaginam o UML como uma “metodologia” estrita onde todos os diagramas devem ser desenvolvidos com regras e sintaxe rígidas.

Ao contrário disso, a UML se propõe exatamente ao que o nome indica: uma linguagem formal onde cada equipe de desenvolvimento pode definir como e quando vai usá-la, de acordo com sua própria metodologia.

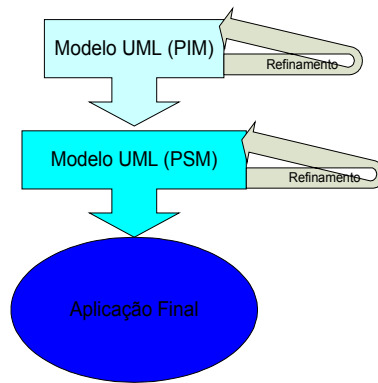
Uma das grandes vantagens de se usar modelos formais é a facilitação da comunicação entre os participantes da equipe de desenvolvimento e com o próprio usuário e/ou analista de negócios, uma vez que vários dos diagramas previstos no UML são compreensíveis por pessoas com pouca ou nenhuma qualificação técnica, a partir do conhecimento básico dos elementos de alto nível.

## **MDA**

A MDA (assim como a UML) é um padrão aberto regido pelo OMG (Object Management Group) e surgiu como uma evolução natural do uso de modelos de software no que diz respeito a usar os modelos não apenas como ferramentas de comunicação e documentação, mas sim como elementos ativos e participantes do processo de desenvolvimento como um todo. A MDA prevê que os modelos devem ser usados na geração de programas, scripts de banco, documentação de usuário, configurações e quaisquer outros elementos que façam parte do processo de desenvolvimento.

Apesar de ser considerada uma disciplina independente do UML e de prever que quaisquer tipos de modelos formais podem ser usados, na prática o que se vê como padrão é a utilização do UML associado à MDA.

A MDA dita que os artefatos a serem gerados durante o desenvolvimento devem ser principalmente o resultado de transformações que os modelos sofrem a partir de um estado de alto nível, preferencialmente independente da plataforma de implementação (PIM – Platform Independent Model) e passa por estágios de mais baixo nível onde são criados modelos específicos para a plataforma alvo (PSM – Platform Specific Model) até chegar aos artefatos finais. Na prática, a tendência das ferramentas MDA é nunca expor as transformações intermediárias (PSMs) dos modelos ao desenvolvedor, que só tomará conhecimento do modelo original e dos artefatos finais criados.



*Ilustração 1: A MDA age através de transformações do modelo*

## Qualidade de software

No aspecto da qualidade do software final posso destacar algumas características que diferenciam a MDA.

### **Rastreabilidade**

Todo software está sujeito a algum tipo de manutenção durante todo seu ciclo de vida. É importante então que todos os artefatos finais que implementam qualquer dos requisitos funcionais sejam facilmente localizáveis. Essa característica é chamada de “rastreabilidade dos requisitos do software”. A rastreabilidade pode ser no sentido "para baixo", onde se deseja localizar a implementação de um determinado requisito, ou "para cima", onde se deseja localizar o requisito que deu origem a uma determinada implementação.

Uma vez que na MDA grande parte dos artefatos do processo de desenvolvimento é criado como resultado direto das transformações do modelo, cria-se um vínculo entre os vários elementos do modelo e seus equivalentes nos artefatos permitindo a localização fácil desses elementos em qualquer sentido, o que permite um altíssimo grau de rastreabilidade.

### **Redução do número de erros humanos de implementação**

Na MDA, praticamente todo código fonte é gerado e/ou vinculado a este código, implicando em uma diminuição sensível no número de erros de implementação. Adicionalmente, as metodologias que valorizam as diversas fases de testes podem se beneficiar da capacidade da MDA na geração de scripts de criação de bases para testes, scripts para ferramentas automatizadas e geração de fontes para testes unitários, entre outros.

### **Integridade da documentação**

Uma vez que na MDA praticamente toda manutenção começa pela alteração do modelo, garante-se a integridade e a aderência do modelo e dos outros artefatos durante as manutenções que se façam necessárias.

### **Padronização**

A questão da padronização da codificação dos fontes pode parecer uma questão menor à primeira vista. No entanto, esta dimensão muda quando se percebe que a MDA permite a padronização de todos os artefatos gerados, incluindo scripts de banco e suas particularidades, scripts de criação de executáveis, pastas, código fonte, documentos e nomenclaturas em geral. Adicionalmente, todos estes artefatos servem como exemplo para a criação e configuração dos artefatos que porventura não sejam gerados.

## **Representabilidade**

Aderência e representabilidade são conceitos similares. Enquanto a primeira define o grau de similaridade entre o modelo e sua representação real, a segunda refere-se à capacidade de entendimento do problema por todos os participantes do projeto. É muito comum, no momento das explicações dos técnicos a equipes não técnicas, ouvir a frase “corte a parte técnica. Não a entendemos, é muito complexa”. Na verdade, a frase correta é “Toda essa técnica não nos diz nada, não representa nada do que conhecemos ou nos importamos”.

A arma mais eficaz contra o problema da falta de representabilidade é a junção UML, que permite uma descrição de mais alto nível e mais representativa e a MDA, que viabiliza e reforça o uso ótimo da UML, além de garantir a integridade dos documentos gerados (como exposto na seção acima).

## **A MDA hoje**

Todos os grandes fornecedores de ferramentas de desenvolvimento de software (proprietários e open source) estão incorporando a MDA às suas ferramentas, seja na forma de iniciativas localizadas e vinculadas a aspectos específicos da UML ou como centro do processo de desenvolvimento que essas ferramentas suportam.

No cenário open source ocorreu uma grande evolução nos últimos quatro anos com o surgimento e amadurecimento de várias soluções de alta qualidade, tais como o AndroMDA, o openMDX e o MetaBoss.

## **Conclusão**

Apesar de existirem iniciativas bem sucedidas de geração de código já há vários anos a MDA representa uma abordagem mais completa e consistente do que todas as anteriores, até mesmo porque sua atuação não é restrita ao código fonte propriamente dito. De fato, com a MDA, praticamente todos os artefatos resultantes do processo de desenvolvimento se beneficiam em algum grau de sua capacidade de automatização.

Adotar a UML e a MDA de forma consciente e madura significa dar um passo importante em direção à criação de aplicações mais completas, com mais qualidade e com menos esforço.