UMA ANÁLISE SOBRE O DESENVOLVIMENTO DIRIGIDO A MODELOS E SUA CONTRIBUIÇÃO NA REALIZAÇÃO DE TESTES DE SOFTWARE

Arian Dantas Meneses*

Resumo – *O desenvolvimento de software dirigido a modelos tem o intuito de diminuir problemas relacionados a custo e qualidade, uma vez que os gastos com codificação são reduzidos e o software produzido possui uma estrutura bem definida de qualidade. Alinhado a esse processo, os testes utilizando modelos contribuem para efetividade e produtividade. Neste artigo foi realizada uma análise de como esses processos funcionam e apresentar os pontos positivos propiciados pela utilização de ambos.*
Palavras-chave – *Desenvolvimento de Software; Modelos; MDD; MDT.*

Abstract – *Model-Driven Development has the intention to reduce some problems related to cost and quality, once spending with coding is reduced and the produced software has a well-defined structure of quality. Aligned to this process, tests using models contribute to effectiveness and productivity. This paper has the aims to do a analysis of how these process work and show the positive points coming from the use of both.*
Keywords – *Software Development; Models; MDD; MDT.*

1. INTRODUÇÃO

O processo de desenvolvimento de software vem evoluindo e passando por mudanças (ALVES et al., 2008). Novos processos surgem constantemente. Porém eles não têm conseguido acompanhar as expectativas de clientes e usuários, em se tratando de qualidade de software (ALMEIDA, 2014). Segundo Pressman (2011), apenas possuir um processo de desenvolvimento bem definido não garante que o software será de qualidade. Por esse motivo, Tankovic et al. (2012) diz que a modelagem de software tende a ser a "próxima geração de metodologia de desenvolvimento de software".

Com a utilização de modelos de software, as construções de especificações se tornam mais simples e mais fáceis de serem entendidas (HERRERA et al., 2012). São nesses quesitos que o desenvolvimento dirigido a modelos começou a ganhar forma, fazendo com que os modelos que antes eram utilizados apenas em esboços e diagramas de software passassem a ser também legíveis por máquina, para geração de artefatos e código (SWITHINBANK et al., 2005). De acordo com os autores, o desenvolvimento dirigido a modelos, ou MDD como é conhecido, "tem o

*Bacharel em Sistemas de Informação pela Universidade Federal de Sergipe - UFS

potencial de uma grande redução no custo de desenvolvimento de uma solução e melhorar a consistência e qualidade dessas soluções".

Utilizar o MDD não isenta o software construído de possuir erros. Para ser confiável, um sistema precisa passar por um estruturado processo de testes (ALVES et al., 2008). Assim, começou-se a pensar em também utilizar os modelos para geração dos artefatos de teste. O desenvolvimento dirigido a testes ganhou força nesse momento, proporcionando benefícios como geração automática de testes e execução completa dos casos de teste (REID, 2006).

Este trabalho objetiva analisar esses assuntos abordados e está organizado da seguinte maneira: A seção 2 descreve o desenvolvimento dirigido a modelos, a seção 3 traz uma breve descrição sobre testes de software e a seção 4 representa uma análise sobre importância do MDD na realização de testes. Por fim, a seção 5 apresenta as conclusões.

2. DESENVOLVIMENTO DIRIGIDO A MODELOS (MDD)

O desenvolvimento dirigido a modelos (*Model-Driven Development* – MDD) é um processo de desenvolvimento de software com foco em modelos, capaz de geração de código automática (DEMIR, 2006). Para Swithinbank et al. (2005), um modelo é a representação de um sistema a partir de uma perspectiva própria, de modo que suas características sejam percebidas com clareza. Por meio dos modelos criados, é possível ignorar conceitos supérfluos e focar no conteúdo específico do que deseja ser modelado. Segundo Mellor et al. (2003), "um modelo é um conjunto coerente de elementos formais descrevendo alguma coisa (...) construído com a finalidade de ser passível a uma forma particular de análise". Para os autores, é importante a abstração de linguagens de programação de alto nível para linguagens de modelagem. A Figura 1 demonstra que devemos utilizar um assunto abstrato em um modelo para transformá-lo em uma implementação concreta, de baixo nível.

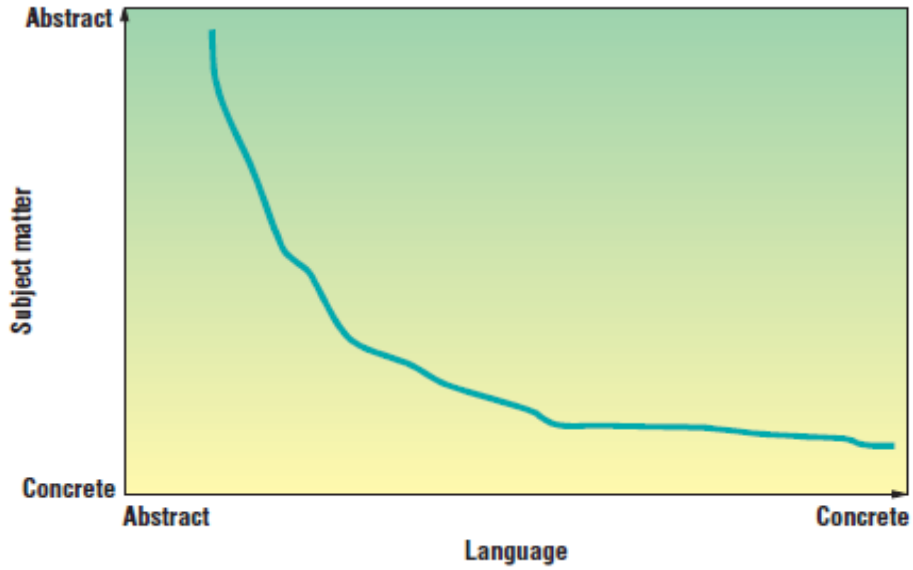

Figura 1 – Abstração de Modelo X Linguagem de Programação (Fonte: MELLOR et al., 2003).

Os modelos atuais são utilizados apenas para especificação de um software, ou seja, só possuem a utilidade de documentação (DEMIR, 2006). Diferente disso, Lucrédio (2007) *apud* Almeida (2014) acredita que os modelos representam mais do que o papel, eles fazem parte do software. Assim, France e Rumpe (2007) classificam os modelos de duas formas:

- Modelos de Desenvolvimento: nesses modelos de software o nível de abstração está acima do nível de código.
- Modelos de Tempo de Execução: esses modelos apresentam uma visão de alguns aspectos de um sistema de execução.

Almeida (2014) resume que através de um modelo efetivo, é possível tornar eficaz todo o processo de desenvolvimento. É nesse ponto que o MDD se encaixa.

Segundo Demir (2006), utilizando o MDD o desenvolvimento de software se torna mais fácil e a qualidade de software aumenta, já que o código é produzido a partir de uma estrutura bem definida. Como o foco passa a ser o modelo, e não mais o código, os desenvolvedores se concentram na construção de um modelo passível a gerar o código de forma automática, ou até mesmo outros modelos (ALVES et al., 2008). É o que pode ser notado na Figura 2, em que existe a combinação dos principais elementos do MDD.

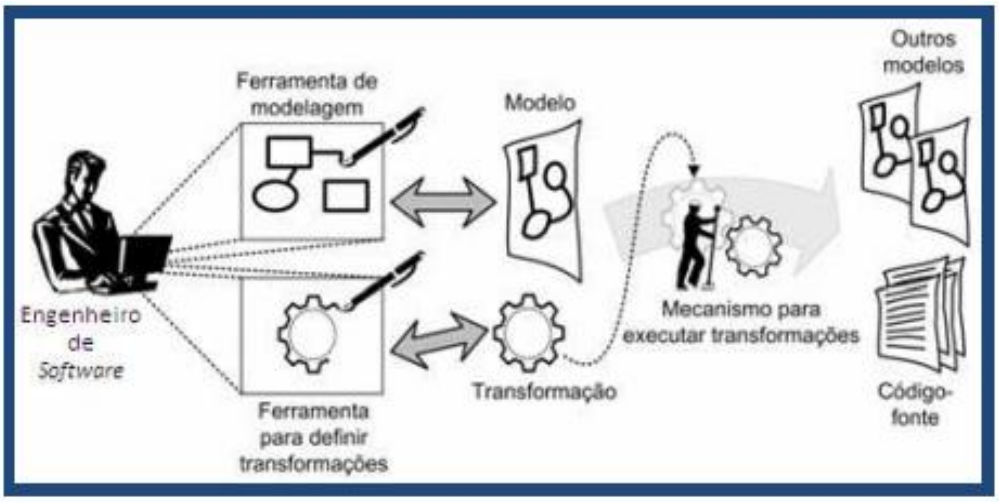

Figura 2 – Principais elementos do MDD (Fonte: Lucrédio, 2007 *apud* Almeida, 2014)

Além de aumentar a qualidade, outro fator importante é com relação ao custo. Mellor et al. (2003) fala que é mais barato construir um modelo utilizando os diagramas da UML do que codificar utilizando alguma linguagem de programação, como Java. É uma situação parecida ao que ocorreu nos anos 60, quando se começou a utilizar as linguagens procedurais em vez de utilizar a linguagem assembly (MACDONALD et al., 2005).

Para MacDonald et al. (2005), o desenvolvimento dirigido a modelos é uma extensão da arquitetura dirigida a modelos (*Model-Driven Architecture* - MDA), desenvolvida pela *Object Management Group* - OMG. Na MDA, existe uma separação entre o negócio e a lógica de aplicação da plataforma de tecnologia (OMG, 2014). É o mesmo que ocorre no MDD, no qual é mais importante resolver o problema do que se preocupar com detalhes de implementação. Alves et al. (2008) cita que tanto o MDD quanto o MDA também englobam os conceitos de portabilidade. Para os autores, com o modelo inicial de um software que não possua uma dependência de plataforma e com as regras de transformação adequada, o sistema poderá ser transformado automaticamente de uma plataforma para outra e posteriormente para o código final. Com essas mudanças de plataformas, apenas os PSM (*Platform Specifc Model*), que são modelos padronizados na MDA, sofreriam alterações, já que possuem algumas características de implementação (OMG, 2014). Assim, uma organização pode permanecer com uma mesma arquitetura de software por vários anos, sem apresentar problemas com mudanças de tecnologia.

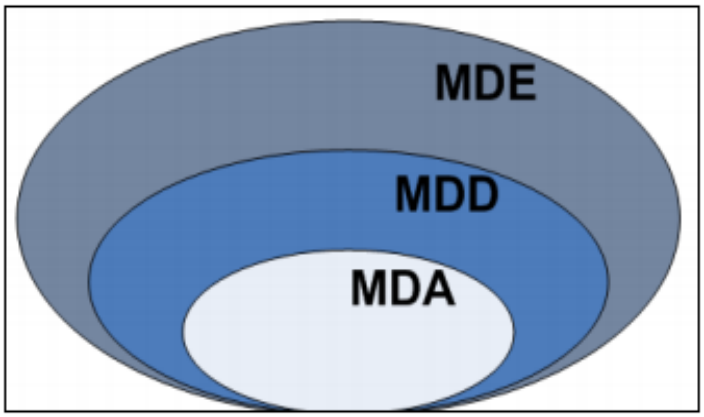

Figura 3 – Visão de MDD. (Fonte: DEMIR, 2006).

A Figura 3 ilustra o conceito de o MDD ser uma extensão do MDA. Ela também apresenta o MDE (*Model-Driven Engineering*). O MDE, ou Engenharia Dirigida a Modelos, é utilizado para descrever as abordagens de desenvolvimento de software em que modelos de software abstratos são transformados em implementação (FRANCE, RUMPE, 2007). No conceito de Ameller (2009) *apud* Almeida (2014) “a MDE é um superconjunto do MDD, pelo fato de englobar outras tarefas dirigidas por modelos do processo completo de Engenharia de Software”.

Em suma, os benefícios da utilização do MDD são apresentados por MacDonald et al. (2005) dessa forma:

- Rápido desenvolvimento com menos bugs;
- Soluções independentes de plataforma;
- Geração automática de código;
- Execução de modelos para encontrar bug mais cedo;
- Comunicação facilitada do design;
- Manutenção aperfeiçoada.

3. O PROCESSO DE TESTES DE SOFTWARE

Sistemas que são desenvolvidos utilizando o MDD não estão isentos de erros, eles também devem ser testados a fim de garantir a qualidade do software (ALVES et al., 2008). Segundo Pressman (2011) “o teste proporciona o último elemento a partir do qual a qualidade pode ser estimada e, mais pragmaticamente, os erros podem ser descobertos”. Ele também pode ser considerado como uma das técnicas

de V&V (Verificação e Validação) mais difundidas, já que verifica se o software foi feito corretamente como desejado pelo cliente (ALMEIDA, 2014).

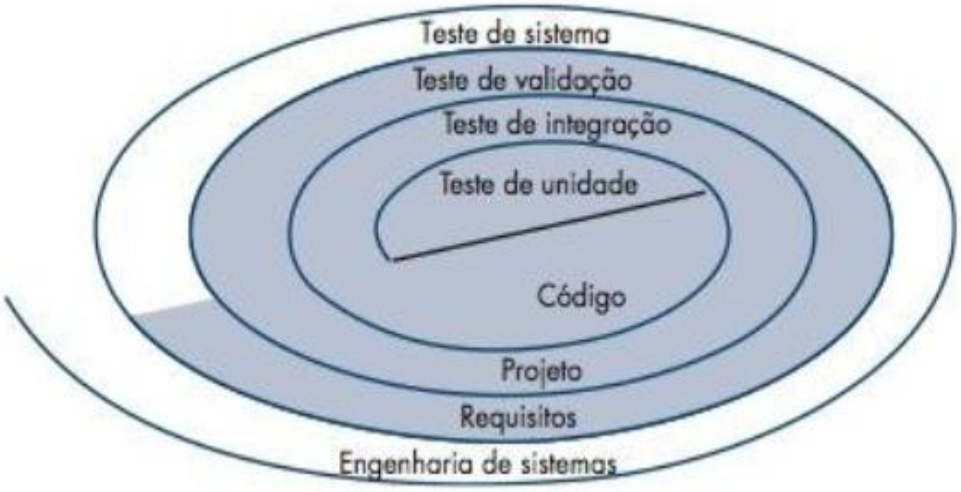

Figura 4 – Estratégia de Testes (Fonte: PRESSMAN, 2011).

A Figura 4 mostra uma estratégia para realização de testes apresentada por Pressman (2011). Nela os testes começam pelo teste de unidade, presente no interior da espiral, tendo por base o código fonte. O teste seguinte, de integração, já tem um foco no projeto do software. Movendo-se um pouco mais para o exterior da espiral, existe o teste de validação. Nesse teste, os requisitos anteriormente estabelecidos são avaliados e validados de acordo com o software desenvolvido. Por fim, é realizado o teste de sistemas, em que o software é testado como um todo, com tudo que lhe envolve. Essa abordagem pode ser visualizada de outra forma, constituída de quatros etapas que são executadas sequencialmente, conforme a Figura 5.

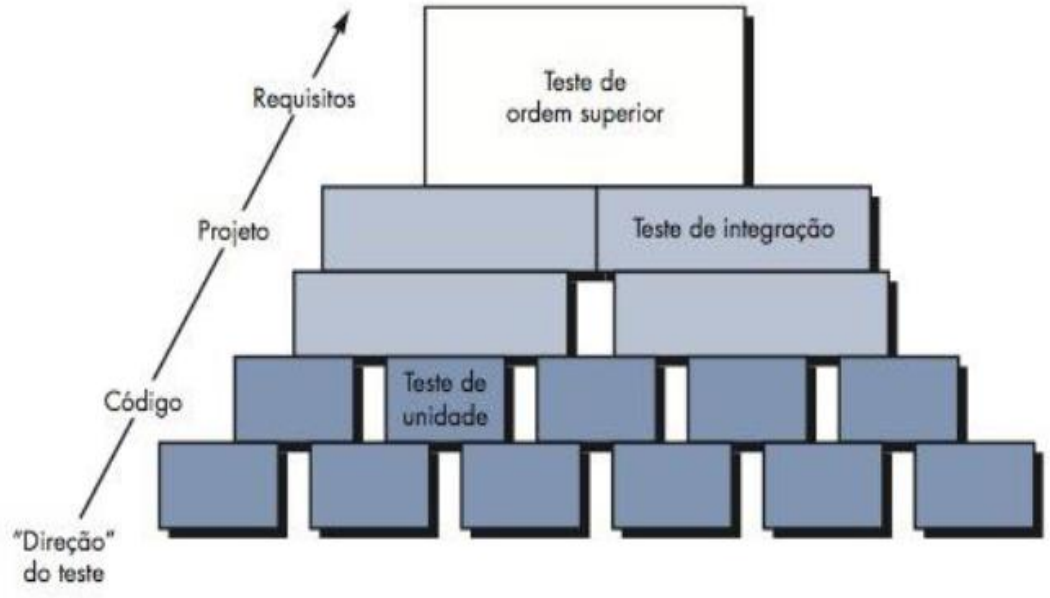

Figura 5 – Etapas de Teste de Software (Fonte: PRESSMAN, 2011).

4. MDD E SUA PARTICIPAÇÃO NOS TESTES DE SOFTWARE

Alguns problemas podem ocorrer na realização de testes sobre códigos gerados automaticamente. Um dos fatores é que a complexidade dos softwares aumentou exponencialmente com o passar dos anos (ALMEIDA, 2014). Já Mellor et al. (2003) cita que um dos obstáculos na transformação de um modelo para o código de implementação é que a linguagem utilizada na modelagem muitas vezes é mal definida, o que ocasiona alguns erros no código. Pensando nesse assunto, Baker et al. (2007) propôs o teste dirigido a modelo (*Model-Driven Testing* – MDT) com práticas do MDD, no qual é gerado todos os artefatos para realização dos testes e esses testes são realizados de forma automática fazendo uso das regras de transformação de modelos. Alves et al. (2008) fala que “esses artefatos gerados poderão servir para a execução em diferentes plataformas”.

O MDT foi proposto com base no MBT (*Model-Based Testing*), abordagem que começou a ser utilizada com o intuito de tornar o teste de software menos custoso (BAKER et al., 2007). Pode-se perguntar: “Por que não ficar utilizando o MBT?”. A resposta pode ser formulada com base nos argumentos apresentados por Almeida (2014). O autor comenta que “os casos de teste são derivados a partir de modelos essencialmente relacionados à especificação funcional do software”. Dessa forma, o mesmo autor diz que por tal motivo existe grande resistência por parte dos

testadores de software, devido à necessidade de aperfeiçoamento na construção de modelos e também por não formalizar a construção da infraestrutura de testes, já que todo o foco fica centrado na geração de casos de teste. Por existir esses obstáculos, Alves et al. (2008) comenta:

> Dentre as vantagens da utilização do MDT em relação à MBT é que, em MBT, os casos de teste são derivados a partir dos modelos de desenvolvimento fracamente conectados, onde estes são normalmente incompletos de informações necessárias para os testes (...). Com a utilização das práticas de MDD, onde os modelos são o centro do desenvolvimento, tais informações poderão ser naturalmente incorporadas.

É com essa ideia que o MDT, com sua automatização tanto na criação, como também na análise e localização de problemas no software, tem se tornado uma das principais abordagens utilizadas em testes de software (Jiao et al., 2006 *apud* Almeida, 2014). Essas fases são estruturadas por Reid (2006) e detalhadas por Almeida (2014) conforme a Figura 6.

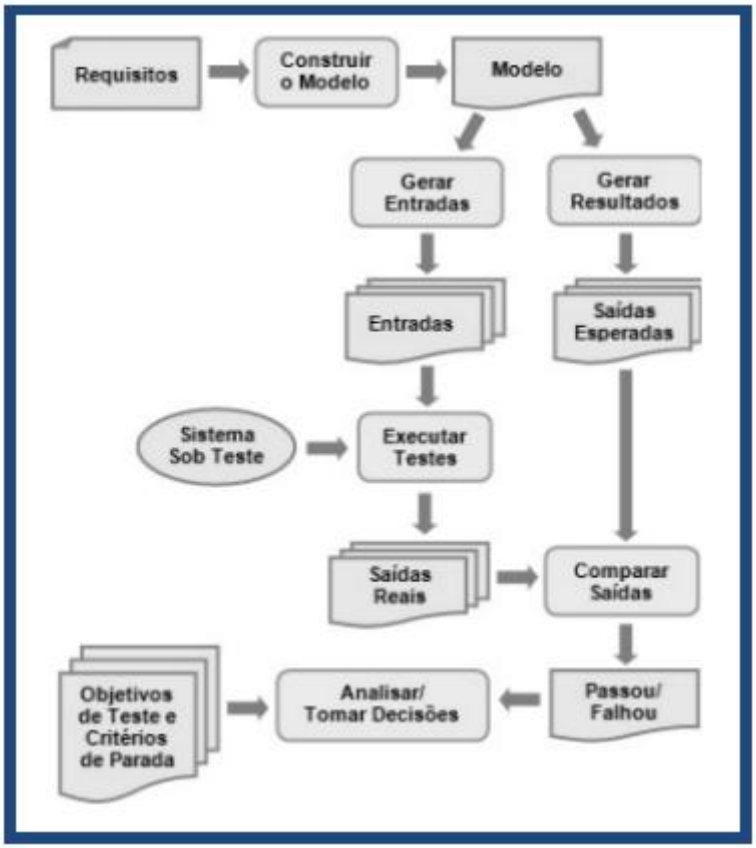

Figura 6 – Etapas do MDT. (Fonte: Maciel, 2010 *apud* Almeida, 2014)

Percebe-se na Figura 6 que a primeira fase do processo consiste nos requisitos. Com esses requisitos definidos é construído o modelo inicial. Com esse modelo em mãos, são gerados as entradas e os resultados. As entradas são testadas com intuito de verificar o comportamento do sistema. Os resultados desses testes são comparados com os resultados esperados, com a finalidade de ter uma resposta sobre se o resultado foi consistente. A partir desses resultados, é feita uma análise para ter uma base para futuras decisões a serem tomadas.

Com as características que possuem, a integração entre o MDD e o MDT pode apresentar vários benefícios para o processo de desenvolvimento de software. Alves et al. (2008) cita alguns benefícios, como:

- Menor custo no processo de desenvolvimento de aplicações e dos casos de testes a serem realizados, independente de qual plataforma esteja sendo utilizada;
- O processo de desenvolvimento e testes apresentará altos níveis de automação;
- Novos requisitos podem ser incorporados mais facilmente, assim como correção dos já existentes;
- Manutenção do software terá seu custo reduzido.

Porém, algumas questões devem ser consideradas antes de qualquer coisa. É extremamente necessário ter bastante atenção ao integrar o MDD e o MDT, verificando as exigências que cada processo apresenta. É preciso realizar um alinhamento das características principais de cada um (ALVES et al., 2008). Sem esse alinhamento, provavelmente o resultado não será o que era esperado inicialmente, transformando os benefícios de ambas as abordagens em prejuízos consideráveis.

5. CONCLUSÃO

Este trabalho teve o objetivo de abordar sobre o desenvolvimento dirigido a modelos, uma metodologia que tem ganhado espaço no âmbito do processo de desenvolvimento de software. Os benefícios provenientes do MDD têm se mostrado convincentes e tendem a difundir sua utilização nós próximos anos. Para isso, alguns obstáculos deverão ser ultrapassados, como normalmente ocorre com uma metodologia nova que aparece no mercado.

A utilização do MDD também tem sua importância na realização dos testes de software. Ter modelos bem definidos e estruturados capazes de gerar automaticamente os artefatos necessários à realização dos testes de software impulsiona a qualidade do mesmo, deixando-o mais coeso. É com essa automatização que o MDT tem se expandido e vem sendo utilizado com sucesso em diversos projetos. Como tanto o MDD como o MDT tem seu foco nos modelos, desenvolvedores e testadores tendem a ser especializar em uma linguagem de modelagem, como a UML e, por ser uma abordagem utilizada por todos os envolvidos no processo de desenvolvimento de software, a integração e comunicação tende a ser facilitada.

Melhorar todo o processo de desenvolvimento de software, com toda a criticidade exigida pelo mesmo atualmente, é essencial. E o MDD e MDT têm se aprimorado e mostrado que são alternativas que devem ser consideradas para quem quer um foco maior na qualidade com custos reduzidos.

REFERÊNCIAS

ALMEIDA, Carla Cássia de Jesus. **Qualitas: Um Modelo de Processo de Desenvolvimento de Software Orientado a Modelos**. Aracaju, 2014. Tese de Mestrado. Disponível em: bdtd.ufs.br/tde_busca/arquivo.php?codArquivo=1644. Acesso em: outubro de 2014.

ALVES, Everton L.G., MACHADO, Patrícia D.L., RAMALHO, Franklin. **Uma abordagem integrada para desenvolvimento e teste dirigido por modelos**. 2nd Brazilian Workshop on Systematic and Automated Software Testing, 2008.

BAKER, P., DAI, Z. R., GRABOWSKI, J., HAUGEN, O., SCHIEFERDECKER, I., WILLIAMS, C.. **Model-Driven Testing: Using the UML Testing Profile**. 2007.

DEMIR, Ahmet. **Comparison of Model-Driven Architecture and Software Factories in the Context of Model-Driven Development**. Proceedings of the Fourth Workshop on Model-Based Development of Computer-Based Systems and Third International Workshop on Model-Based Methodologies for Pervasive and Embedded Software, 2006.

HERRERA, F., PEÑIL, P., POSADAS, H., VILLAR, E.. **A Model-Driven Methodology for the Development of Systemc Executable Environments**. Specification and Design Languages (FDL), 2012 Forum on. 2012.

FRANCE, R., RUMPE, B.. **Model-driven Development of Complex Software: A Research Roadmap**. 2007. Disponível em: http://www.cs.colostate.edu/~france/publications/francer-MDD.pdf. Acesso em: outubro de 2014.

MACDONALD, Anthony; RUSSEL, Danny; ATCHISON, Brenton. **Model-Driven Development within a Legacy System: An industry experience report**. Proceedings of the 2005 Australian Software Engineering Conference. 2005.

MELLOR, Stephen, J.; CLARK, Anthony N.; FUTAGAMI, Takao. **Model-Driven Development**. Published by the IEEE Computer Society, 2003.

OMG (Object Management Group). **Mda – Executive Overview**. Disponível em: http://www.omg.org/mda/executive_overview.htm. Acesso em: outubro de 2014

PRESSMAN, R.S. **Engenharia de Software**. 7ª Edição, Porto Alegre, 2011.

SWITHINBANK, P., CHESSELL, M., GARDNER, T., GRIFFIN, C., MAN, J., WYLIE, H., YUSUF, L.. **Patterns: Model-Driven Development Using IBM Rational Software Architect**. 1ª Edição, 2005.

REID, Stuart. **Model-Based Testing**. Proceedings of Test Automation Workshop. 2006.

TANKOVIC, Nikola, VUKOVITC, Drazen, ZAGAR, Mario. **Rethinking Model Driven Development: Analysis and Opportunities**. Proceedins of the ITI 2012 34th Int. Conf. on Information Technology Interfaces. 2012.