Relatório Parcial da USP-SC - Autor: Paulo Masiero

Genarch

É uma ferramenta desenvolvida no ambiente Eclipse que usa vários outras ferramentas do tipo "plug-in" disponíveis nesse ambiente. É composta por um modelo de features simples (obrigatórias, opcionais e alternativas); por um modelo de arquitetura, que é composto por componentes – classes em Java, encapsuladas em um componente –; e um modelo de configuração, que indica quais componentes (ou classes) implementam as features. As classes do modelo arquitetural podem ser generalizadas por meio de gabaritos (templates) usando a linguagem XPand do plugin oAW. Ou então podem ser codificadas de forma fixa. Além disso, usa também anotações no código. A partir desses três modelos pode gerar o código fonte de uma aplicação, que depois deverá ser compilada. Se o interior dos métodos não for codificado, gerará um esqueleto de código. Permite tanto a instanciação de frameworks como a derivação de membros de uma família de programas. Na versão atual (artigo de 2008) não usa aspectos para apoiar a derivação, mas isso está nos planos futuros. O trabalho não considera um processo de desenvolvimento de software, como por exemplo, engenharia de requisitos, análise de requisitos e partes do projeto de software, que levariam aos três modelos que permitem gerar código.

CrossMDA

É uma abordagem de desenvolvimento baseada em modelos que permite a combinação automática de modelos de negócio (ou base) com modelos de interesses transversais. Os modelos combinados são basicamente modelos de classes da UML, que estão na forma de PIM (Platform Independent Model) e depois são transformados em um modelo dependente de plataforma (MDA). O modelo gerado pode ser compilado e executado se o modelador tiver incluído o código dos métodos e adendos, senão será um esqueleto de código. As ferramentas trabalham com o modelo especificado na linguagem XMI. O processo é composto de quatro fases: modelagem (modelo de negócios e de aspectos), seleção dos modelos fonte, mapeamento e composição. As classes do modelo base e os aspectos são estendidos com estereótipos de um perfil UML definido pelos autores, que combinam propostas de dois outros perfis: de Stein e de Camargo e Masiero. A composição dos modelos é feita por um programa que usa transformações baseadas em uma linguagem do padrão MOF-QVT. O resultado gerado (PSM) é um modelo em Java/AspectJ (a confirmar). O artigo estudado, de 2008, foca muito os algoritmos definidos para as transformações que combinam os aspectos com o modelo de negócios. Não há ênfase em um processo de software que leve dos requisitos ao projeto dos modelos de negócio e de aspectos.

Captor/AO

A ferramenta Captor/AO é uma versão orientada a aspectos do Gerador de Aplicações Captor. Nos casos, são usados para gerar famílias de produtos (ou LPS). O gerador original é baseado em composição e funciona usando três elementos de entrada. O primeiro é uma arquitetura geral da aplicação a ser usada, em que os diferentes programas (classes) estão codificados ou de forma fixa ou

então como gabaritos (templates), contendo marcações que serão substituídas por código real durante a geração e, dessa forma, permitindo variações nos produtos gerados. As partes fixas fazem parte de todos os produtos. Os produtos gerados podem estar codificados em qualquer linguagem e depois deverão ser compilados no ambiente EME que deverão executar. O segundo modelo é uma especificação na forma de uma linguagem de domínio, que indica quais são as variabilidades a serem incorporadas no produto que será gerado. Esta linguagem pode ser baseada em um modelo de features, mas não necessariamente. O terceiro elemento é um mapeamento que especifica para cada variabilidade contida na instância da LMA, que partes dos códigos variáveis deverão ser substituídas por um código real previamente definido. O Captor-AP permite que domínios transversais sejam especificados (eles tem portanto sua própria LMA, código fixo e variável e um mapeamento) e que sejam também definidos. Além disso, podem-se criar Pontos de Junção abstratos que especificam como durante a geração, os aspectos abstratos do domínio transversal poderão ser concretizados de forma a entrecortar o domínio-base e inserir novos comportamentos.

	Genarch	CrossMDA	Captor/AO
Processo de Desenvolvimento	Não (para as fases iniciais)	Não (para as fases iniciais)	Não (para as fases iniciais)
Uso de Aspectos	Não	Sim	Sim
Tipo de aplicações que pode derivar	Famílias de produtos (LPS) Frameworks instanciados	Sistemas únicos	Famílias de Produtos, Frameworks instanciados
Resultado	Código fonte Java compilável	Modelo de classes com aspectos combinados	Código fonte compilável (qualquer linguagem)
Extensões	Ao código gerado podem ser adicionados outros códigos prontos pré-existes (o preparo deve ser manual)	--nada observado	Códigos existentes podem ser adicionados ao código gerado, de forma manual.
Modelos usados	Features, componentes/classes (anotadas e com código concreto ou com gabaritos) definindo uma arquitetura da aplicação. Mapeamento entre componentes e features	Classes (usando tanto para o modelo de negócios como para o de aspectos)	LMA Código(concreto ou com gabaritos), definindo uma arquitetura da aplicação. Mapeamento entre LMA é cógio
Modelo de	Usa a linguagem XPand, do	Usa a	Utiliza place-

Templates	Eclipse. Ele possui comandos para gerar código e para navegar na estrutura de arquivos do diretório onde está o programa (pareceu-me que é similar a navegar na AST de programas compilados)	linguagem ATL (ATLAS Transformation Language) para especificar templates de classes e métodos, para uso na transformação de modelos. Talvez possa ser usada para gerar instâncias diferentes de classes ou aspectos, mas no contexto, parece que esse não é o uso adequado dela.	holders que são substituídos por trechos de código, de acordo com a AML. Esses trechos de código podem ser recuperados de arquivos ou estarem codificados no próprio código que realiza a instanciação.