Captor-AO: Gerador de Aplicações apoiado pela Programação Orientada a Aspectos

Carlos Alberto de Freitas Pereira Júnior[1]*
Paulo Cesar Masiero[1]
Rosana Teresinha Vaccare Braga[1]

[1]Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo (USP)
Caixa Postal 668 – 13560-970 – São Carlos – SP – Brasil

{cafpj,masiero,rtvb}@icmc.usp.br

Resumo. *Geradores de aplicações são ferramentas que promovem o reúso de software por gerar aplicações de forma automática, partindo de uma especificação em alto nível. São especialmente importantes em abordagens de desenvolvimento de linhas de produtos de software, em que artefatos podem ser combinados para obter uma grande variedade de diferentes produtos. Este artigo apresenta o Captor-AO, que é uma extensão do gerador de aplicações Captor, também desenvolvido pelo grupo de pesquisa dos autores. O Captor-AO utiliza conceitos da programação orientada a aspectos para auxiliar o desenvolvimento de linhas de produtos, compondo diversos domínios-base e transversais que atualmente precisam ser compostos de forma manual.*

1. Introdução

Geradores de Aplicações são sistemas de software que transformam especificações, geralmente fornecidas em um nível de abstração alto, em aplicações, tipicamente na forma de código-fonte. As especificações descrevem o problema ou a tarefa que deve ser realizada pelo gerador [Cleaveland 1988]. Geradores de aplicações facilitam a implementação de linhas de produtos de software (LPS) [Weiss and Lai 1999, Gomaa 2004], pois podem automatizar a fase de geração dos produtos, em que é realizada a composição de artefatos representando o núcleo de funcionalidades comuns da LPS e de gabaritos (*templates*) que correspondem à parte variável da LPS. Processos de desenvolvimento que utilizam geradores durante a engenharia de aplicação podem ser mais rápidos e menos suscetíveis a erros humanos do que processos tradicionais de codificação, ou seja, os geradores podem produzir código de forma sistemática e mais segura em relação aos métodos tradicionais [Czarnecki and Eisenercker 2002, Smaragdakis and Batory 2000].

A maioria dos geradores de aplicações é desenvolvida especificamente para um domínio de aplicação e permite gerar produtos em tal domínio, bastando para isso que o engenheiro de aplicação forneça a especificação do produto desejado em termos de uma linguagem de modelagem de aplicações (LMA) própria do domínio. Já o gerador de aplicações Captor [Shimabukuro et al. 2006] é configurável, i.e., ele pode ser alimentado com diferentes domínios, que são armazenados em um repositório. No entanto, o Captor trabalha com um domínio de cada vez, ou seja, ao desenvolver uma aplicação, o engenheiro

*Auxílio financeiro da FAPESP - Fundação de Amparo à Pesquisa do Estado de São Paulo

de aplicações deve sinalizar a que domínio ela pertence e, desta forma, o Captor gerará os artefatos com base na configuração do domínio escolhido.

Nesse contexto, este artigo apresenta a ferramenta Captor-AO, que é uma extensão do gerador Captor capaz de realizar a composição simultânea de vários domínios distintos por meio da programação orientada a aspectos (POA) [Kiczales et al. 1997], tornando possível a geração de aplicações formadas por características de um domínio específico (domínio base) e características de diferentes domínios transversais, desde que a união dessas características não produza conflitos no sistema. O interesse dos autores está justamente em utilizar o Captor na geração de produtos de uma LPS, portanto deste ponto em diante será utilizado o termo "geração de produtos".

Este artigo está organizado da seguinte forma: a seção 2 fornece uma visão geral da ferramenta, apresentando a arquitetura e seu funcionamento. Na seção 3 são mostradas outras pesquisas relacionadas a este trabalho. Finalmente, na seção 4 são apresentadas as conclusões e trabalhos futuros.

2. Captor-AO: Extensão do gerador de aplicações Captor

O Captor (do inglês *Configurable APplication generaTOR*) é um gerador de aplicações baseado em composição, que pode ser configurado para diferentes domínios [Shimabukuro et al. 2006]. Para cada domínio, deve-se fornecer ao Captor a especificação de sua LMA, os artefatos a serem utilizados durante a geração (tanto artefatos fixos quanto os gabaritos com as partes variáveis) e o mapeamento entre a LMA e os artefatos. Desde sua criação, o Captor já foi configurado para diversos domínios diferentes. No entanto, cada um dos domínios é utilizado isoladamente na criação de um produto, ou seja, ao iniciar um novo projeto o usuário do Captor deve escolher apenas um entre os domínios previamente configurados. Contudo, alguns produtos de um determinado domínio poderiam ser combinados com outros produtos de um segundo domínio, produzindo novas aplicações funcionais. Surgiu então o interesse em estender a ferramenta para permitir que a geração de novos produtos tenha a capacidade de compor características de diferentes domínios.

A extensão Captor-AO apresentada neste artigo implementa um meio de composição entre domínios-base[1] e domínios transversais, conforme mostrado na Figura 1. O termo "domínio transversal" é utilizado neste artigo para definir domínios de aplicação que usam a tecnologia de aspectos e possuem características transversais. Essas características devem ser projetadas de tal forma que possam ser aplicadas sobre artefatos do maior número possível de domínios-base.

A composição entre domínios é feita utilizando domínios transversais para tirar proveito da facilidade de combinação de características proporcionada pelos aspectos. O Captor-AO possibilita a exploração das propriedades ortogonais de um domínio transversal e dos possíveis pontos de junção de um domínio base. O gerador também implementa regras de validação entre domínios para evitar composições entre domínios incompatíveis. Tanto o código do gerador Captor-AO quanto os binários executáveis podem ser

[1]também denominados domínios específicos, eles representam a parte central do produto gerado e são compostos de um núcleo de características comuns a todos os produtos da LPS e de um conjunto de características que variam de produto para produto

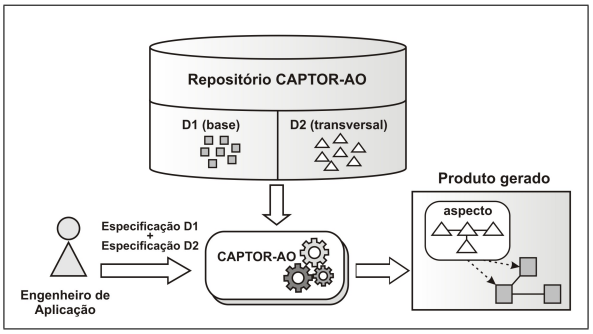

Figura 1. Proposta de geração de aplicações utilizando composição de domínios

obtidos na página do projeto: `http://captor.googlecode.com`. A ferramenta pode ser compilada e empacotada utilizando o gerenciador de projetos Maven [2].

2.1. Arquitetura da Ferramenta

A ferramenta Captor-AO utiliza a abordagem de geração por composição, isto é, a geração de produtos é feita a partir da composição de gabaritos implementados. A especificação da LMA deverá determinar quais gabaritos serão usados para gerar um produto particular [Weiss and Lai 1999]. A arquitetura do Captor-AO é apresentada na Figura 2.

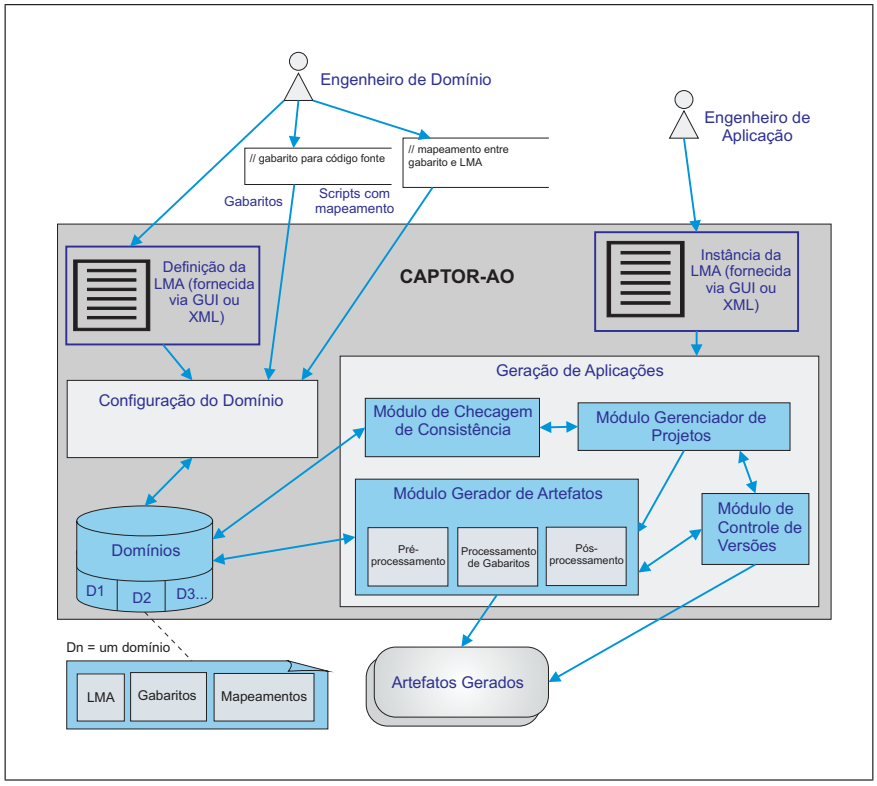

Figura 2. Arquitetura do Captor-AO

A utilização da ferramenta Captor-AO pode ser dividida em duas fases: engenharia de domínio e engenharia de aplicação. Na primeira, o Captor-AO é alimentado com

[2]Apache Maven Project - `http://maven.apache.org/`

os domínios (sejam eles base ou transversais). Essa fase é realizada predominantemente pelo engenheiro de domínio. Na segunda, os produtos da LPS são gerados, sendo essa atividade realizada tipicamente por um engenheiro de aplicações.

Cada uma dessas fases é apoiada por uma parte do gerador (retângulos em cinza claro na Figura 2): Configuração do Domínio e Geração de Aplicações. A Configuração do Domínio é responsável por manter os arquivos que contém todas as informações necessárias sobre o domínio. A Geração de Aplicações é realizada por diversos módulos: o módulo Gerenciador de Projetos é responsável pela manutenção dos dados dos projetos criados no Captor-AO, ou seja, criação, manutenção e remoção de projetos de novos domínios e de novos produtos. Durante a geração de artefatos, o módulo Gerenciador de Artefatos tem a responsabilidade de selecionar os gabaritos do(s) domínio(s) envolvido(s) que devem ser gerados e utilizar seu sub-módulo de Processamento de Gabaritos para produzir os artefatos da aplicação gerada, com base em arquivos de gabarito e mapeamentos contidos no repositório de domínios e no arquivo com os dados da especificação. Há ainda módulos para garantir a consistência das especificações fornecidas e para auxiliar no controle de versões dos produtos.

2.2. Engenharia de Domínio

Existem dois processos diferenciados para a engenharia de domínio, dependente do tipo de domínio, ou seja, se ele é base ou transversal. Para um domínio base, o processo é bastante parecido com o utilizado na versão original da ferramenta, e possui os seguintes passos: criar uma linguagem de modelagem de aplicações (LMA) para o domínio, criar os artefatos reutilizáveis desse domínio, criar gabaritos utilizando a linguagem XSL, e criar um arquivo de mapeamento entre a LMA e os gabaritos. Na Figura 3 é mostrada a interface utilizada durante a criação de um novo domínio base que, nesse caso, é um domínio fictício de sistemas para video locadoras. Pode-se observar na figura que o Captor solicita ao engenheiro de domínio o tipo de domínio (base ou transversal). Caso seja escolhido o tipo base, solicita-se ao engenheiro que defina, dentre os domínios transversais já registrados no repositório, quais são compatíveis com o novo domínio base.

Já o processo de engenharia de domínio transversal possui, além dos passos em comum com o processo do domínio base, três passos adicionais: modelagem dos pontos de junção abstratos, definição dos domínios-base compatíveis e inclusão dos conjuntos de extensão nos domínios-base. A seguir são detalhados esses novos conceitos.

Um domínio transversal pode possuir dois tipos de variabilidades, denominadas neste artigo de variabilidade funcional e variabilidade de composição: a primeira é inerente às regras de negócios do domínio e dita as características que variam de um produto para outro da LPS; a segunda diz respeito aos pontos de junção que permitirão combinar o domínio transversal com um ou mais domínios-base. Pode-se citar como exemplo de domínio transversal o framework para persistência de dados apresentado em Camargo [2006] . Nesse framework, que pode ser combinado com diversos domínios-base que possuam o interesse de persistência, há variabilidades funcionais, tais como escolha de *drivers* e senhas de conexão, e variabilidades de composição, tais como a definição das entidades de uma aplicação base a serem persistidas.

Para implementar as variabilidades de composição, o Captor-AO utiliza *pontos de junção abstratos* (PJAs), que são os elementos responsáveis pelo acoplamento entre

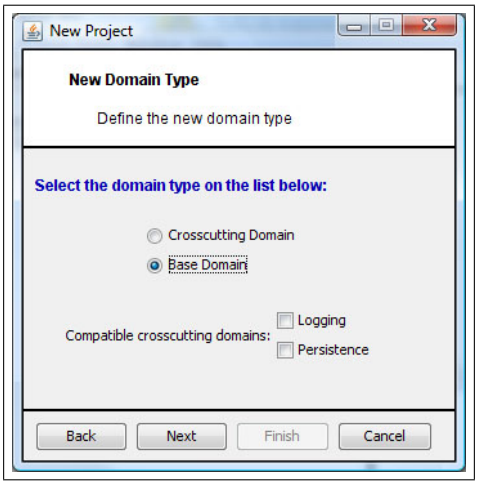

Figura 3. Interface do Captor-AO utilizada durante a engenharia de domínio.

as características ortogonais do domínio transversal e o domínio base. Esse conceito é baseado na noção de aspectos abstratos, que definem pelo menos um conjunto de junção abstrato que deve ser sobreposto em tempo de compilação.

Para obter a separação entre variabilidades funcionais e variabilidades de composição, o engenheiro de domínio deve utilizar um elemento de formulário específico disponibilizado pelo Captor-AO, chamado `abstractPointcutPanel`, sempre que quiser representar um PJA. Para as variabilidades funcionais, ele utiliza os demais elementos de acordo com o tipo de informação a ser lida (campos de entradas, painéis de texto, listas, botões, etc.). Cada `abstractPointcutPanel` possui um identificador que é utilizado por marcações especiais nos gabaritos do domínio transversal. Essas anotações são responsáveis pela concretização do PJA durante a geração de um produto. Cada anotação pode acessar o valor concreto fornecido pelo engenheiro de aplicação no respectivo elemento de formulário. Um mesmo PJA pode ser referenciado em diversos gabaritos e compartilhado por mais de um domínio transversal.

Criados os pontos de junção do domínio transversal, o engenheiro de domínio deve definir quais dos domínios-base já configurados podem ser entrecortados por esses pontos, i.e., quais domínios são compatíveis e podem ser compostos com o novo domínio transversal. Entende-se como compatibilidade entre domínios a possibilidade de união e/ou complementação de características desses domínios sem a ocorrência de conflitos, resultando em novas características funcionais. A versão atual da ferramenta permite a escolha de um domínio base e de zero ou mais domínios transversais. Para cada domínio base compatível, deve ser adicionada uma referência do domínio transversal em um arquivo de configuração específico. Antes da escolha dos domínios transversais compatíveis com um domínio base, o engenheiro de domínio deve realizar uma análise detalhada do impacto das *features* transversais sobre o código base.

De acordo com o princípio da inconsciência (*obliviousness*), a dependência entre uma característica transversal e o código base deveria garantir que a aplicação base desconheça a existência do código transversal [Griswold et al. 2006]. Contudo, na prática, existem situações em que se torna necessário realizar pequenos ajustes em um domínio base para permitir a compatibilidade com determinados domínios transversais. Por exemplo,

em certos frameworks transversais de persistência, o código base deve possuir determinados chamadas de métodos para que framework possa definir os pontos da aplicação onde serão realizadas as operações de remoção e busca (métodos `delete()` e `find()`, por exemplo). Essa condição cria uma dependência nas aplicações do domínio base quando o domínio transversal é utilizado. No Captor-AO foi implementado um recurso, chamado *conjunto de extensão*, que permite realizar adaptações nos templates de um domínio base de acordo com os domínios transversais aplicados no momento da geração. Um conjunto de extensão inclui uma anotação especial contendo um trecho de gabarito que é processado somente se um certo domínio transversal fizer parte da composição. No exemplo anterior, os gabaritos de um domínio base poderiam declarar um conjunto de extensão para a inserção dos métodos `save()` e `find()` no locais necessários dentro da nova aplicação caso o domínio de persistência tenha sido selecionado.

2.3. Engenharia de Aplicação

Na fase de engenharia de aplicação, o Captor-AO permite ao engenheiro selecionar qual domínio base e, opcionalmente, quais domínios transversais serão utilizados durante a geração do novo produto. No exemplo da Figura 4 um novo produto é criado utilizando um domínio base (VideoRentalStore) e um transversal (Persistence). O processo de instanciação de membros da LPS, como pode ser observado na Figura 5, é baseado no preechimento de formulários referentes às LMA's de cada domínio envolvido na composição. O nome do domínio base aparece entre parênteses na árvore mais à esquerda da figura, enquanto os domínios transversais aparecem entre "<" e ">".

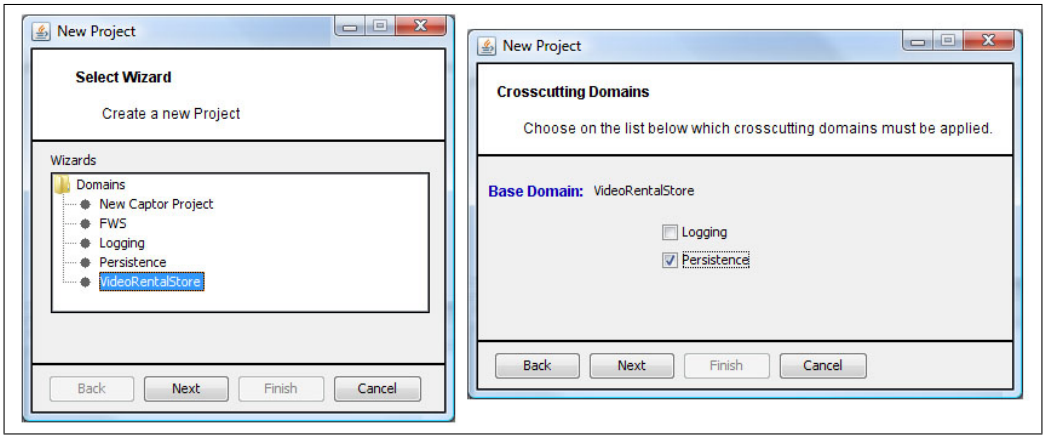

Figura 4. Definição de domínios para um novo produto.

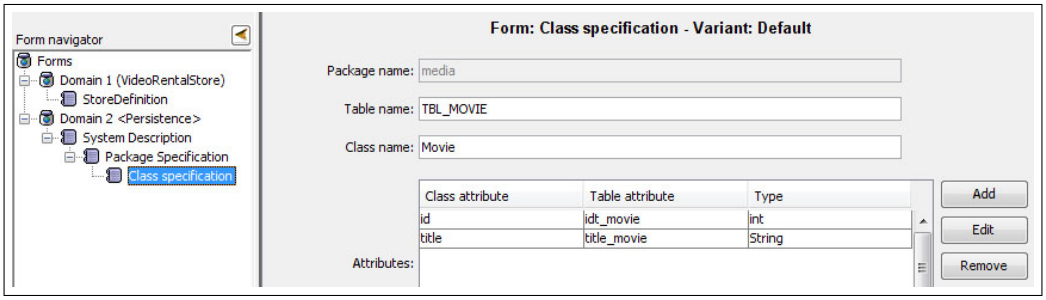

Figura 5. Interface utilizada durante a engenharia de aplicação.

3. Trabalhos Relacionados

No trabalho de Lee et al. [2006], existe interesse em criar um processo bem definido para o desenvolvimento de *features* apoiado pela POA. Atualmente, a maior parte dos trabalhos tratam *features* como camadas sobrepostas ao sistema base. Novas *features* podem ser inseridas por meio de refinamentos (incrementos) nas *features* já existentes. Esses trabalhos utilizam a POA para permitir uma maior modularização das *features* transversais da LPS. Nessa linha estão inclusas as pesquisas baseadas nas arquitetura CAESAR [Mezini and Ostermann 2004] e AHEAD [Apel et al. 2005] e a metodologia apresentada em Pacios [2006].

No trabalho de Lesaint and Papamargaritis [2004] é apresentado um gerador de aplicações por compilação baseado no modelo de arquitetura GenVoca [Batory et al. 2000], que gerador possui uma linguagem para geração de membros que implementa os componentes do núcleo da LPS como classes modulares tradicionais e os refinamentos como componentes aspectuais. Kulesza et al. [2007] apresentam um processo de derivação de arquiteturas de LPS orientadas a aspectos, com o objetivo de gerar uma arquitetura de fácil acoplagem/desacoplagem de *features*. Nesse sentido, define-se o conceito de *Extension Join Points* (EJPs) [Kulesza et al. 2006], que são pontos de entrecorte bem definidos na arquitetura que permitem não só a composição de *features* opcionais e alternativas da LPS, mas também a evolução da arquitetura por meio de aspectos de extensão. Essa abordagem utiliza gabaritos para que a configuração e a instanciação de produtos possa ser apoiada por ferramentas de automação [Cirilo et al. 2007].

Nota-se que nas pesquisas mencionadas utiliza-se POA com a intenção de modularizar *features* e facilitar a instanciação de produtos de um mesmo domínio. A abordagem proposta pelo Captor-AO propõe a utilização de aspectos para a criação de produto composto por *features* de diferentes domínios.

4. Conclusões e Trabalhos Futuros

Este artigo apresentou a ferramenta Captor-AO, um gerador de aplicações capaz de criar aplicações utilizando composições entre domínios-base e transversais. O Captor-AO permite diferenciar as variabilidades inerentes ao domínio das variabilidades decorrentes do uso de aspectos na implementação dos domínios transversais.

A ferramenta possui recursos para configuração de novos domínios e geração de produtos, adaptados para fornecer suporte à utilização de características transversais. Estão sendo feitos ajustes em sua interface com o objetivo de melhorar a usabilidade. Futuramente, o Captor-AO será alterado para que também possa realizar combinações entre dois ou mais domínios-base. Pretende-se testar o desempenho do Captor-AO utilizando estudos de caso com famílias de produtos mais complexas.

Referências

Apel, S., Leich, T., Rosenmuller, M., and Saake, G. (2005). Combining feature-oriented and aspect-oriented programming to support software evolution. In *2nd ECOOP Workshop on Reflection, AOP and Meta-Data for Software Evolution (RAM-SE'05)*.

Batory, D., Cardone, R., and Smaragdakis, Y. (2000). Object-oriented framework and product lines. In *Proc. of the 1st conference on Software product lines: experience and research directions*, pages 227–247, MA, USA. Kluwer Academic Publishers.

Camargo, V. V. (2006). *Frameworks transversais: definições, classificações, arquitetura e utilização em um processo de desenvolvimento de software*. Tese de doutorado, ICMC-USP, São Carlos, SP.

Cirilo, E., Kulesza, U., and Lucena, C. (2007). Genarch: a model-based product derivation tool. In *Proceedings of Simpósio Brasileiro de Componentes, Arquitetura e Reutilização de Software (SBCARS'2007)*, pages 31–46.

Cleaveland, J. C. (1988). Building application generators. *IEEE Software*, 9(4):25–33.

Czarnecki, K. and Eisenercker, U. W. (2002). *Generative programming*. Addison-Wesley.

Gomaa, H. (2004). *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*. Addison-Wesley.

Griswold, W. G., Sullivan, K., et al. (2006). Modular software design with crosscutting interfaces. *IEEE Software*, 23(1):51–60.

Kiczales, G., Lamping, J., Menhdhekar, A., Maeda, C., Lopes, C., Loingtier, J. M., and Irwin, J. (1997). Aspect-oriented programming. In *European Conference on Object-Oriented Programming*, pages 220–242. Springer-Verlag.

Kulesza, U., Alves, V., Garcia, A., Neto, A., Cirillo, E., Lucena, C., and Borba, P. (2007). Mapping features to aspects: A model-based generative approach. 10th International Workshop on Early Aspects, AOSD'07.

Kulesza, U., Coelho, R., Alves, V., Garcia, A., von Staa, A., Lucena, C., and Borba, P. (2006). Implementing framework crosscutting extensions with ejps and aspectj. In *Proceedings of ACM SIGSoft XX Brazilian Symposium on Software Engineering*.

Lee, K., Kang, K. C., Kim, M., and Park, S. (2006). Combining feature-oriented analysis and aspect-oriented programming for product line asset development. In *SPLC '06: Proceedings of the 10th International on Software Product Line Conference*, pages 103–112, Washington, DC, USA. IEEE Computer Society.

Lesaint, D. and Papamargaritis, G. (2004). Aspects and constraints for implementing configurable product-line architectures. In *Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA 2004)*, pages 135–144, USA. IEEE Computer Society.

Mezini, M. and Ostermann, K. (2004). Variability management with feature-oriented programming and aspects. *SIGSOFT Softw. Eng. Notes*, 29(6):127–136.

Pacios, S. F. (2006). Uma abordagem orientada a aspectos para desenvolvimento de linhas de produtos de software. Master's thesis, ICMC/USP, São Carlos, SP.

Shimabukuro, E. K., Masiero, P. C., and Braga, R. T. V. (2006). Captor: Um gerador de aplicações configurável. Sessão de ferramentas do 20° Simpósio Brasileiro de Engenharia de Software (XX SBES).

Smaragdakis, Y. and Batory, D. (2000). *Application generators. Encyclopedia of Electrical and Electronics Engineering*. John Wiley and Sons.

Weiss, D. M. and Lai, C. T. R. (1999). *Software product-line engineering: a family-based software development process*. Addison-Wesley.