6
Trabalhos Relacionados

Nesse capítulo a abordagem de derivação implementada pela ferramenta GenArch é comparada com diversos trabalhos já desenvolvidos. A abordagem é confrontada com: (i) uma abordagem de derivação de LPS; (ii) duas abordagens para instanciação de framewoks OO; e (iii) uma abordagem de configuração de aplicação corporativas.

6.1.
Abordagem para Derivação de Linhas de Produtos de Softaware

Jansen (Jansen et al. 2004) apresenta uma abordagem para especificação, implementação e derivação de LPS baseada na composição de características. Nessa abordagem a seleção e composição de um conjunto de componentes durante a derivação é feita automaticamente a partir do modelo de características e de um algoritmo de composição. O modelo de característica proposto é formado por características compostas por *roles* que representam as funcionalidades da LPS. Uma *role* é um conjunto de interfaces e/ou pedaços de código que representam as variabilidades da LPS. A composição entre os componentes que formam o núcleo da LPS (partes comuns) e as *roles* (variabilidades) se dá através de *actors*. Um *actor* permite a integração automatizada entre componentes bases e *roles* de acordo com algumas regras de composição: (i) assinaturas de operações e implementação são todas diferentes; (ii) assinaturas de operação são diferentes e as implementações são iguais; (iii) assinaturas de operações e implementações são iguais; e (iv) assinaturas de operações são iguais e implementações são diferentes. Apesar de algumas dessas regras de composições não serem automaticamente resolvidas de forma adequada, somente a ultima é realmente problemática. A resolução da combinação entre implementações diferentes com mesma assinatura pode ser feita através da concatenação das diferentes implementações, quando não há parâmetros de entrada nem de saída. Quando a saída é igual à entrada a combinação pode ser feita através do encadeamento

das implementações. No caso da saída ser diferente da entrada um código de composição se torna necessário.

Assim como essa abordagem de derivação, a abordagem apresentada nesta dissertação de mestrado também suporta a derivação automática de produtos baseada no modelo de características. Durante o processo de derivação, entretanto, tal abordagem se concentra na composição de *roles, actors* e componentes, diferentemente da abordagem apresentada nesse trabalho que esta centrada na seleção e geração de elementos de implementação (classes, aspectos, componentes, arquivos extras). A abordagem de composição proposta por Jansen (Jansen et al. 2004) permite a composição em nível de operações. Já a abordagem implementada pela ferramenta GenArch somente permite customização em nível de classe, componentes e abstrações de mais alto nível.

6.2. Abordagem para Instanciação de Framewoks OO

Filho et al (Filho et al. 2004) propõem uma abordagem para instanciação de frameworks centrada no uso do modelo de característica e da linguagem UML. A abordagem é estruturada em três passos: (i) *processo de documentação do framework* – um diagrama de classes do framework é relacionado com o seu respectivo modelo de característica. Elementos de projeto (classes, métodos, atributos) que representam pontos de extensão do framework são explicitamente anotados no diagrama de classes representando o framework. Tais elementos de projeto foram definidos como extensões do meta-modelo de UML. Finalmente, relações de dependência UML são definidas para relacionar elementos de ambos os modelos; (ii) *processo de geração de script de instanciação* – nesse passo da instanciação do framework, desenvolvedores selecionam as características desejadas e a ferramenta de posse do modelo de característica e do diagrama de classes que descreve o framework, gera um script na linguagem RDL (*Reuse Description Language*) (Oliveira et al. 2004) que contém os passos necessários para instanciar a aplicação; e (iii) *instanciação do framework* – no último passo da instanciação o script RDL gerado pela atividade anterior é processado para estender o diagrama de classes do projeto original do framework de forma a inserir as classes, métodos e atributos que implementam a concretização dos seus pontos de extensão. Esse processo é semi-automático, intervenções humanas podem ocorrer de forma a solicitar dos desenvolvedores informações adicionais, tais como, nomes de classes, métodos e atributos.

Trabalhos recentes têm estendido a abordagem proposta para lidar com a instanciação de frameworks orientados a aspectos (Penczek et al. 2006).

Existem similaridades e diferenças entre o modelo generativo da ferramenta GenArch para instanciação de produtos de software e a abordagem proposta por Filho (Filho et al. 2004). Ambas as abordagens são centradas na definição de três modelos: característica, configuração e de arquitetura/implementação. O modelo de característica tanto na abordagem apresentada nesse trabalho quanto no trabalho de Filho (Filho et al. 2004) é utilizado para especificar as características de uma LPS. O modelo de arquitetura usado por tais autores é o diagrama de classes UML, enquanto na nossa abordagem tal modelo é uma representação dos artefatos de implementação. As relações de dependência definidas em nosso modelo de configuração têm um propósito similar àquelas propostas por tais autores. Uma diferença fundamental entre as abordagens, é o uso de templates pela ferramenta GenArch para expressar variabilidades ocorrendo em elementos de implementação da arquitetura, enquanto Filho, Oliveira et al (Filho et al. 2004) usam anotações explícitas em modelos de projeto para representar os pontos de extensão. Templates é uma tecnologia amplamente difundida pela comunidade de geração de código, e adotada em muitos cenários pela comunidade de desenvolvedores.

(Antkiewicz et al. 2006) propõem o uso de linguagens de modelagem específica para frameworks (FSML) para suportar o desenvolvimento de aplicações baseadas em frameworks. Uma FSML é uma linguagem de modelagem específica de domínio projetada para modelar os conceitos que norteiam um framework e especificar como esses podem ser utilizados e implementados no código da aplicação. Tal linguagem consiste de uma sintaxe abstrata, um mapeamento entre a sintaxe abstrata e a API do framework e, opcionalmente, uma sintaxe concreta. FSML é designada para capturar os conceitos e características providas pelo framework como abstrações de uma linguagem. A sintaxe abstrata codifica todas as configurações válidas das escolhas de implementação estipuladas pelo framework. O mapeamento entre a sintaxe abstrata e a API do framework define como conceitos e características são mapeados para pontos de extensão. O mapeamento é composto por duas partes: (i) *forward mapping* – define como gerar código ou atualizar código existente para um conceito; e (ii) *reverse mapping* – define como reconhecer uma instância de um conceito no código. Juntos, *forward mapping* e *reverse mapping* permitem *round-trip* automático, onde código pode ser criado a partir

dos modelos, modelos podem ser criados a partir do código e modificações feitas no código ou modelo podem ser identificadas e ambos reconciliados. A partir de uma FSML, desenvolvedores de aplicações podem criar modelos que representam uma instância em particular de um framework. Esses modelos, denominados modelos específicos do framework, contêm valores concretos para os campos abstratos definidos na FSML. O uso de uma FSML para representar os conceitos de um framework como elementos de implementação permite um melhor entendimento de quais pontos do framework precisam ser completados, facilitando, com isso, o processo de instanciação. Um modelo específico do framework permite que o processo de implementação de uma instância seja feito de forma automatizada, através dos *forward mapping* definidos na FSML. Além disso, modificações no código da instância podem ser refletidas no modelo, a partir dos *reverse mapping* definidos na FSML.

Assim como o trabalho de Antkiewicz et al, a abordagem apresentada nesta dissertação também permite mapear conceitos de um framework para elementos de implementação, mas em níveis de granularidade diferentes. No trabalho de Antkiewicz et al, conceitos podem ser mapeados para elementos de implementação de baixa granularidade (classes, métodos, atributos, parâmetros). Já a ferramenta GenArch somente permite mapeamentos de granularidade mais alta (classes, aspectos, pacotes, arquivos). Assim como a ferramenta GenArch, uma FSML também fomenta um processo de instanciação automatizado. No entanto, uma FSML vai um pouco mais além, permitindo também, de forma automatizada, a codificação de alguns pontos, como: valores de atributos; parâmetros; dentre outros. Ambos permitem engenharia de *round-trip*, onde os modelos podem ser atualizados ou criados a partir do código fonte e atualizações nos modelos podem ser refletidas no código. O trabalho de Antkiewicz et al implementa uma engenharia de *round-trip* mais sofisticada do que a ferramenta GenArch, onde não somente o código do framework, mas também o código da aplicação influencia em modificações nos modelos e vice-versa.

6.3. Abordagem para Configuração Automática de Aplicações Corporativas

White (White et al. 2007) apresenta uma abordagem automatizada para configuração de aplicações corporativas com o objetivo principal de permitir que tal configuração seja executada por equipes separadas geograficamente. Para

atingir este objetivo, a abordagem se baseia na definição de um modelo de características e de uma série de pontos de verificação. Para endereçar os desafios da configuração descentralizada à abordagem envolve, primeiramente, a construção de um modelo de características formal, onde deve-se determinar quais variabilidades possuem restrições e valores para as demais variabilidades que estão consistentes com as restritas. Após a definição formal do modelo de característica, deve-se criar um conjunto de pontos de verificação que automaticamente validam as decisões feitas por cada equipe, de forma a certificá-las com as restrições definidas no modelo de características. Um ponto de verificação, na abordagem, é um código Java executável que mede o valor de uma determinada propriedade. A partir de uma instância do modelo de características corretamente criada a abordagem permite a configuração automática uma aplicação. A configuração é feita através de arquivos de configurações XML com anotações que mapeiam pedaços de código para características. Estes mapeamentos permitem a seleção das funcionalidades, descritas em pedaços do arquivo de configuração, que a aplicação implementará. Isso também permite a customização de arquivos de configuração para diferentes seleções de características.

Assim como tal abordagem, a extensão da ferramenta GenArch apresentada no capítulo 5 também habilita a configuração de aplicações corporativas, no caso, implementadas com as tecnologias Spring e/ou OSGi. A vantagem da ferramenta GenArch é que sua abordagem permite que o mapeamento entre componentes que podem variar e características seja feito facilmente através dos modelos de derivação. No caso da abordagem de White et al (White et al. 2007) o mapeamento em um arquivo XML pode ser custoso e sujeito a erros. Mas a possibilidade de se permitir a configuração por equipes diferentes e em linguagens apropriada para cada uma delas é uma atividade que não é possível de ser executada com a ferramenta GenArch.