1 Introdução

Pesquisas e experiências práticas recentes (Clements et al. 2001) (Greenfield et al. 2005) (Weiss et al. 1999) sugerem que para obter um progresso significativo com respeito ao reuso é necessário focalizar a modelagem e desenvolvimento de software seguindo uma sistemática de engenharia de linha de produtos, ao invés de desenvolver sistemas individualmente. A engenharia de linha de produtos busca explorar os pontos comuns e variáveis existentes de um conjunto de sistemas, de forma que diferentes customizações de aplicações de um mesmo domínio possam ser rapidamente criadas a partir de um conjunto de artefatos reusáveis. Uma linha de produtos pode ser vista como uma família de sistemas (Parnas 1976) que endereça um segmento de mercado específico.

Nos últimos anos, várias abordagens para o desenvolvimento de família de sistemas e linha de produtos foram propostas (Weiss et al. 1999) (Clements et al. 2001) (Czarnecki et al. 2000) (Greenfield et al. 2005). Várias dessas abordagens procuram definir, modelar e implementar uma arquitetura comum e flexível que atenda as características comuns e variáveis requeridas pela linha de produto. Uma característica (*feature*) (Czarnecki et al. 2006) é uma propriedade ou funcionalidade da linha de produto que é relevante para algum dos interessados no seu desenvolvimento. Essa arquitetura é tipicamente definida em um processo denominado *engenharia de domínio* (Czarnecki et al. 2000).

Arquiteturas de linha de produtos de software (LPSs) são implementadas a partir da definição de um conjunto de artefatos reutilizáveis, tais como: frameworks orientados a objetos, bibliotecas de softwares, componentes, arquivos de configuração, dentre outros. Cada um desses artefatos endereça um pequeno conjunto de características comuns e/ou variáveis. Mais recentemente, novas técnicas de programação, tais como, programação orientada a aspectos (Alves et al. 2006) (Kulesza et al. 2006), programação orientada a características (Batory et al. 1999) (Smaragdakis et al. 2002), (Mezini et al. 2004) e geração de código (Czarnecki et al. 2000), vem sendo exploradas no intuito de melhorar a gerência e modularização de tais características.

1.1. Problemas

Apesar de todos os benefícios apontados, a adoção de abordagens de desenvolvimento de LPSs não é uma atividade simples e trivial. Krueger (Krueger 2001) afirma que, em parte essa situação ocorre, porque durante várias décadas foram desenvolvidos métodos, técnicas e ferramentas centrados principalmente na construção de sistemas de software únicos, havendo poucas ferramentas para o gerenciamento e manipulação de variações de LPSs.

Em um processo típico de desenvolvimento de LPSs (Czarnecki et al. 2000), após a especificação e implementação da arquitetura de LPS na engenharia de domínio, ocorre à etapa de *engenharia de aplicação*. Durante esse estágio, os artefatos de implementação produzidos anteriormente são compostos e integrados para gerar uma instância (produto) da LPS. Esse processo é também conhecido como *derivação de produto* (Sybren et al. 2005). Idealmente, este processo deve ser realizado com o apoio de uma ferramenta de instanciação que facilite a seleção, composição e configuração dos artefatos e suas respectivas variabilidades. Abordagens modernas de engenharia de software, tais como Desenvolvimento Generativo (Czarnecki et al. 2000) e Fábricas de Software (Greenfield et al. 2005), oferecem fundamentos para a definição de técnicas e mecanismos que auxiliam na automação do processo de derivação de produtos. O objetivo central de tais abordagens é melhorar a produtividade e qualidade do processo de derivação. Linguagens específicas de domínios e geradores de código são duas das principais tecnologias adotadas com tal propósito.

A definição de ferramentas e mecanismos para apoiar o processo automatizado de derivação de produtos é fundamental para lidar com a complexidade e escala de LPSs, além da variedade de tecnologias de implementação adotadas na sua implementação. Recentemente algumas ferramentas para gerência de variabilidades e derivação automática de produtos de software foram desenvolvidas. Gears (Gears 2007) e pure::variants (pure::variants 2007) representam relevantes ferramentas disponíveis na indústria atualmente para derivação de LPSs.

1.2. Limitação das Abordagens Atuais

Embora as ferramentas de derivação de produto atuais, tais como Gears e pure::variants, ofereçam uma série de funcionalidades úteis para o desenvolvimento e derivação automática de LPSs, elas são, em geral, bastante complexas para serem usadas pela comunidade tradicional de desenvolvedores que possuem pouca familiaridade com vários dos novos conceitos propostos pela comunidade de LPSs. Além disso, a maioria das ferramentas de LPSs oferece poucos recursos de integração com novas tecnologias que vêm sendo adotadas pela comunidade, tais como, programação orientada a aspectos (Kiczales 1997) (Filman et al. 2005) e infra-estruturas de desenvolvimento baseada em componentes (Johnson et al. 2005) (OSGi 2003) (SpringDynamicModules 2008).

1.3. Solução Proposta

Nesse contexto, este trabalho propõe uma abordagem baseada em modelos que visa simplificar a preparação de LPSs para serem derivadas automaticamente. Uma ferramenta, denominada GenArch – *Generative Architectures*, oferece suporte para a abordagem em atividades de configuração e derivação de uma LPSs ou de instâncias de um *framework* OO. A derivação automática de produtos na ferramenta é realizada através da definição de três modelos: (i) característica – modelo utilizado na abordagem para representar os pontos de variabilidade presentes na arquitetura da LPS; (ii) configuração – define um conjunto de mapeamentos entre elementos do modelo de implementação da LPS e características do modelo de características; e (iii) implementação – define uma representação visual dos elementos que implementam a LPS, com o objetivo de facilitar o mapeamento entre tais elementos e características presentes no modelo de características. Versões iniciais desses modelos são geradas automaticamente pela ferramenta a partir: (i) da especificação de anotações no código de artefatos de implementação que agregam as variabilidades da LPS; ou (ii) da análise gramatical (*parsing*) de arquivos de configuração que especificam os componentes da aplicação e suas propriedades.

1.4. Objetivos

Este trabalho tem como objetivo principal o desenvolvimento de uma ferramenta, denominada GenArch, fundamentada em uma abordagem baseada em modelos que visa simplificar a preparação de LPSs ou *frameworks* OO para serem derivados automaticamente. Em particular, o trabalho tem os seguintes objetivos específicos:

- Propor uma abordagem que simplifique a preparação de linhas de produtos ou *frameworks* para serem derivados automaticamente;
- Propor, projetar e implementar uma ferramenta baseada em modelos que facilite, através da automação, a configuração e derivação de LPSs ou *frameworks*;
- Desenvolver estudos de casos como forma de validação da abordagem e da ferramenta proposta;
- Projetar e implementar extensões para a ferramenta que enderece a derivação de LPS implementadas com as tecnologias de componentes Spring e OSGi.

1.5. Organização do Texto

Essa dissertação está organizada em mais 7 capítulos, além desse introdutório.

O capítulo 2 apresenta os conceitos que envolvem o desenvolvimento de LPS. Ele traz um resumo das principais tecnologias de desenvolvimento de LPS existentes, tais como: (i) desenvolvimento generativo; (ii) frameworks orientados a objetos; (iii) desenvolvimento orientado a aspectos; e (iv) tecnologia de componentes. O capítulo também apresenta duas das principais ferramentas para derivação de produtos de software existentes no mercado: *pure::variants* e *Gears*.

O capítulo 3 apresenta uma visão geral da abordagem de derivação de produtos proposta, assim como a ferramenta GenArch, responsável pela automação da abordagem. São também detalhadas a arquitetura e tecnologias utilizadas no desenvolvimento de tal ferramenta, usando tecnologias baseadas na plataforma Eclipse (Shavor et al. 2003).

O capítulo 4 descreve dois estudos de casos de frameworks e arquitetura de LPSs que foram preparadas e, posteriormente, derivadas automaticamente

pela ferramenta. Para cada estudo de caso é apresentado, detalhadamente, os passos de preparação da arquitetura (criação dos modelos) para ser derivada e o processo de derivação de produtos específicos para cada LPS.

O capítulo 5 descreve uma extensão da ferramenta GenArch para endereçar a preparação e derivação de arquiteturas de LPS implementadas usando tecnologias de componentes Spring e/ou OSGi. O capítulo discute como cada modelo da ferramenta e etapas originais da abordagem de derivação foram estendidos para permitir a modelagem e derivação de tais arquiteturas.

O capítulo 6 descreve alguns trabalhos relacionados com a abordagem proposta nesta dissertação. Finalmente, o capítulo 7, conclui essa dissertação resumindo as contribuições e sugerindo direções para trabalhos futuros.