7
Trabalhos Relacionados

"A idéia é tentar dar todas as informações que ajudem os outros a julgar o valor da sua contribuição; não apenas as informações que levem o julgamento a uma direção em particular." (Richard Feynman)

Neste capítulo são apresentados alguns trabalhos relacionados com esta dissertação de mestrado. Na Seção 7.1 são apresentados alguns trabalhos na área de LP-SMA. Na Seção 7.2 são apresentados alguns estudos empíricos no contexto de LPS. Na Seção 7.3 o trabalho é confrontado com alguns estudos empíricos no contexto de agentes de software. Por fim, as limitações dos trabalhos relacionados e as principais diferenças em relação a este trabalho são apresentadas na Seção 7.4.

7.1
Abordagens para Linha de Produtos de Sistemas Multi-agentes

Alguns trabalhos de pesquisa recentes começaram a investigar a integração das tecnologias de SMA e LPS. Dehlinger & Lutz (Dehlinger and Lutz 2005) propõem um *template* de especificação de requisitos orientado a agentes extensível para sistemas distribuídos. O objetivo desses *templates* é prover um suporte ao reuso de requisitos. A proposta dos autores adota a abordagem de linha de produto em SMA e utiliza a metodologia Gaia. Os requisitos são especificados em dois níveis esquemáticos: papel e pontos de variação do papel. O primeiro nível define o papel e as possíveis variações que ele pode exercer durante o seu ciclo de vida. O segundo captura os requisitos dos pontos de variação do papel. Essa proposta permite o reuso da configuração do agente ao longo da evolução do sistema. Cada configuração do agente pode ser dinamicamente modificada e reusada em aplicações similares.

Pena et al. (Pena et al. 2007) descrevem uma abordagem para descrever, entender e analisar sistemas evolucionários. A abordagem é baseada na visão de diferentes instâncias de um sistema e como o mesmo pode evoluir para diferentes produtos em uma LPS. Seguindo essa abordagem, uma LPS é desenvolvida com uma metodologia de agentes. Os autores propõem um

conjunto de técnicas de engenharia de software baseada em uma metodologia orientada a agentes chamada MacMAS (*Methodology for analyzing Complex Multi-Agent Systems*), cujo objetivo é lidar com sistemas complexos imprevisíveis. MacMAS permite a descrição de uma mesma *feature* em diferentes níveis de abstração. Neste trabalho também é apresentado um processo para construir a arquitetura núcleo de uma LP-SMA no estágio de engenharia de domínio, cujas atividades são: (i) construir os relacionamentos da organização; (ii) construir o modelo de *features*; e (iii) analisar as *features* obrigatórias e variáveis. A principal vantagem da abordagem está no fato de tornar possível a derivação de um modelo formal do sistema em cada estado que ele pode desempenhar.

7.2 Estudos Empíricos em Linha de Produtos de Software

Alguns trabalhos recentes apresentam estudos com o uso de POA no desenvolvimento de LPS (Griss 2000, Alves et al. 2007, Colyer et al. 2004). A seguir, segue uma pequena relação de trabalhos que focam em estudos empíricos qualitativos e quantitativos em LPS.

Figueiredo et al. (Figueiredo et al. 2008) apresentam um estudo empírico de duas LPS para dispositivos móveis: MobileMedia e BestLap. Esse trabalho avalia quantitativamente e qualitativamente os efeitos positivos e negativos de POA em um conjunto de cenários de evolução que são aplicados à arquitetura núcleo e as *features* variáveis da LPS. O objetivo desse trabalho foi observar e analisar em quais aspectos os mecanismos de POA melhoram ou não a modularidade e a estabilidade da LPS na presença de mudanças reais. Eles analisam qualitativamente e quantitativamente a evolução e estabilidade dessas LPS através de um conjunto de métricas de software. Essas métricas focam em atributos de modularidade (separação de *concerns*, coesão, acoplamento, tamanho), impacto de mudanças (número de componentes adicionados, modificados e removidos, número de linhas de código adicionadas, modificadas e removidas, número de operações adicionados e removidos) e interação entre *features*. Nesse estudo, duas técnicas de implementação de variabilidades foram consideradas nas implementações das LPS: OO com suporte à compilação condicional e POA. Como conclusões desse estudo, foi observado que as implementações OA das LPS têm um projeto mais estável quando as mudanças focam em *features* opcionais e alternativas. Isto indicou que as decomposições aspectuais são superiores, principalmente quando se considera o princípio *Open-Closed* (Meyer 2000). Porém, os mecanismos de OA não foram adequados na inclusão de *features* obrigatórias ou quando houve a transformação de

uma *feature* obrigatória em duas *features* alternativas.

Em (Apel and Batory 2006) é apresentado um estudo comparativo entre a programação orientada a features (FOP - *Feature Oriented Programming*) e POA para implementação de *features* em uma LPS. A FOP procura lidar com o encapsulamento de *features* que pode ser usado para estender a funcionalidade de programas já existentes. *Mixin-Layers* é uma das técnicas utilizadas para se implementar *features* em FOP. Uma *mixin-layer* é um módulo que encapsula fragmentos de diversas classes. A implementação da LPS usa AML (*Aspectual Mixin Layers*), que é uma abordagem para integrar FOP e AOP. Cada AML pode implementar papéis de extensão a uma classe, assim como aspectos que contribuem para a implementação de uma característica do sistema. Nesse artigo são apresentados os benefícios de se utilizar a abordagem de AML numa LPS de redes sobrepostas. As conclusões do estudo mostraram que a combinação de POA e FOP melhoram a modularidade das *features* na LPS.

Kastner et al. (Kastner et al. 2007) apresentam um estudo de caso que refatora um sistema legado em uma LPS utilizando OA para implementar as *features*. O estudo de caso utilizado foi um sistema de banco de dados chamado Berkeley, implementado com a linguagem Java. O sistema foi decomposto em 38 *features*. No processo de refatoramento do código original para POA foram utilizados refatoramentos já conhecidos na literatura (Hanenberg et al. 2003, Monteiro and Jo 2005). O objetivo desse trabalho foi implementar as *features* com a linguagem AspectJ, a fim de destacar os pontos fortes e fracos da linguagem. Como conclusões desse estudo, eles relataram: (i) um forte acoplamento entre classes e aspectos, tornando a evolução e manutenção da LPS muito mais difícil; e (ii) a sintaxe da linguagem AspectJ pode ser de difícil entendimento e leitura. Os autores relataram também que a linguagem AspectJ adiciona complexidade a aplicação, a qual não está totalmente vinculada a refatoramentos orientados a *features*. Porém, AspectJ pode ser útil para se implementar *features* em uma LPS que foi projetada para ser extensível. Esta característica não está presente no estudo de caso utilizado, pois o Berkeley DB não foi projetado para ter extensibilidade de *features*.

7.3 Estudos Empíricos com Agentes e Aspectos

Garcia et al. (Garcia et al. 2003a) apresentam um experimento que faz uso de duas diferentes técnicas para o desenvolvimento de SMA, que são: desenvolvimento orientado a aspectos e desenvolvimento orientado a padrões. Esse estudo é baseado em um conjunto de atributos de modularidade para avaliar o reuso e a manutenibilidade de alguns interesses transversais de

SMA, tais como: mobilidade, autonomia, interação, colaboração. O sistema utilizado neste trabalho foi o *PortalWare*, que é um ambiente baseado na *Web* para permitir o desenvolvimento e o gerenciamento de portais. Os resultados colhidos mostraram que as abstrações OA permitiram a construção de um SMA com uma melhor separação de interesses. Os autores concluíram também que a OA produziu linhas de código, vocabulário externo e interno dos componentes mais concisos. A técnica orientada a padrões produziu um alto grau de acoplamento devido ao uso do mecanismo de herança. Em termos de reuso e e evolução, a OA apresentou melhores resultados, através das métricas: entidades modificadas, relacionamentos modificados e adicionados, linhas de código adicionadas e modificadas.

Lobato et al. (Lobato et al. 2008) avaliam quantitativamente e qualitativamente os efeitos positivos e negativos da POA a partir de um número de modificações no *framework* MobiGrid. O estudo analisou quatro requisitos de evolução do framework MobiGrid. A evolução do MobiGrid foi observada em duas versões: Java e AspectJ. As mudanças emergiram da necessidade de satisfazer quatro requisitos básicos em cenários de evolução de *frameworks*: (i) isolamento dos interesses de mobilidade existentes; (ii) inclusão de novos interesses de mobilidade; (iii) melhoria nas variabilidades existentes no MobiGrid de certos interesses entrelaçados e não entrelaçados; e (iv) composição do MobiGrid com o JADE, *framework* de mobilidade e comunicação. Os autores concluíram que a versão OA melhorou a modularidade e a estabilidade da arquitetura do MobiGrid para a maioria das mudanças. Além disso, a arquitetura OA permitiu a inclusão de novos interesses transversais de maneira transparente. Na composição de dois *frameworks*, MobiGrid e JADE, o projeto OA conseguiu reduzir a intrusão do código associado aos *frameworks*. Porém, quando são necessárias mudanças em métodos e relacionamentos não transversais no projeto OA, foram observadas algumas instabilidades na arquitetura.

7.4 Diferenças em Relação aos Trabalhos Relacionados

Na Seção 7.1 foram apresentados dois trabalhos na área de LP-SMA. Basicamente, esses trabalhos focam em metodologias para especificação e desenvolvimento de LP-SMA. Apesar de utilizarem a abordagem de LP-SMA, eles não focam no estudo empírico de variabilidades em LP-SMA.

Na Seção 7.2 foram apresentados alguns trabalhos baseados em LPS. O primeiro considera uma abordagem diferente da utilizada neste trabalho, que é a abordagem de LP-SMA e a inclusão de *features* de agentes em cenários de evolução. O segundo apresenta um estudo comparativo onde foram utilizadas

métricas, porém somente as métricas de tamanho foram consideradas, como: número de linhas de código e número de componentes (classes, *mixins* e aspectos) na implementação da LPS. Além disso, esse estudo não considerou cenários de mudanças na implementação da LPS. Por fim, as diferenças em relação ao último trabalho é que a análise da modularização das *features* com OA é feita apenas sobre o ponto de vista qualitativo, ou seja, não foram utilizadas métricas de software para fornecer um estudo mais sistemático.

Na Seção 7.3 foram apresentados dois estudos quantitativos utilizando alguns interesses de agentes de software. Os dois trabalhos não focam no domínio de LP-SMA e não analisam quantitativamente e qualitativamente sobre o ponto de vista da utilização de diferentes plataformas de desenvolvimento de SMA e técnicas de implementação.

Resumidamente, os trabalhos citados não consideram o domínio de LP-SMA e a inclusão de *features* de agentes durante a evolução da LP-SMA. Além disso, os trabalhos não consideram a utilização de diferentes plataformas para o desenvolvimento de SMA (JADE e Jadex) e o uso de diferentes técnicas de implementação das *features* de agentes, tais como: compilação condicional, arquivos de configuração e OA.