# 6
# Discussões e Lições Aprendidas

"Todo aquele que participa de uma discussão defende duas coisas: uma tese e a si mesmo." (Paul Valery)

Neste capítulo são relatadas as discussões e lições aprendidas através da experiência com o desenvolvimento e o processo empírico das LP-SMAs. O objetivo deste capítulo é fornecer lições de como melhor implementar certos tipos de *features* de agentes a partir da experiência no desenvolvimento das LP-SMA e aplicação das métricas de software.

## 6.1
## Benefícios do Estudo

Com o estudo empírico realizado neste trabalho, tentou-se ter uma abrangência em diversos aspectos. Dentre os principais aspectos, pode-se citar:

(i) **utilização de diferentes técnicas de implementação**: a utilização de diferentes técnicas de implementação permitiu uma melhor comparação e abrangência na implementação das *features* de agentes. As técnicas utilizadas possuem características diferentes na manipulação das variações em LP-SMA. A primeira delas é a compilação condicional, que é uma técnica para selecionar determinadas partes do código que devem compilar e excluir outras partes que não devem ser compiladas. De forma geral, pré-processadores indicam quais pedaços de código deveriam ser compilados ou não baseados nos valores das variáveis pré-processadas. A segunda são os arquivos de configuração Spring (Source 2008), que permitem a injeção de dependências dentro de pontos variáveis da arquitetura. Por fim, a última técnica utilizada foi a POA (Kiczales et al. 1997), que é uma técnica utilizada para modularizar *concerns* que interceptam os componentes do software através de uma nova abstração chamada aspecto.

(ii) **utilização de duas plataformas de desenvolvimento de SMA**: no estudo realizado, duas plataformas de desenvolvimento de SMA foram

utilizadas: JADE e Jadex. A abstração de agentes permite o desenvolvimento de sistemas como uma composição de unidades organizacionais, como forma a reduzir o acoplamento dentro do sistema. A plataforma JADE é centrada numa abordagem orientada a tarefas, onde o comportamento do agente é decomposto em partes menores chamadas *Behaviours*. A plataforma Jadex é centrada no modelo BDI (*Beliefs-Desires-Intentions*) e orientada a objetivos.

(iii) **modelagem do estudo empírico**: o processo de definição do estudo empírico seguiu os princípios, técnicas e práticas definidos pela Engenharia de Software Empírica (Wohlin et al. 2000). Modelar o estudo seguindo as práticas definidas pela Engenharia de Software empírica faz com que o estudo tenha uma maior completude e consistência.

(iv) **domínios de estudos**: as duas LP-SMA desenvolvidas e utilizadas no processo experimental focaram no domínio *Web*. O objetivo de realizar estes estudos no domínio *Web* se dá ao atual crescimento das aplicações e a facilidade de interação dos usuários com estas aplicações. Além disso, as arquiteturas das LP-SMA foram intencionalmente semelhantes, com o objetivo de realizar um estudo focado em uma arquitetura específica de LP-SMA para o domínio *Web* e derivar um padrão arquitetural (Nunes et al. 2008b).



## 6.2 Discussões e Lições Aprendidas

Essa seção discute e analisa de acordo os resultados obtidos no Capítulo 5, algumas lições aprendidas com o uso das técnicas de implementação e plataformas de SMA utilizadas.

### 6.2.1 Implementação de Papéis

Na implementação do papel Revisor na LP-SMA do EC, a solução OA foi mais efetiva que a solução OO usando técnicas de compilação condicional. Com relação à inclusão de papéis aos agentes de usuário, a solução OA apresentou um maior número de componentes e operações (métricas CDC e CDO). Porém, diminuiu bastante o entrelaçamento com outros *concerns* (métrica CDLOC).

Com relação à inclusão de novos papéis nas *releases* da LP-SMA OLIS, a solução Jadex apresentou melhores resultados em termos das métricas CDC, CDO e número de atributos, considerando os valores absolutos (R4, R5 e R6). A implementação dos agentes de software são geralmente conseguidas pelo uso de plataformas de agentes, tais como, JADE (agentes são implementados

com classes Java) e Jadex (agentes são implementados com arquivos XML e classes Java de acordo com a especificação das capacidades dos agentes). A implementação de papéis na plataforma Jadex tem mostrado que uma boa estrutura de *framework* OO pode prover abstrações chaves para os desenvolvedores, as quais apresentam bons resultados em termos das modularizações de *features* em LP-SMA. A estrutura de *frameworks* OO pode também ser enriquecida para prover uma completa separação de papéis usando abstrações de OA, como demonstrado na nossa versão da LP-SMA OLIS. Isto contribui para evitar os problemas relacionados com o entrelaçamento de *features*, apresentados nas Seções 5.1.4 e 5.2.4.

Sendo assim, de forma geral, pode-se dizer que a implementação de papéis pode ser melhor utilizando a plataforma Jadex e arquivos de configuração. Os aspectos positivos que favorecem a implementação de papéis com Jadex são a facilidade de evolução, devido a presença de melhores resultados em termos das métricas de modularidade.



### 6.2.2 Modularidade das features Alternativas

A inclusão de *features* alternativas na LP-SMA OLIS ocorreu nas *releases* R3 e R8. Na R3, todas as soluções apresentaram os mesmos valores, como visto na Seção 5.2.4. Porém, a inclusão da *feature* para permitir eventos de viagem apresentou valores diferentes para as implementações. Considerando as diferentes métricas de *concern* na R8, a implementação JADE foi melhor para a métrica CDC; Jadex foi melhor para a métrica CDO e número de atributos; e a solução OA foi melhor para a métrica CDLOC. Estes resultados enfatizam que não existe uma clara vantagem para usar uma solução específica para implementar as *features* de agentes alternativas. Mesmo assim, para a implementação das *features* alternativas na LP-SMA OLIS, acreditamos que as implementações JADE e Jadex OO são soluções mais adequadas. Essas conclusões são fundamentadas nos resultados obtidos para muitas das métricas coletadas. Adicionalmente, o uso de herança e polimorfismo na implementação nos tipos de eventos alternativos representam uma forma natural para modularizar este tipo de *feature*, a qual já é bem entendida e largamente adotada na comunidade de desenvolvimento. Também provê um bom suporte a lidar com cenários de evolução, como demonstrado no estudo.

### 6.2.3 Estabilidade das Implementações

As implementações utilizando JADE, Jadex e JADE com técnicas de OA são comparáveis porque elas representam diferentes formas de se implementar *features* de agentes. Nas métricas de estabilidade da LP-SMA do EC, a solução OA em linhas gerais se comportou da seguinte forma: apresentou uma maior quantidade de componentes adicionadas para todas as *releases*, linhas de código adicionadas e operações adicionadas. De acordo com os resultados obtidos para as métricas de impacto de mudanças na LP-SMA OLIS, as implementações apresentaram as seguintes propriedades de estabilidade: (i) a implementação OA foi superior (menor valor) às outras implementações em termos de linhas de código adicionadas, operações modificadas e CDLOC para muitas das *releases* (e.g. R2 and R4); (ii) a implementação Jadex foi superior às outras implementações para todas as *releases* em termos de operações adicionadas; e (iii) a implementação JADE foi superior, ou pelo menos apresentou resultados similares as outras implementações. Os resultados reforçam que ótimos resultados podem ser conseguidos quando se utiliza de uma plataforma de SMA, tal como Jadex. Por outro lado, os resultados de boa estabilidade conseguidos pela implementação OA motiva a investigação da possibilidade de integração de POA na plataforma Jadex para reagir melhor em cenários de manutenção e evolução.

Na solução OA, as mudanças geralmente causam a propagação em muitos componentes não relacionados, enquanto que a solução OO exige mais mudanças extensivas, fazendo a modificação nos componentes existentes mais óbvia e clara. A solução OA sempre apresenta mais componentes, operações e linhas de código, devido à característica de permitir mudanças não invasivas. Na solução OO com técnicas de compilação condicional, as mudanças são diretamente feitas nos componentes existentes, aumentando assim o entrelaçamento entre as *features*. Já com a técnica de arquivos de configuração, a utilização de polimorfismo também provoca um aumento no número de componentes e operações.

### 6.2.4 Validação com Testes Estatísticos

Apesar dos resultados absolutos terem mostrado os benefícios de implementações específicas comparada com outras (Seção 5.2.4), isto não foi suficiente para dizer qual solução foi melhor em termos de atributos de modularidade e manutenibilidade. As diferenças de valores não foram tão significantes para generalizar os benefícios de uma determinada solução. O teste não parame-

trizado de Kruskal-Wallis (Wohlin et al. 2000) foi aplicado para demonstrar que as três implementações tem o mesmo grau de modularidade e manutenibilidade. Assim, de acordo com os resultados do teste aplicado, o uso de um projeto OO bem modularizado em combinação com mecanismos de injenção de dependências, tais como arquivos de configuração Spring, devem ser suficientes em muitos casos para suportar o gerenciamento de variabilidade modular. Na LP-SMA OLIS, para muitos dos dados coletados, o uso da plataforma JADE com técnicas de OA para implementar algumas *features* de agentes apresentou melhores resultados quando comparadas com as implementações Jadex e Jade OO. Adicionalmente, a necessidade para gerenciar componentes adicionais (classes e arquivos de configuração) com mais linhas de código nestas implementações OO poderiam ser resolvidos com o uso de geradores de código simples e linguagens específicas de domínio que provém ajuda à instanciação de *frameworks*.



### 6.2.5 Evolução das LP-SMA com Orientação a Aspectos

Muitas das *features* de agentes são implementadas por um conjunto de diferentes componentes, agentes e classes. Estas features são caracterizadas como *features* transversais (*crosscutting features*), porque seu projeto e implementação estão espalhados e entrelaçados por diferentes componentes do sistema. Durante o desenvolvimento e evolução das LP-SMA, diversos aspectos auxiliaram na integração entre a arquitetura base da LPS e as diferentes *features* obrigatórias, opcionais e alternativas para implementação das variabilidades de agentes. Esta vantagem da POA permite que a inclusão de código ocorra de maneira não invasiva, pois não há alteração direta no código da arquitetura base da LPS. Diversos aspectos nas LP-SMAs funcionaram como código "cola" entre a estrutura OO e as diferentes *features* de agentes adicionadas ao núcleo. A decisão de projeto é muito útil porque permite a injeção de novas propriedades e comportamentos de uma forma transparente na estrutura OO. Além disso, a modularização das *features* em diversos componentes aspectuais facilita a ativação/desativação automática de diferentes comportamentos autônomos implementados para o sistema. A integração dos agentes de software com a arquitetura base da LPS, através de uma versão OA do padrão *Observer* (Gamma et al. 1995) permitiram essa integração com o mínimo impacto possível.

A inclusão dos agentes de software e os papéis foram possíveis, pois os aspectos interceptaram algumas classes adicionando funcionalidades específicas. Com relação à integração dos agentes com o sistema, a implementação do pa-

drão *Observer* com POA permitiu a interceptação dos métodos das classes de serviços. De forma geral, a POA mostrou bons resultados com relação as métricas de estabilidade e separação de *concerns*, mais especificamente a métrica que mede o entrelaçamento (CDLOC). Por outro lado, as soluções com OA apresentaram mais componentes e operações na implementação das *features*.

### 6.2.6 Gerenciamento das Features

Durante a evolução da LP-SMA do EC, a inclusão das *features* de agentes causou um alto número de novos componentes (classes e aspectos) na solução OA. Exemplos na LP-SMA do EC são a inclusão dos agentes de usuário e do agente notificador. Embora o uso de OA aumente o número de componentes nestes casos, ela foi útil para reduzir o entrelaçamento e o acoplamento entre os *concerns/features*. Os resultados obtidos para as métricas de tamanho mostraram que embora a implementação OA na LP-SMA do EC tenha melhorado a modularização das *features* de agentes, o alto número nas métricas NOC, LOC e NOO pode trazer dificuldades para o entendimento e evolução da LPS, dificultando assim o seu gerenciamento.

Na evolução da LP-SMA OLIS, a implementação Jadex com arquivos de configuração apresentou sempre um maior número de componentes para implementação das *features* de agentes. Esse número alto de componentes é devido a um alto número de planos e arquivos XML para representar os agentes. Como dito anteriormente, esse número significativo de componentes pode dificultar a evolução da LP-SMA.

### 6.2.7 Estratégias de Adoção de Linhas de Produto de Software

Como discutido no Capítulo 2, existem três estratégias para desenvolvimento de LPS, são elas: pró-ativa, extrativa e reativa. Embora somente a estratégia reativa tenha sido utilizada no desenvolvimento das LP-SMAs, os resultados do estudo podem ser estendidos para os outros dois tipos de desenvolvimento de LPS. Na abordagem extrativa, as diretrizes de desenvolvimento podem ser úteis para se escolher o tipo de técnica de implementação e a plataforma de desenvolvimento de SMA que melhores se aplicam no processo de refatoramento de uma determinada *feature*. Finalmente, na abordagem pró-ativa, que leva em consideração todos os produtos já existentes, as técnicas de implementação e a plataforma são escolhidas logo no início do desenvolvimento da LPS, considerando os tipos de *features* que existem nesses tipos de produtos.

## 6.3 Resumo

Neste capítulo foram discutidos os benefícios e lições aprendidas a partir da experiência na realização de dois experimentos com duas LP-SMAs. Inicialmente, foram apresentados os benefícios da realização deste estudo, reforçando as principais características do estudo. Por fim, o capítulo foi finalizado com as discussões e lições aprendidas com este estudo no que se refere: (i) implementação de papéis; (ii) modularidade das implementações; (iii) estabilidade das implementações; (iv) validação com testes estatísticos; (v) evolução das LP-SMAs com orientação a aspectos; (vi) gerenciamento das *features*; e (vii) estratégias de adoção de linhas de produto de software.