8
O estudo quantitativo

Com o aumento de tamanho e complexidade dos sistemas multiagentes (SMAs), a separação de seus concerns ao longo das diferentes fases de desenvolvimento é crucial para os engenheiros. Assim, é necessário investigar sistematicamente se as abstrações da orientação a objetos e a orientação a aspectos conseguem isolar de forma explícita os concerns do SMA. Este capítulo apresenta um estudo empírico quantitativo que avaliou o grau com que essas abstrações permitem a modularização desses concerns. Esse experimento concentrou-se na comparação quantitativa entre a abordagem orientada a aspectos proposta (Capítulos 4 e 5) e uma abordagem orientada a padrões (Seção 7.2.1).

Diferente da avaliação qualitativa, o objetivo geral dessa atividade empírica era avaliar o suporte à manutenção e reutilização das abordagens investigadas. Além disso, esse estudo foi planejado como um experimento mais controlado [17], e alguns procedimentos foram considerados a fim de minimizar problemas comuns em estudos empíricos [17]. A avaliação das duas abordagens foi conduzida usando o framework de avaliação (Capítulo 6) e a metodologia de GQM de Basili [16].

Os resultados coletados demonstraram que o uso da abordagem orientada a aspectos permitiu a construção de um SMA com um melhora significativa na modularização de diferentes concerns. Além disso, o uso de aspectos resultou em: (i) menos linhas de códigos, (ii) menos componentes de implementação e projeto (iii) menor acoplamento entre os componentes. Contudo, a abordagem orientada a aspectos produziu menos coesão nos componentes do SMA. Finalmente, uma importante descoberta desse estudo empírico foi que os aspectos oferecem suporte a um melhor alinhamento com níveis mais alto de abstrações para modelos orientados a agentes.

Os resultados reunidos neste estudo oferecem uma compreensão clara dos pontos fortes e fracos das duas abordagens investigadas e sua compatibilidade e

divergências. Os resultados também são importantes fontes em direção à convergência potencial de técnicas orientadas a agentes e orientadas a objetos. Também são úteis para engenheiros de SMAs realísticos que precisam implementar os modelos orientados a agentes usando linguagens de programação orientada a objetos. As conclusões também podem ser do interesse dos metodologistas orientados a agentes uma vez que podem decidir incorporar soluções para os problemas detectados nesse estudo diretamente como parte de suas metodologias.

Este estudo quantitativo foi publicado como um capítulo de livro [98]. O restante deste capítulo está organizado da seguinte forma. A Seção 8.1. descreve as abordagens investigadas neste estudo experimental. A Seção 8.2 apresenta a organização do estudo em termos da metodologia usada. A Seção 8.3 descreve as questões e os objetivos do estudo. A Seção 8.4 apresenta a hipótese. A Seção 8.5 discute o projeto do SMA usado no experimento. A Seção 8.6 descreve os sujeitos envolvidos e as fases do estudo. A Seção 8.7 apresenta os resultados do estudo, interpretados e discutidos na Seção 8.8, com base na hipótese apresentada. Já a Seção 8.9 discute os trabalhos relacionados. Finalmente, a Seção 8.10 inclui algumas conclusões e direções para trabalhos futuros.

8.1
As abordagens investigadas

Esse estudo empírico concentrou-se na comparação da abordagem orientada a aspectos (Capítulos 4 e 5) e da mesma abordagem orientada a padrões usada no estudo qualitativo (Seção 7.2.1). A abordagem orientada a padrões foi escolhida porque os padrões de projeto são reconhecidos como abstrações essenciais para construir sistemas orientados a objetos de alta qualidade (Seção 2.1). Além disso, eles incorporam soluções para melhorar a separação de concerns, reduzir o acoplamento, aumentar a coesão e promover a reutilização e a manutenção.

A diferença básica entre as abordagens investigadas é que a abordagem orientada a padrões está concentrada no uso de padrões orientados a objetos, e a abordagem orientada a aspectos está concentrada no uso de abstrações orientadas a aspectos. Os sujeitos experimentais usaram as abordagens para ajudar no

desenvolvimento de um sistema multiagentes (Seção 8.5). São três os propósitos das duas abordagens: (1) promover a separação de concerns de agência, (2) oferecer suporte à construção de SMAs grandes e com reusabilidade e manutenibilidade e (3) minimizar os desalinhamentos entre os modelos baseados em agentes de alto nível e a implementação e o projeto orientado a objetos.

8.2
A metodologia

A avaliação das abordagens da engenharia de software é uma tarefa notoriamente complicada. Há poucas metodologias estabelecidas para o planejamento de medições e a coleta de dados. A organização do experimento baseava-se em uma aplicação complementar da metodologia GQM (Goal/Question/Metric – Objetivo/Questão/Métrica) [16] e o framework de avaliação (Capítulo 6). A GQM foi usada para estruturar o experimento em termos de seus objetivos, e o framework de avaliação foi usado para extrair as qualidades, fatores e critérios investigados neste experimento.

Ela foi selecionada para avaliar as abordagens investigadas uma vez que ganhou uma ampla popularidade e suporte dentro da comunidade da engenharia de software. O paradigma da GQM foi proposto como uma abordagem orientada a objetivos para a medição de produtos e processos. Essa metodologia baseia-se na premissa de que, para ganhar uma medida prática, deve-se primeiro entender e especificar os objetivos dos artefatos de software sendo medidos e os objetivos do processo de medição. A abordagem GQM oferece um framework que envolve três etapas: (i) listar os principais objetivos do estudo empírico; (ii) deduzir a partir de cada objetivo as questões que devem ser respondidas para determinar se os objetivos foram alcançados e (iii) decidir o que pode ser medido a fim de responder as questões de forma adequada. A Seção 8.3 define as questões e os objetivos.

8.3
Objetivos e questões

O objetivo geral deste estudo é avaliar a manutenibilidade e a reusabilidade das duas abordagens investigadas no contexto do SMA. Esse objetivo foi redefinido

em um conjunto de questões, que representam sua definição operacional. A idéia é gerar quantas questões possíveis, incluindo as redundantes ou inválidas. Com o seguimento do processo, desenvolvemos um conjunto hierárquico de questões que foram estreitadas em seguida. Para cada questão, foram associadas métricas relevantes (Capítulo 6). A Figura 84 apresenta uma amostra das questões geradas. Um relatório [83] apresenta todos os objetivos e questões restantes depois do refinamento. A Tabela 6 mostra como as questões são associadas a métricas e atributos internos.

Objetivo

Avaliar a facilidade da evolução e da reusabilidade dos sistemas multiagentes implementados a fim de comparar a abordagem orientada a padrões e a abordagem orientada a aspectos.

Questões

1. Qual a facilidade encontrada ao desenvolver o sistema?

1.1.Quão fácil é entender o sistema?
- 1.1.1 Quão conciso é o sistema?
  - 1.1.1.1 Há quantos componentes?
  - 1.1.1.2. Há quantas linhas de código?
  - 1.1.1.3. Há quantos atributos?
  - 1.1.1.4. Há quantos métodos e advices?
- 1.1.2 Com que facilidade os concerns são localizados?
  - 1.1.2.1 Qual o entrelaçamento e o espalhamento da definição <concern1 name>?
  - 1.1.2.N Qual o entrelaçamento e o espalhamento da definição <concernN name>?
- 1.1.3 Quão alto é o acoplamento do sistema?
  - 1.1.3.1. Quão alto é o acoplamento entre os componentes?
- 1.1.4. Quão alta é a coesão do sistema?
  - 1.1.4.1. Quão alta é a coesão dos componentes do sistema?

1.2 Qual a flexibilidade do sistema?
- 1.2.1 Com que facilidade os concerns são localizados?
  - 1.2.1.1. Qual o entrelaçamento e o espalhamento da definição <concern1 name>?
  - 1.2.1.N. Qual o entrelaçamento e o espalhamento da definição <concernN name>?
- 1.2.2 Quão alto é o acoplamento do sistema?
  - 1.2.2.1. Quão alto é o acoplamento entre os componentes?
- 1.2.3. Quão alta é a coesão do sistema?
  - 1.2.3.1. Quão alta é a coesão dos componentes do sistema?

2. Qual a facilidade encontrada ao reutilizar os elementos do sistema?

2.1. Quão fácil é entender o sistema?
- 2.1.1. Quão conciso é o sistema?
  - 2.1.1.1. Há quantos componentes?
  - 2.1.1.2. Há quantas linhas de código?
  - 2.1.1.3. Há quantos atributos?
  - 2.1.1.4. Há quantos métodos e advices?
- 2.1.2. Com que facilidade os concerns são localizados?
  - 2.1.2.1. Qual o entrelaçamento e o espalhamento da definição <concern1 name>?
  - 2.1.2.N. Qual o entrelaçamento e o espalhamento da definição <concernN name>?
- 2.1.3. Quão alto é o acoplamento do sistema?
  - 2.1.3.1. Quão alto é o acoplamento entre os componentes?
- 2.1.4. Quão alta é a coesão do sistema?
  - 2.1.4.1. Quão alta é a coesão dos componentes do sistema?

2.2. Qual a flexibilidade do sistema?
- 2.2.1. Com que facilidade os concerns são localizados?
  - 2.2.1.1. Qual o entrelaçamento e o espalhamento da definição <concern1 name>?
  - 2.2.1.N. Qual o entrelaçamento e o espalhamento da definição <concernN name>?
- 2.2.2. Quão alto é o acoplamento do sistema?
  - 2.2.2.1. Quão alto é o acoplamento entre os componentes?
- 2.2.3. Quão alta é a coesão do sistema?
  - 2.2.3.1. Quão alta é a coesão dos componentes do sistema?

Figura 84. Objetivo e questões.

8.4
Hipótese

A hipótese deve ser testada conforme descrito a seguir: “a abordagem orientada a aspectos proposta oferece um melhor suporte à manutenibilidade e a reusabilidade de SMAs do que a abordagem orientada a padrões.” Essa hipótese baseia-se em um estudo qualitativo, conduzido previamente [82] (Seção 7.2). Também está diretamente relacionada ao objetivo do experimento (Seção 8.3).

Metrics	Answered Questions	Criteria
Vocabulary Size (VS)	How many components are there?	Size
Lines of Code (LOC)	How many lines of code are there?	Size
Number of Attributes (NOA)	How many attributes are there?	Size
Weighted Operations per Component (WOC)	How many methods and advices are there?	Size
Coupling Between Components (CBC)	How high is the coupling between components?	Coupling
Lack of Cohesion in Operations (LCOO)	How high is the cohesion of the systems components?	Cohesion
Depth of Inheritance Tree (DIT)	How high is the coupling between components? How high is the cohesion of the systems components?	Coupling and Cohesion
Concern Diffusion over Components (CDC)	How well are the agency concerns localized? (and sub-questions)	Separation of Concerns
Concern Diffusion over Operations (CDO)	How well are the agency concerns localized? (and sub-questions)	Separation of Concerns
Concern Diffusion over LOC (CDLOC)	How well are the agency concerns localized? (and sub-questions)	Separation of Concerns

Tabela 6. Métricas, questões GQM e atributos internos

8.5
O projeto do SMA

O Portalware foi o projeto de SMA usado neste estudo quantitativo. Ele foi descrito na Seção 4.1. Os concerns do SMA tratados neste projeto são os típicos de muitos domínios de aplicações existentes de SMAs reativos do mundo real. Conforme previamente mencionado (Seção 4.1), esse SMA incorpora vários concerns de agência, incluindo tipos de agente, papéis, colaboração, interação, adaptação, autonomia etc. Esse ambiente inclui alguns tipos de agente. A modelagem do sistema é baseada na linguagem de modelagem de Elammari [66] e no framework de modelagem TAO [223]. TAO foi usado porque extrai as abstrações comuns usadas

no projeto e na análise do SMA. A linguagem Java e as notações UML [24] foram respectivamente usadas para gerar a implementação e os projetos orientados a padrões. A extensão da UML para o projeto orientado a aspectos [41, 43] (Seção 2.2.2) e a linguagem de programação AspectJ [140] foram usadas para gerar implementações e projetos orientados a aspectos.

Os sujeitos do experimento desenvolveram duas versões do sistema Portalware, usando as abordagens orientadas a padrões e a aspectos. As Figuras 85 e 86 representam respectivamente porções dos projetos orientados a aspectos e padrões do SMA do Portalware. A Figura 85 mostra uma combinação de diferentes padrões de projeto para cuidar dos concerns de SMAs. Cada padrão está marcado por uma linha pontilhada. Na Figura 86, foi usada uma forma de losango para expressar os aspectos. Cada losango pode estar relacionado a um ou mais retângulos usados para descrever as classes. Esse relacionamento é expresso como uma linha do aspecto para a classe. Essas figuras também ilustram algumas alterações necessárias para a manutenção e reutilização dos cenários para um futuro esclarecimento na próxima seção.

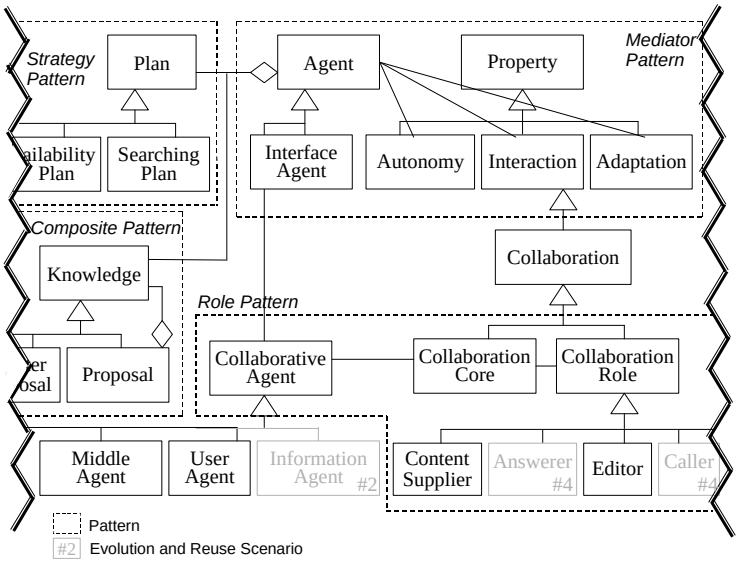

Figura 85. Um subconjunto do projeto orientado a padrões do SMA.

8.6
Os sujeitos e as fases do estudo

Quatro sujeitos participaram do estudo. Três deles eram candidatos a Ph.D e um era estudante de mestrado da PUC-Rio. Todos fizeram parte do desenvolvimento dos dois sistemas. Eles participaram do desenvolvimento do sistema orientado a aspectos (OA) e do desenvolvimento do sistema orientado a padrões (OP). Todos tinham experiência na análise de software OO, projeto orientado a padrões e programação de SMAs. Os candidatos a Ph.D já tinham implementado programas Java grandes (> 10 mil linhas de código). Entre eles, dois tinham bastante experiência na programação orientada a aspectos. Eles relataram problemas relacionados ao uso das abordagens investigadas e as ferramentas e os métodos associados. O conjunto de modelos orientados a agentes, o projeto detalhado e a implementação baseavam-se nas mesmas especificações de requisito e devem satisfazer o mesmo conjunto de cenários.

O estudo foi dividido em duas fases principais: (1) a fase de Construção e (2) a fase de Reutilização e Evolução. Na fase de Construção, os indivíduos deveriam desenvolver o SMA selecionado (Seção 8.5) usando as abordagens investigadas (Seção 8.1). A fase de Reutilização e Evolução envolveu os mesmos sujeitos. O objetivo dessa fase era comparar a reusabilidade e a manutenibilidade da solução orientada a padrões e a solução orientada a aspectos. Para avaliar a modificabilidade e a extensibilidade dos sistemas produzidos, foi desenvolvido um conjunto de cenários de alterações e reutilizações para o código e os projetos originais. Como algumas métricas (Seção 6.4) são orientadas à quantidade de componentes do projeto e de linhas de código, esse estudo envolveu uma outra fase de padronização antes da coleta de dados. Ela objetivava garantir que os dois SMAs desenvolvidos implementassem as mesmas funcionalidades. Essa fase também eliminou problemas relacionados a diferentes estilos de codificação.

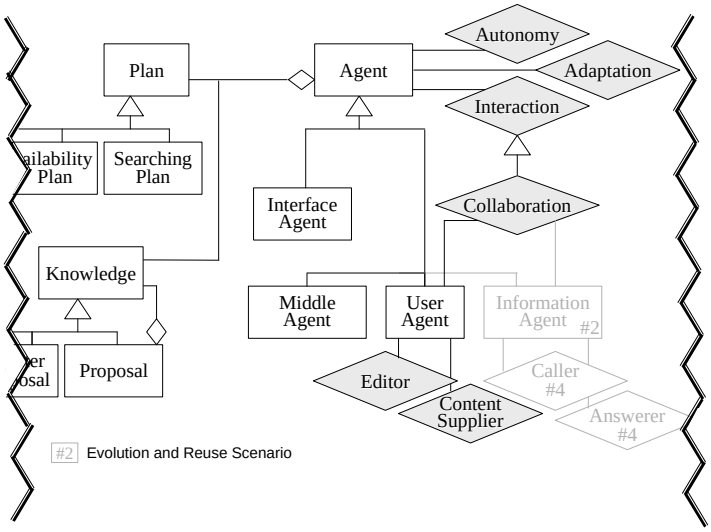

Figura 86. Um subconjunto do projeto orientado a aspectos do SMA.

Na fase de Reutilização e Evolução, simulamos alterações simples e complexas que envolviam concerns de agência para as soluções OP[8] e OA[9] a fim de medir seu suporte à modificabilidade e extensibilidade. Selecionamos 7 cenários de reutilização e alteração recorrentes em SMAs de larga escala, como a inclusão de novos agentes, a reutilização de papéis e capacidades colaborativas etc. A lista de cenários é esta:

C1) Alteração nos papéis do agente (Evolução)

C2) Criação de um tipo de agente (Evolução)

C3) Reutilização do concern de agência (Reutilização)

C4) Inclusão de colaboração em um tipo de agente (Reutilização e Evolução)

C5) Reutilização de Papéis (Reutilização)

C6) Criação de uma nova instância do agente (Evolução)

C7) Alteração da definição de agência (Reutilização e Evolução)

Para cada alteração feita aos concerns do sistema, a dificuldade da modificabilidade foi definida em termos dos itens a seguir: (1) número de componentes (aspectos/classes) adicionados, (2) número de componentes alterados, (3) número de relacionamentos incluídos, (4) número de relacionamentos alterados,

[8] Orientada a padrões
[9] Orientada a aspectos

(5) número de novas LOCs (6) número de LOCs modificadas, (7) número de operações (métodos/advices) adicionadas e (8) número de operações (métodos/advices) alteradas.

Para cada tentativa feita de reutilizar algum concern, a dificuldade de extensibilidade era medida com base ns itens de modificabilidade e nos itens a seguir: (9) número de entidades copiadas e (10) número de LOCs copiadas. Todos esses itens foram observados sob o ponto de vista de códigos e modelos de projeto comportamental e estrutural. Esses elementos no desenvolvimento orientado a aspectos e aqueles no desenvolvimento orientado a padrões são comparáveis porque representam os mesmos concerns de modelos de agente de alto nível.

8.7
Avaliação das abordagens

Esta seção apresenta os resultados do processo de medição. A Seção 8.7.1 apresenta uma visão geral dos resultados obtidos no final da fase de construção de SMAs. Os dados foram coletados com base no conjunto de métricas definidas (Seção 6.4). A descrição completa dos dados coletados na fase de construção está relatada em [83]. A Seção 8.7.2 descreve os resultados da fase de reutilização e evolução do SMA com base nos cenários selecionados e as métricas associadas (Seção 8.6). A discussão sobre os resultados é apresentada na Seção 8.8.

8.7.1
A fase de construção do SMA

Os dados foram parcialmente reunidos pela ferramenta do CASE Together 6.0 [25] (Figura 87). Ela oferece suporte a algumas métricas: LOC, NOA, OOC (WMPC2 em Together), CBC (CeO em Together), LCOO (LOCOM1 em Together) e DIT (DOIH em Together). A Figura 88 apresenta os resultados gerais dos projetos OA e OP para todas as métricas, excluindo-se as métricas da separação de concerns. Esses resultados foram reunidos a partir do ponto de vista do sistema. O número de componentes do sistema no projeto OP foi 7% maior do que no projeto OA (métrica VS). Os cálculos das linhas de código (LOC) e o número de atributos (NOA) para o

SMA desenvolvido na implementação OP eram respectivamente 12% e 19% maior do que no código OA. O projeto OA também produziu resultados melhores em termos da complexidade das operações (6%), acoplamentos de componentes (9%) e coesão de componentes (3%). A DIT máxima era cinco para o projeto OP e três para o projeto OA, ou seja, 40% menor do que no projeto OP.

As medições absolutas reunidas a partir da perspectiva do sistema (Figura 88) mostram diferenças significativas em favor da solução baseada em aspectos. No entanto, nem sempre o resultado foi o mesmo ao analisá-lo sob diferentes pontos de vista. As subseções a seguir apresentam em detalhes os resultados das métricas de tamanho, acoplamento, coesão e separação de concerns.

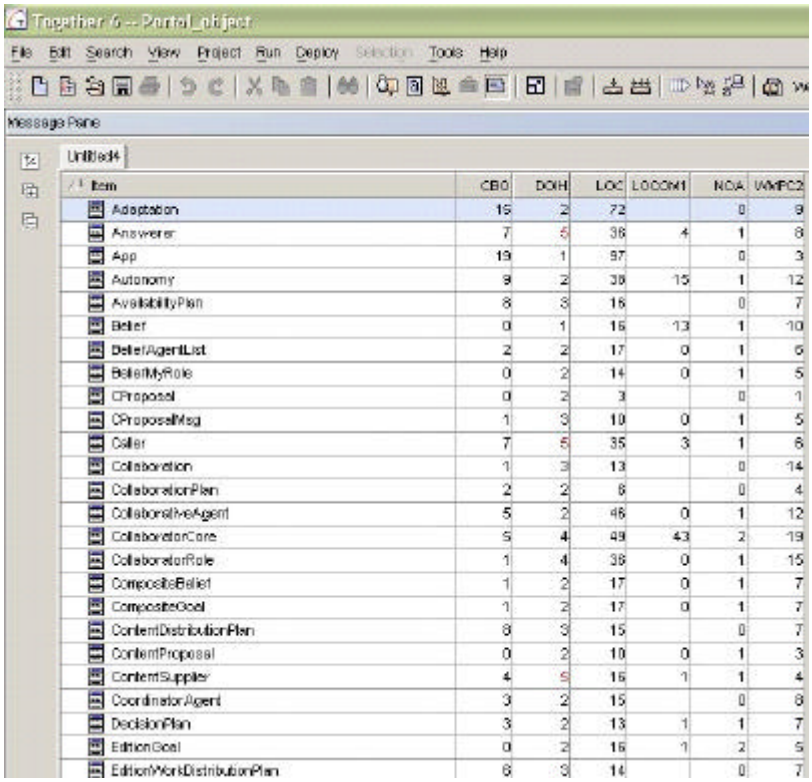

Figura 87. Uso do Together [25] para aplicar as métricas definidas.

Resultados da métrica de tamanho

Tamanho do vocabulário. O VS conta o vocabulário externo do sistema, ou seja, o número de componentes (classes/aspectos) dentro do sistema (Seção 6.4.4). O vocabulário externo do SMA orientado a aspectos é mais simples do que no SMA orientado a padrões, uma vez que a quantidade de componentes de implementação e

projeto no último (VS = 60) foi maior do que no primeiro (VS = 56). A principal razão para esse resultado é que os padrões Papel e Mediador requerem outras classes para tratar da decomposição e composição de vários papéis do agente e propriedade do agente, respectivamente. A solução orientada a aspectos não precisa dessas classes adicionais, porque a composição é especificada pelos pointcuts, definidos internamente para os aspectos.

Número de Atributos. Essa métrica conta o vocabulário interno de cada componente, ou seja, o número de atributos de cada classe ou aspecto (Seção 6.4.4). Os atributos herdados não são incluídos no cálculo. O vocabulário interno dos componentes da solução OP é mais complexo do que nos componentes OA. O número de atributos dos componentes do SMA na solução OP foi maior do que na solução OA. Isso porque os objetos Agent no projeto OP precisam ter referências explícitas aos objetos que representam três propriedades básicas do agente e os papéis do agente. Por exemplo, a classe Agent (a classe que encapsula o comportamento e a estrutura de agência) é composta por 9 atributos no sistema OP, enquanto no sistema AO são 6 atributos. Além disso, o concern de colaboração é modularizado diretamente pelos aspectos Collaboration e Role, enquanto se espalha por 5 classes diferentes (Collaboration, CollaborationCore, CollaborationRole, CollaborativeAgent e Role) no projeto OP e, como conseqüência, esse é um fator que aumenta o NOA porque cada uma dessas classes precisa de referências à outra.

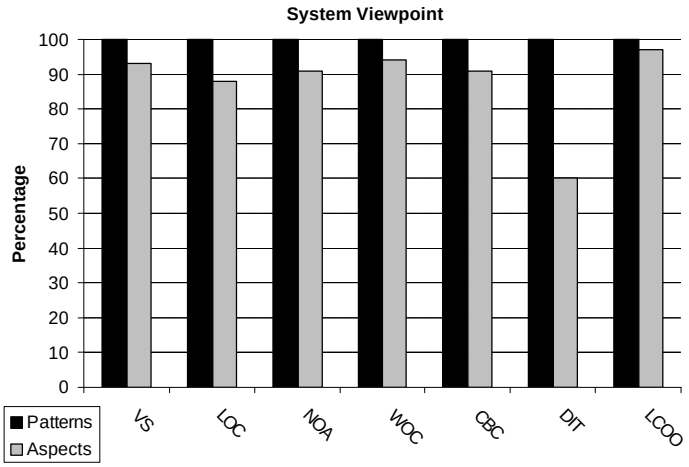

Figura 88. Comparação dos resultados dos dois projetos.

Linhas de Código. A LOC foi 1445[10] na implementação OP e 1271 na implementação OA, ou seja, o código OP possui 174 linhas a mais do que o código OA. Em geral, a implementação de cada par (plano, papel) no código OP incluiu 10 linhas a mais do que o código OA. Isso porque a solução OP precisa de outras chamadas a métodos nas subclasses Role para: (i) ativar e desativar papéis e (ii) obter referências aos objetos do papel. Essas chamadas não são necessárias no projeto orientado a aspectos porque a composição é especificada pelos pointcuts nos aspectos de papéis. Ademais, foram necessárias menos linhas de código para implementar o concern de colaboração no sistema OA uma vez que o sistema OP implementa outras classes do padrão Papel, conforme descrito anteriormente.

Operações Consideradas por Componente. Essa métrica mede a complexidade de um componente em termos de suas operações (Seção 6.4.4). Quanto maior o número e a complexidade das operações do componente, mais difícil é entender o sistema. Em geral, essas medidas ressaltaram mais OCC na solução OA. Isso acontece porque a modularização de alguns concerns nos aspectos requer que o contexto seja recapturado por meio de join points e pointcuts. Alguns exemplos das medidas de OCC são: (i) 9 para a classe Adaptation na solução OP em comparação com 10 para o aspecto Adaptation na solução OA, (ii) 12 para a classe Autonomy em comparação com 14 para o aspecto Autonomy e (iii) 17 para a classe Interaction em comparação com 25 para o aspecto Interaction. A situação recorrente nesses exemplos é que os aspectos Interaction, Adaptation e Autonomy recebem a referência do objeto do agente como parâmetros nos pointcuts, enquanto essa referência é acessada em um atributo local nas respectivas classes do projeto OP.

Resultados da métrica de coesão e acoplamento

Profundidade da Árvore de Herança. A DIT é definida como o comprimento máximo de um nó até a raiz da árvore (Seção 6.4.2). Há muitos problemas no projeto OP ao considerar essa métrica. O uso do padrão papel leva a uma hierarquia de 5

[10] O cálculo não inclui o código relacionado aos concerns de auditoria e rastreamento e a implementação dos cenários apresentados na Seção 8.6. Ele também não inclui a implementação das funcionalidades presentes apenas em uma das versões do Portalware. O cálculo absoluto é 2763 LOCs no sistema OO e 2168 LOCs no sistema AO.

níveis para estruturar os papéis do agente, o que leva potencialmente a um alto acoplamento de herança. Na solução OA, o DIT = 1 porque os papéis são encapsulados nos aspectos que afetam a hierarquia do tipo de agente. Além disso, o uso do padrão mediador leva a uma hierarquia de 3 níveis para estruturar os tipos de agente, enquanto a mesma hierarquia no sistema OA possui 2 níveis. Isso porque é necessário criar um outro nível (a classe CollaborativeAgent) na solução OP a fim de separar os agentes colaborativos dos não-colaborativos. Na solução OA, essa separação é alcançada de forma transparente pelo aspecto de colaboração que define quais tipos de agente são colaborativos e, como conseqüência, não afeta o tamanho da hierarquia do tipo de agente.

Acoplamento entre Componentes. Essa métrica é definida para uma classe ou aspecto como um cálculo do número das demais classes e aspectos aos quais está acoplado (Seção 6.4.2). Há uma diferença significativa entre a CBC e as duas soluções da classe Agent. A métrica CBC é 12 no sistema OP e 9 na solução OA. Dessa vez, essa diferença ocorre porque a classe Agent precisa ter referências explícitas às classes que representam as propriedades de agente básicas. O contrário também é necessário, ou seja, as classes Autonomy, Interaction e Adaptation requerem referências explícitas ao objeto do agente. No projeto OA, apenas os aspectos fazem referência às classes Agent.

Falta de Coesão nas Operações. Essa métrica mede a coesão de um componente (Seção 6.4.3). Um valor baixo para LCOO indica alto acoplamento entre os métodos (ou sejam alta coesão), o que é uma situação desejável. Componentes com baixa coesão sugerem um projeto inapropriado, porque determina o encapsulamento de entidades de programas não relacionadas e que não devem ficar juntas [15]. Também indica potencialmente menos componentes com reusabilidade e manutenibilidade. Muitos componentes do sistema OP produziram resultados melhores em termos de coesão do que os componentes do sistema OA. Por exemplo, a LCOO para a classe Agent é 50 no projeto OP e 57 no projeto OA. A LCOO para o concern de interação é 29 no projeto OP e 48 no projeto OA.

Resultados da métrica de separação de concerns

Difusão de Concerns em Componentes. Todos os concerns do SMA requerem mais componentes na definição da solução OP do que na solução OA. Todos os papéis precisaram de mais de 5 classes para sua definição, enquanto um único aspecto foi capaz de encapsular cada papel do sistema: Respondedor (6 contra 1), Chamador (7 contra 1), ContentSupplier (6 contra 1), e Editor (7 contra 1). As propriedades de agência também precisaram de mais componentes na implementação e no projeto OP: adaptação (3 contra 1), autonomia (3 contra 2), colaboração (15 contra 6) e interação (7 contra 6). Finalmente, também foram usados mais componentes na implementação e projeto OP dos tipos de agente – 37 contra 33, para os tipos de agente do usuário e informação – e suas respectivas instâncias – 3 contra 1 para agentes de informação e 4 contra 2 para agentes do usuário.

Difusão de Concerns em Operações. Mais uma vez, todos os concerns requerem mais operações (métodos/advices) no sistema OP do que no sistema OA. Muitos concerns na solução OP foram implementados com mais do que o dobro do número de operações usadas no sistema OA. Por exemplo, o caso do concern de adaptação e o concern de autonomia. Há alguns casos em que a diferença é ainda maior (por exemplo, todos os papéis), menos do que o dobro (por exemplo, colaboração e os tipos de agente) ou quase igual (por exemplo, agência e interação).

Difusão de Concerns em LOC. As medidas aqui também demonstraram que a solução OA foi mais eficaz na modularização dos concerns de SMAs. Os resultados dos tipos de agente não mostraram qualquer diferença. Todos os outros casos, exceto o concern de agência, foram melhor encapsulados na implementação e projeto OA. As diferenças detectadas foram bastante significativas nos concerns a seguir: (i) as propriedades básicas do agente, (ii) os papéis do agente, (iii) as instâncias do agente e (iv) a propriedade de colaboração. O problema com o concern de agência ocorreu porque o aspecto Interaction especifica em sua definição que ele deve ser executado antes dos aspectos Collaboration, aumentando o número de pontos de transição.

8.7.2
A fase de evolução e reutilização do SMA

A seguir, discutiremos cada cenário (Seção 8.6) e seus respectivos resultados nas soluções orientadas a padrões e aspectos. Os dados reunidos aqui confirmaram muitos dos problemas identificados no estudo qualitativo (Seção 7.2.2). A Tabela 7 resume os principais resultados dos cenários de reutilização e evolução; os resultados mais significativos estão em cinza. As Figuras 85 e 86 ilustram algumas mudanças básicas nos artefatos do projeto, necessárias nos cenários C2 e C4.

Cenário C1 - Alteração nos papéis do agente

Um novo papel e seus respectivos planos foram incorporados ao sistema para encapsular um comportamento revisor e planos associados. Esse papel deve estar associado ao tipo UserAgent que é o tipo de agente que modela os usuários do sistema. Essa alteração resultou em um impacto similar nas soluções OA e OP. Contudo, foi necessário adicionar algumas outras linhas na alteração do sistema OP.

	MODIFIABILITY																			
	EXTENSIBILITY																			
	Changed Comp.		Changed Operat.		Added Comp.		Added Operat.		Changed Relations.		Added Relations.		Added LOCs		Changed LOCs		Copied Comp.		Copied LOCs	
	PO	**AO**	**PO**	**AO**	**PO**	**AO**	**PO**	**AO**	**PO**	**AO**	**PO**	**AO**	**PO**	**AO**	**PO**	**AO**	**PO**	**AO**	**PO**	**AO**
S1	1	1	3	3	5	5	2	3	0	0	15	15	101	98	1	1	-	-	-	-
S2	0	0	2	2	4	4	0	0	0	0	10	10	84	86	0	0	-	-	-	-
S3	0	0	2	2	4	4	0	0	0	0	10	10	84	86	0	0	0	0	0	0
S4	0	0	2	3	8	8	0	0	0	0	29	25	188	167	0	8	0	0	0	0
S5	1	1	2	1	0	0	1	1	0	0	4	2	16	14	0	0	0	0	6	6
S6	0	0	0	0	0	0	0	0	0	0	0	0	15	15	0	0	-	-	-	-
S7	5	1	0	0	0	0	0	0	5	2	1	1	0	0	5	1	0	0	40	0

Tabela 7. Os resultados dos cenários de manutenção e reutilização

Cenário C2 - Criação de um tipo de agente

Os usuários do Portalware normalmente precisam buscar informações armazenadas em vários bancos de dados e disponíveis na Web para produzir o material necessário para os editores. Como conseqüência, um novo tipo de agente, chamado InformationAgent, foi incluído no sistema para automatizar essa tarefa que consome tanto tempo. Cada instância InformationAgent contém diferentes planos de busca e está acoplada a uma fonte de informações. Essa alteração resultou em um impacto

similar nas soluções OA e OP. Contudo, esse cenário disparou outros dois cenários de mudança (C3 e C4), descritos a seguir.

Cenário C3 - Reutilização de concerns de agência

Como todos os tipos de agente do sistema incorporam os recursos de agência, a incorporação do tipo InformationAgent exigiu a reutilização desses recursos. Conforme apresentado na Tabela 7, esse cenário também resultou em mudanças similares nos sistemas OP e OA.

Cenário C4 – Inclusão de colaboração em um tipo de agente

A inclusão do agente de informação também exigiu a reutilização do concern de colaboração anteriormente definido no sistema. Isso acontece porque os agentes de informação colaboram entre si quando um agente de informação não consegue encontrar as informações necessárias na respectiva fonte de informação. Como resultado, os agentes de informação exercem os papéis de chamador e respondedor, respectivamente para: (1) chamar outros agentes de informação e pedir uma informação e (2) receber solicitações e enviar resultados de busca ao chamador. Portanto, esse cenário incluiu duas tarefas principais: (1) a reutilização do concern de colaboração predefinido no contexto de InformationAgent e (2) a criação e o acoplamento de dois papéis ao tipo InformationAgent. Esse foi o cenário que resultou em diferenças mais substanciais entre as mudanças na solução OP e OA: (1) o código OP precisou de 20 linhas a mais do que o código OA, (2) foram adicionados mais relacionamentos no projeto OP e (3) oito linhas foram removidas do código OA enquanto no código OP não houve qualquer mudança nas linhas.

Cenário C5 - Reutilização de Papéis

A introdução de agentes de informação ao sistema tornou os serviços de informação disponíveis para outros agentes. Como alguns agentes de usuário exercem o papel de fornecedor de conteúdo, o sistema pode automatizar a tarefa de seleção de informações relevantes para determinados contextos em nome dos respectivos usuários. Nesse sentido, os agentes do usuário devem usar os serviços dos agentes de informação que exercem um papel de chamador, já definido no SMA. Em outras palavras, o papel de chamador deve ser reutilizado e acoplado ao tipo UserAgent e o

papel ContentSupplier nas soluções OP e OA. Esse cenário de reutilização exigiu mais esforço no projeto OP do que no projeto OA da seguinte forma: (1) número de operações alteradas (2 contra 1), (2) número de relacionamentos adicionados (4 contra 2) e (3) número de LOCs adicionadas (16 contra 14).

Cenário C6 - Criação de uma nova instância do agente

Esse cenário investigou o impacto da adição de novas instâncias do agente ao sistema. Em especial, criamos uma nova instância do tipo UserAgent para exercer o papel do fornecedor de conteúdo. Esse cenário de manutenção precisou das mesmas mudanças nas versões OP e OA do SMA. A adição de 15 LOCs foi implementada nas duas versões.

Cenário C7 – Alteração da definição de agência

O último cenário foi criado para similar uma mudança realmente difusa nas duas soluções. Com a inclusão do agente de informação no sistema, todos os outros tipos de agente conseguiam usar seus serviços para alcançar os objetivos. A definição de agência deve ser estendida para incluir o concern de colaboração. Então todos os tipos de agente devem ser colaborativos, o que implica remoção do concern de colaboração da classe UserAgent e associação do componente de colaboração à classe Agent. Então, esse cenário requer a reutilização da definição de colaboração anterior em um novo contexto. A solução OA ofereceu um maior suporte à modificabilidade e extensibilidade nesse caso: (1) 5 componentes foram alterados no projeto OP e apenas um no projeto OA, (2) 5 relacionamentos foram alterados no projeto OP enquanto 2 foram modificados no projeto OA, (3) 40 linhas foram alteradas no código OP e nenhuma foi modificada no código OA.

8.8 Discussão

Apesar de as conclusões não poderem se extrapoladas para todos os SMAs, este estudo foi conduzido em um sistema que inclui as características canônicas de SMAs reativos. Alguns procedimentos foram considerados a fim de minimizar os problemas comuns enfrentados em experimentos da engenharia de software. Além disso, este trabalho oferece um outro framework (e seleciona um conjunto de

concerns de SMAs) que outros desenvolvedores de SMAs podem reutilizar e refinar em suas aplicações de SMAs a fim de melhorar o conhecimento da comunidade de software sobre a relação entre os agentes e objetos e as dificuldades associadas ao projetar e implementar SMAs com abordagens OO.

Não podemos garantir que as soluções orientadas a padrões usadas sejam as melhores para o problema existente. Entretanto, as abordagens investigadas baseiam suas diretrizes de projeto em soluções bem documentadas [79, 82, 137, 240]. A Seção 8.8.1 fornece uma descrição detalhada dos resultados e uma análise da hipótese apresentada. A Seção 8.8.2 discute como este estudo comparativo proporciona uma melhor compreensão da interação entre agentes e objetos. A Seção 8.8.3 discute como este estudo pode ser considerado uma primeira avaliação do framework de avaliação. A Seção 8.8.4 descreve as restrições da validade dessa avaliação.

8.8.1 Comparação das abordagens

Em geral, os resultados (Seção 8.7) confirmaram a hipótese apresentada (Seção 8.4). A abordagem orientada a aspectos oferece um melhor suporte à manutenção e a reutilização do projeto selecionado do que a abordagem orientada a padrões. Apesar de o projeto orientado a aspectos ter sido realizado antes do projeto orientado a padrões, muitas medidas indicaram que a solução anterior apresentou resultados melhores. A abordagem orientada a aspectos produziu um SMA mais conciso em termos de linhas de código, vocabulário externo e interno dos componentes. O uso de padrões de projeto leva a um aumento no número de classes, com o objetivo de superar as limitações dos mecanismos de composição das linguagens de programação OO. Essa conclusão tem o suporte de todas as métricas de tamanho, exceto a métrica de OCC.

A abordagem orientada a aspectos produziu operações mais complexas, ou seja, advices, do que a abordagem orientada a padrões. Isso acontece porque a modularização de alguns concerns nos aspectos requer que o contexto do objeto seja recapturado por meio de pointcuts. Esse resultado teve o suporte dos valores de OCC. A abordagem orientada a padrões também levou ao abuso do mecanismo de herança,

o que é fundamental para o estabelecimento de acoplamentos de muita herança. Esse problema foi detectado pelos valores de DIT.

A abordagem orientada a padrões produziu componentes mais altamente acoplados do que a abordagem orientada a aspectos. Essa é uma conseqüência da falta de força de expressividade dessa abordagem para modularizar concerns de SMAs. Esse resultado teve o suporte da métrica CBC. No entanto, a abordagem orientada a padrões produziu resultados melhores em termos de coesão do que a abordagem orientada a aspectos. A falta de coesão nos aspectos ocorre porque um aspecto deve encapsular o comportamento crosscutting aplicado a diferentes componentes. Contudo, esses comportamentos não podem estar diretamente relacionados entre si, produzindo valores de LCOO altos.

A abordagem orientada a aspectos oferece claramente mais suporte à separação de concerns de SMAs. Os mecanismos orientados a aspectos oferece mais suporte à modularização de concerns de agência. Essa descoberta tem o suporte de todas as métricas de separação de concerns. Finalmente, em termos dos cenários de reutilização e evolução (Seção 8.6), a abordagem orientada a aspectos apresentou resultados melhores. Essa conclusão teve o suporte destas métricas: componentes alterados, relacionamentos alterados, relacionamentos adicionados, linhas de código adicionadas e linhas de código copiadas.

8.8.2 Agentes *versus* objetos

Em geral, descobrimos que as abordagens avançadas OO, que têm o suporte de métodos eficazes, podem lidar com sucesso com vários concerns de SMAs. As demais abstrações (padrões e aspectos) foram importantes para cobrir lacunas conceituais entre agentes e objetos. As abstrações puras OO foram úteis para encapsular alguns concerns de agência básicos. As classes e o mecanismo de herança oferecem suporte direto para a estruturação de vários tipos de agente. A herança foi útil para promover a reutilização das operações comuns a todos os tipos de agente, como as atualizações de planos e crenças.

Os padrões e os aspectos minimizaram os desalinhamentos entre os artefatos de software. As abordagens investigadas foram muito úteis para promover a rastreabilidade entre modelos orientados a agentes e código e projetos OO. Entretanto, a noção de aspectos proporcionou uma melhor rastreabilidade porque a propriedade foi adicionada ou removida do modelo orientado a agentes, isso foi alcançado diretamente nas threads de código e projeto orientado a aspectos. A abstração do aspecto também é mais apropriada para lidar com papéis a partir do ponto de vista da manutenção e reutilização (Seção 8.7.2). Essas descobertas são semelhantes àquelas relatadas em [136].

Os padrões de projeto não têm representações de primeira classe no nível da implementação. A implementação de um padrão de projeto, portanto, não pode ser reutilizada e, apesar de o projeto ser reutilizado, o desenvolvedor do SMA é forçado a implementar o padrão várias vezes. Diferente dos padrões, os aspectos apresentam representação de primeira classe no nível da implementação para concerns de agência (como interação, colaboração, papéis etc), oferecendo suporte à reutilização dos níveis de implementação e projeto. Por exemplo, a reutilização da propriedade de Colaboração no contexto dos agentes do usuário precisou da associação do aspecto Collaboration à classe UserAgent, descrevendo os join points de interesse, enquanto na abordagem baseada em padrões, algumas outras modificações (Tabela 7) foram feitas para introduzir a associação e as chamadas explícitas aos métodos definidos na interface da classe Collaboration.

Apesar de os resultados terem demonstrado que a abordagem orientada a aspectos fornece mais reusabilidade e manutenibilidade, a abordagem orientada a padrões também ofereceu suporte a alguns concerns de SMAs. Então os aspectos e os padrões OO podem coexistir naturalmente em uma solução híbrida. Em [199], Rashid apresenta um exemplo de uma abordagem híbrida à separação de concerns.

8.8.3
Avaliação do framework de avaliação

O objetivo da fase de Reutilização e Evolução (Seção 8.7.2) também foi importante para confirmar que as métricas propostas (Seção 6.4) são mecanismos

úteis para predizer a reusabilidade e a manutenibilidade. Simulamos alterações simples e complexas que envolviam concerns de agência para as soluções OO e OA a fim de medir a facilidade encontrada ao desenvolver e reutilizar os componentes. Selecionamos sete cenários de manutenção e reutilização que são recorrentes em SMAs de larga escala (Seção 8.6). A Tabela 7 apresentou a lista completa das medidas.

Os resultados apresentados na Tabela 7 confirmaram as previsões dos dados usando as métricas propostas, o que proporcionou comprovações substanciais da utilidade das métricas. Por exemplo, a inclusão dos novos papéis (C1) precisou de algumas linhas adicionais para alterar o sistema OO. Isso ocorreu porque a abordagem orientada a aspectos oferece suporte a uma melhor separação de concerns, conforme indicado nas medidas de SoC. A introdução dos recursos de colaboração a um tipo de agente específico (C4) foi o cenário que resultou em diferenças mais substanciais entre as mudanças na solução OP e OA: (1) o código OO precisou de 20 linhas a mais do que o código OA, (2) foram adicionados mais relacionamentos no projeto OO e (3) oito linhas foram removidas do código OA enquanto no código OO não houve qualquer mudança nas linhas. Esses resultados confirmaram que o acoplamento (métricas DIT e CBC), número de componentes e atributos (métricas VS e NOA) e a separação de concerns (métrica SoC) causam um impacto direto nas atividades de reutilização e manutenção. Essa descoberta se confirmou nos cenários C5 e C7. Contudo, não foi possível entender a interação entre a coesão (métrica LCOO) e a reusabilidade e manutenibilidade dos sistemas produzidos.

8.8.4
Ameaças à validade

Este estudo tem algumas restrições de validade em termos da validade do construto, da validade interna e externa. De acordo com a validade do construto, manutenibilidade, reusabilidade, compreensibilidade e flexibilidade são conceitos difíceis de medir. As variáveis dependentes usadas baseavam-se em um estudo realizado anteriormente por Li et al. [158]. Em seu trabalho, o conceito “esforço de manutenção” foi concretizado como o número de linhas de código alterado. Em

futuros trabalhos, o autor planeja usar outras medidas representativas, como “tempo para compreender, desenvolver e implementar modificações” [14]. Apesar de ser desejável conseguir a validade do construto para as variáveis independentes (as métricas descritas na Seção 6.4), isso está além do escopo deste trabalho. Entretanto, algumas métricas são extensões das métricas de CK que foram teoricamente validadas.

Em relação à validade interna, este estudo empírico não pode ser considerado um experimento controlado, uma vez que todos os sujeitos fizeram parte do desenvolvimento dos dois sistemas. Contudo, tentamos minimizar as influências, selecionando os dois sujeitos que defenderam (antes do estudo) a abordagem orientada a padrões e dois outros que defenderam a abordagem orientada a aspectos. Apesar do efeito causado pelo aprendizado dos sujeitos, à medida que o estudo prosseguia, o sistema OA, desenvolvido primeiro, apresentou melhores resultados.

Finalmente, em relação à validade externa, o tamanho limitado e a complexidade do sistema e o uso de sujeitos estudantes podem restringir a extrapolação dos resultados do estudo. Todavia, embora os resultados não possam ser diretamente generalizados para os desenvolvedores profissionais e sistemas do mundo real, a definição acadêmica permite aos engenheiros de software fazerem avaliações iniciais úteis sobre quais abordagens investigadas devem ser mais estudadas. Apesar de suas limitações, o estudo constitui um trabalho empírico inicial sobre as abordagens investigadas e é um complemento ao estudo qualitativo realizado anteriormente [82] (Seção 7.2).

8.9
Comparação com trabalhos relacionados

No entendimento do autor, este é o primeiro estudo quantitativo do DSOA. Até agora, muitos estudos empíricos que envolviam aspectos baseavam-se em critérios subjetivos e em uma investigação qualitativa [82, 115, 138]. Por exemplo, Hannemann e Kiczales comparam as implementações de Java e as implementações de AspectJ dos padrões de projeto GoF [79] em termos de critérios de medição fragilmente definidos, como composabilidade e *plugability*. Apenas alguns trabalhos

usam métricas de software para o desenvolvimento de software orientado a aspectos, como o trabalho de Lopes [162] (Seção 6.6). Além disso, a definição de seu estudo é bastante acoplada à distribuição de concerns no código Java.

8.10 Resumo

A separação de concerns de SMAs é essencial para os engenheiros de SMAs porque eles podem decidir estender e modificar esses concerns à medida que o sistema evolui. O desenvolvimento de SMAs complexos enfrentou uma transição de modelos orientados a agentes para uma implementação e projeto orientado a objetos. Essa transição precisa garantir que os concerns de SMAs, que são encapsulados por abstrações em modelos orientados a agente de alto nível, também sejam mapeados com sucesso em abstrações disponíveis em linguagens de programação OO e frameworks de implementação. Entre os problemas inerentes a essa transição, nenhum é mais sério do que a dificuldade de lidar com diferenças conceituais entre agentes e objetos, o que requer a aplicação de princípios bem estabelecidos e técnicas e métodos de suporte. De forma mais geral, há a necessidade de compreender os relacionamentos entre objetos e agentes.

Em um trabalho anterior [223], identificamos várias similaridades e diferenças entre abstrações orientadas a agentes e objetos dentro de um framework conceitual unificador. Entretanto, não deixamos explícito como, para os engenheiros de SMAs, os concerns são modularizados com sucesso com abstrações OO nas fases de implementação e projeto. Este capítulo apresentou um estudo empírico para entender esse fenômeno e comparar o suporte à manutenção e à reutilização da abordagem orientada a aspectos e uma abordagem orientada a padrões para o desenvolvimento de SMAs. Os resultados demonstraram que o uso da abordagem orientada a aspectos permitiu a construção de um SMA com uma melhor estruturação para a reutilização e manutenção dos concerns de SMAs. O uso dos aspectos resultou em uma melhor separação de concerns, menos acoplamento entre seus componentes (apesar de menos coeso) e menos linhas de código. Outra importante conclusão deste estudo empírico foi que a abordagem orientada a aspectos também ofereceu um melhor alinhamento

com as abstrações de nível mais alto de modelos orientados a agentes. Como este é o primeiro estudo explanatório, para confirmar as descobertas, são necessários outros experimentos controlados e rigorosos.

É importante observar que, a partir desta experiência, em especial em uma área não rigorosa como a engenharia de software, não podemos chegar a conclusões gerais. O escopo de nossa experiência limitou-se a (a) concerns de agência definidos em TAO (Seção 3.1), (b) técnicas usadas para a implementação de concerns de agência (por exemplo a colaboração baseada em sincronização) e (c) um dado subconjunto de cenários de aplicação que foram retirados da experiência de desenvolvimento do autor. Contudo, o objetivo aqui é fornecer algumas comprovações para uma discussão mais geral sobre os benefícios e perigos criados pela abordagem orientada a aspectos proposta, assim como quais e quando as características da abordagem orientada a aspectos podem ser úteis. Este capítulo também apresentou um alerta aos engenheiros de software que estão entrando na área de DSOA trazendo expectativas muito altas.