9
Conclusão e trabalhos futuros

O aumento da complexidade das aplicações de software de hoje em dia e o advento de tecnologias inovadoras e recentes fazem com que os sistemas de software incorporem e lidem com cada vez mais concerns do agente, como mobilidade, adaptação e aprendizagem. Por trás desses concerns com propósitos especiais está o concern básico responsável pela funcionalidade básica do sistema. Os concerns do agente são introduzidos para atender requisitos especiais da aplicação ou para gerenciar e otimizar a funcionalidade básica do sistema (por exemplo, personalização, otimização, data mining).

Foram propostas algumas abordagens arquiteturais para separar e integrar concerns do agente com base nos estilos arquiteturais tradicionais (Seção 3.7). Contudo, essas abordagens não oferecem suporte à composição flexível das propriedades de agente e à construção de tipos de agente heterogêneos. Apesar de os concerns poderem ser de alguma forma separados conceitualmente, trazê-los para o código pode acarretar em muitos problemas:

- A programação de código entrelaçado é difícil e complexa, porque todos os concerns têm de ser tratados ao mesmo tempo e no mesmo nível. Com o aumento da complexidade do agente, as linguagens orientadas a objetos não oferecem abstrações adequadas para a separação de concerns do agente no nível da implementação.
- O código entrelaçado é difícil de entender devido à falta de abstração.
- O código entrelaçado é difícil de manter e reutilizar porque os concerns estão altamente ligados.
- O código entrelaçado dá origem a anomalias na herança devido à forte conexão dos diferentes concerns do agente. Torna-se impossível alterar a

implementação de um método ou de um concern especial entrelaçado em uma subclasse sem alterá-los.

9.1 Contribuições

Acreditamos que a separação de concerns do agente não deve ser considerada apenas durante a implementação, mas sim antecipadamente durante as etapas de projeto detalhado e arquitetural a fim de permitir a construção de SMAs com reusabilidade e manutenibilidade. Este trabalho de pesquisa explora os avanços no desenvolvimento de softwares orientados a aspectos para permitir a separação de concerns do agente em uma etapa inicial do projeto. Isso oferece suporte ao desenvolvimento de SMAs orientados a aspectos em diferentes etapas do desenvolvimento: definição da arquitetura, projeto detalhado, implementação e avaliação. Este trabalho oferece as contribuições a seguir:

- *Um framework conceitual unificador*: O framework TAO oferece definições para os concerns de SMAs e os agrupa em diferentes categorias (Capítulo 3). Oferece suporte à caracterização e comparação das arquiteturas orientadas a aspectos (Seção 3.7) e o desenvolvimento do método arquitetural (Capítulo 4) e a linguagem de padrões (Capítulo 5).
- *Uma arquitetura de agente orientada a aspectos:* A arquitetura de agente orientada a aspectos (Seção 4.3) é uma descrição de alto nível de um projeto do agente, que oferece suporte à modularização arquitetural de concerns do agente. A arquitetura aumenta a flexibilidade da composição dos concerns do agente e da construção de tipos de agente heterogêneos. A arquitetura proposta oferece, logo no início da etapa de projeto arquitetural, o contexto no qual podem ser tomadas decisões mais detalhadas sobre a separação de concerns do agente nas etapas posteriores do desenvolvimento do SMA.
- *Um método arquitetural:* O método orientado a aspectos proposto (Seção 4.3) permite que os engenheiros de software analisem o comportamento dos SMAs a partir de diferentes perspectivas e considera os concerns de agência de forma

elegante. Além disso, ele incorpora um conjunto de princípios do projeto que garante e observância de uma melhor separação de concerns. Se aplicados com cuidados, esses princípios podem levar a SMAs que tenham qualidades desejáveis prescritas por requisitos de SMAs gerais, incluindo a reusabilidade e manutenibilidade.

- *Uma linguagem de padrões:* A linguagem de padrões de projetos baseada em aspectos proposta (Capítulo 5) oferece soluções de projeto para lidar com cada um dos aspectos de agência enquanto segue a estrutura geral da arquitetura orientada a aspectos (Seção 4.3). Esses padrões oferecem suporte a uma transição tranqüila do nível arquitetural ao nível de projeto detalhado, porque baseiam-se em abstrações orientadas a aspectos, conforme descrito pelo método orientado a aspectos (Seção 4.3).
- *Implementação de padrões em duas linguagens orientadas a aspectos:* O Apêndice I apresenta uma implementação de AspectJ dos padrões. A disposição no Apêndice II de algumas regras de transformação para mapear uma implementação de AspectJ para uma implementação de Hyper/J mostra que a implementação dos padrões em Hyper/J é possível.
- *Um framework de avaliação para DSOA:* O framework de avaliação proposto (Capítulo 6) oferece suporte à avaliação quantitativa da implementação e do projeto orientados a aspectos. Isso permite que os engenheiros de software meçam importantes atributos de qualidade em artefatos de software orientados a aspectos, e ajuda no controle de qualidade em diferentes contextos de desenvolvimento. O framework de avaliação foi usado em três estudos empíricos heterogêneos (Seção 6.5).
- *Um modelo de qualidade:* O modelo de qualidade (Seção 6.3) define como medir os atributos de qualidade externos, a reusabilidade e manutenibilidade, em relação aos atributos de software internos bem conhecidos.
- *Um conjunto de métricas:* O conjunto de métricas proposto (Seção 6.4) preenche uma lacuna na engenharia de software empírica para o DSOA ao oferecer

mecanismos eficazes para medir a coesão, o acoplamento, o tamanho e a separação de concerns.

- *Estudos qualitativos:* O Capítulo 7 apresenta estudos qualitativos que envolvem crosscutting concerns específicos a SMAs. Eles complementam estudos de caso encontrados na literatura com enfoque em crosscutting concerns conhecidos, como auditoria e tratamento de exceções. Esses estudos permitiram a realização de avaliações iniciais da abordagem orientada a aspectos proposta e ofereceu uma melhor compreensão dos relacionamentos entre concerns do agente ao longo de todo o ciclo de vida do software.
- *Estudo quantitativo:* O Capítulo 8 descreveu um experimento semicontrolado com o objetivo de comparar a abordagem orientada a aspectos e uma abordagem orientada a padrões com base no framework de avaliação quantitativo proposto. No entendimento do autor, este é o primeiro estudo quantitativo do desenvolvimento de software orientado a aspectos.

9.2 Trabalhos em andamento e trabalhos futuros

Essas contribuições representam um primeiro esforço no suporte à separação dos concerns do agente desde a etapa arquitetural até as etapas de implementação e avaliação. Apesar dos benefícios identificados no uso da abordagem proposta, há muitos trabalhos sendo desenvolvidos no momento e trabalhos futuros, alguns dos quais descritos a seguir.

Outros estudos empíricos. A abordagem orientada a aspectos proposta foi avaliada usando sistemas representativos de diferentes domínios (Capítulo 7). O estudo quantitativo do Capítulo 8 oferece fortes comprovações dos benefícios da abordagem orientada a aspectos. O primeiro estudo importante precisou de 2 anos para ser desenvolvido. Apesar de os estudos quantitativos consumirem muito tempo, é necessário duplicar este estudo com uma grande variedade de aplicações de agente e diferentes tipos de facetas de heterogeneidade. Esses estudos duplicados construiriam um corpo de conhecimento aprimorado sobre a relação entre objetos, aspectos e concerns de SMAs. Por exemplo, seria desejável conduzir estudos quantitativos com

a aplicação de técnicas de IA em larga escala a fim de entender como a abordagem orientada a aspectos faz o escalonamento nesse contexto. Um experimento similar que envolvia o sistema Expert Committee (Seção 5.1) está sendo realizado no laboratório LES na PUC-Rio. Além disso, também está sendo realizado um estudo quantitativo na Universidade Federal da Bahia (UFBA) com estudantes de graduação como parte do projeto *RefazendA*[11] [255]. Todavia, é necessário duplicar os estudos fora de nosso ambiente de pesquisa. Esta tese incentiva essa replicação experimental porque oferece um framework de avaliação e a organização do primeiro estudo empírico.

Refatoração orientada a aspectos de SMAs existentes. As métricas propostas podem funcionar como mecanismos para detectar oportunidades de refatoração com aspectos. Elas podem identificar quando algum concern está entrelaçado ou espalhado na implementação do sistema. O conjunto de métricas está sendo aplicado atualmente a um SMA desenvolvido de forma orientada a objetos. O SMA é um programador tolerante a falhas desenvolvido em LIP6 [257]. As métricas estão sendo usadas para detectar problemas [256] no código e no projeto orientado a objetos a fim de usar os padrões orientados a aspectos propostos para melhorar a manutenção e reutilização do sistema. Com base nessa experiência, é possível propor algumas regras de refatoração orientada a aspectos. Este trabalho está sendo desenvolvido no contexto do projeto RefazendA [255] e em cooperação com o Departamento de Ciência da Computação da UFBA.

Abordagem Gerativa. Os SMAs incorporam várias características que tendem a ser espalhadas não apenas nos artefatos de implementação, projeto e arquitetura, mas também nos artefatos produzidos nas primeiras etapas de desenvolvimento, como a especificação de requisitos, a análise e a modelagem conceitual. Como conseqüência, a separação explícita de concerns do agente deve ter suporte dessas fases preliminares. Estamos desenvolvendo uma abordagem gerativa [55] que permite a representação separada dos concerns de SMAs ao longo de todo o ciclo de vida do software e oferece suporte à geração de código. A abordagem gerativa explora o

[11] *RefazendA* significa “Refatoração com aspectos”.

domínio do SMA para permitir a geração de código das arquiteturas de agente orientadas a aspectos (Capítulo 4). O uso dessa abordagem resolveria o problema das tarefas que demandam muito tempo na definição dos aspectos de agência, como uma descrição extensiva de pointcuts (Seção 5.11.2). A abordagem é composta de uma linguagem específica a domínio (DSL) que permite a modelagem de crosscutting concerns e concerns ortogonais, como o conhecimento do agente, a autonomia, interação, adaptação, aprendizagem e papéis. O método arquitetural proposto e a linguagem de padrões proposta têm o suporte de diversas ferramentas e assistentes que automatizam a geração de código em AspectJ. A implementação da abordagem gerativa usa: (i) tecnologias XML para especificar a DSL, (ii) linguagens de programação Java e AspectJ para implementar uma versão concreta da arquitetura orientada a aspectos proposta e (iii) um gerador de código implementado como um *plug-in* Eclipse [213], que mapeia abstrações na DSL para classes e aspectos específicos da arquitetura de agente proposta. Um protótipo da abordagem gerativa proposta foi implementado. Os primeiros resultados foram publicados como um trabalho no workshop sobre “Primeiros Aspectos” [147]. Este trabalho é uma pesquisa de doutorado que é um desdobramento natural desta tese.

Aspectos em modelagem orientada a agentes. De acordo com a experiência desta pesquisa, há alguns crosscutting concerns mesmo no nível de modelagem orientada a agentes, como coordenação, tratamento de exceções e ciência de contexto. Esta tese preocupou-se com uma abordagem de projeto e arquitetura para o suporte de uma integração completa das propriedades de agente em objetos com base em aspectos. O autor está planejando estender o metamodelo TAO [223] e a linguagem de modelagem orientada a agentes (por exemplo [222]) com abstrações orientadas a aspectos para oferecer suporte à representação de crosscutting concerns em modelos orientados a agentes. Essa extensão parece ser uma próxima etapa imediata por causa dos resultados interessantes obtidos no nível de implementação e projeto [98] (Capítulo 8).

Uma abordagem baseada em buscas para a medição e a análise de aspectos. Uma programação orientada a aspectos introduz algumas novas abstrações de implementação, como aspectos, join points, pointcuts, advices e inter-type

declarations (Seção 2.2). Os engenheiros de software precisam ter o suporte de ferramentas ao longo do processo de manutenção e desenvolvimento a fim de compreender os sistemas orientados a aspectos (Seção 5.11.2). Em geral, não é trivial entender o comportamento final de classes por causa da propriedade de transparência dos aspectos (Seção 2.2.2); diversos aspectos podem alterar de forma transparente sua estrutura e comportamento. Assim, há a necessidade de outras ferramentas para oferecer suporte à compreensão de programas orientados a aspectos por meio do desenvolvimento e manutenção. Os Analisadores de Programas Estáticos (Static Program Analyzers - SPA) [126] são ferramentas interativas que melhoram a compreensão do programa durante o desenvolvimento e a manutenção respondendo às perguntas sobre os programas. Dependendo da tarefa de programação disponível, os SPAs devem processar diferentes programas-fonte e responder diferentes tipos de buscas de programa [126]. Desenvolvemos um framework baseado em buscas para oferecer suporte à análise e à medição de programas orientados a aspectos. A parte integral desse framework é um conjunto de notações para a modelagem conceitual do projeto do programa orientado a aspectos e AQL (Aspect Query Language), uma notação para escrever buscas em programas. Este trabalho está sendo desenvolvido em cooperação com o Grupo de Sistemas de Computação da Universidade de Waterloo. Os resultados preliminares podem ser encontrados em um relatório técnico [258].

Avaliação Arquitetural. O framework de avaliação proposto oferece suporte à avaliação da reusabilidade e manutenibilidade da implementação e do projeto orientado a aspectos. O autor planeja estender os recursos do framework de avaliação a fim de oferecer suporte à avaliação de arquiteturas orientadas a aspectos. A idéia é estender os métodos existentes para a avaliação de arquitetura, como SAAM [18, 134].