1
Introdução

A orientação a objetos é atualmente o paradigma dominante de desenvolvimento de software. É consenso que ela trouxe muitos benefícios em relação às abordagens anteriores. Porém, a orientação a objetos também possui algumas limitações. Uma dessas limitações é a incapacidade de modularizar bem todos os *concerns* de um sistema de software [1, 2, 3]. Um *concern*[1] é alguma parte do problema que se deseja tratar como uma unidade conceitual [3]. Existem alguns *concerns* que, inerentemente, atravessam e permeiam vários objetos e ficam espalhados pela estrutura de classes do sistema. São os chamados *crosscutting*[2] *concerns* [3]. Tratamento de exceções, distribuição e sincronização são exemplos de *crosscutting concerns* [1].

Desenvolvimento de software orientado a aspectos (DSOA) [1, 2, 3] é um paradigma novo que promete separação avançada de *concerns* e, conseqüentemente, sistemas de software mais fáceis de manter e reutilizar. O desenvolvimento de software orientado a aspectos introduz uma nova abstração, chamada de aspecto, para modularizar os *crosscutting concerns*, e provê novos mecanismos para fazer a composição entre aspectos e classes. No entanto, como o paradigma de orientação a aspectos ainda está numa fase inicial de sua existência, não se sabe se ele realmente melhora a manutenibilidade e reusabilidade dos sistemas de software. Além disso, ainda é difícil determinar quando se deve usar aspectos como solução arquitetural e quais são as boas decisões de projeto e implementação orientados a aspectos. Só existe até agora um pequeno consenso de que *crosscutting concerns* clássicos e óbvios, como tratamento de exceções, devem ser modularizados por aspectos. Não há orientações que dêem apoio e guiem o projeto orientado a aspectos de outros importantes *crosscutting concerns*

[1] O termo *concern* não possui uma tradução direta para o português no contexto de engenharia de software, por isso foi usado em inglês nesta dissertação.

[2] O termo *crosscutting* não possui uma tradução direta para o português no contexto de engenharia de software, por isso foi usado em inglês nesta dissertação.

dependentes de domínio. Conseqüentemente, os aspectos têm sido usados muitas vezes de forma não sistemática.

A utilidade de novos paradigmas de desenvolvimento de software e suas práticas de projeto e implementação pode ser avaliada por meio de estudos experimentais. Normalmente, métricas de software são usadas em estudos empíricos para indicar os pontos fortes e fracos da abordagem estudada. A realização de estudos experimentais pode ser, portanto, uma forma de avaliar todas essas questões sobre o desenvolvimento de software orientado a aspectos. Alguns trabalhos já foram realizados nesse sentido [4, 5], mas a maioria dos estudos empíricos se baseou em análises qualitativas. Isso se deve, principalmente, ao fato de haver poucas métricas que possam ser aplicadas diretamente ao projeto e ao código de software orientado a aspectos. Além disso, muitos dos estudos empíricos realizados até agora não basearam suas avaliações em princípios e atributos de qualidade bem conhecidos e estabelecidos da engenharia de software, tais como acoplamento, coesão.

Existem muitas propostas de métricas para orientação a objetos, tais como as métricas de Chidamber & Kemerer [6] e as métricas de Henderson-Selers [7]. Muitas delas já foram bastante usadas e validadas empiricamente [8, 9]. Muitas empresas também desenvolveram seus próprios modelos de qualidade baseados em métricas de software [10, 11]. Além disso, alguns ambientes de desenvolvimento, como o Together 6.0 [12], já incorporaram em suas características o apoio à aplicação de métricas de software. No entanto, nenhuma dessas métricas é apropriada para a avaliação de software orientado a aspectos.

Neste contexto, este trabalho propõe um framework cujo objetivo principal é avaliar manutenibilidade e reusabilidade de software orientado a aspectos. O framework é composto por um conjunto de métricas e um modelo de qualidade. No sentido de se aproveitar os benefícios de soluções já bem testadas, os elementos do framework são baseados em princípios bem conhecidos da engenharia de software e métricas já existentes. As métricas são especializadas para serem aplicadas no projeto e no código de software orientado a aspectos. Elas medem quatro atributos de qualidade: separação de *concerns*, acoplamento, coesão e tamanho. O modelo de qualidade relaciona manutenibilidade e reusabilidade às métricas e aos atributos que elas medem. Engenheiros de software podem usar o framework para comparar soluções orientadas a aspectos

com soluções orientadas a objetos ou para avaliar decisões de projeto orientadas a aspectos.

O framework de avaliação proposto foi usado no contexto de dois estudos experimentais de domínios distintos, com características, níveis de controle e níveis de complexidade diferentes. Esses estudos serviram como avaliações iniciais da utilidade e usabilidade do conjunto de métricas e do modelo de qualidade. O primeiro estudo experimental comparou uma abordagem orientada a objetos com uma abordagem orientada a aspectos para o projeto e implementação de um sistema multi-agentes. O segundo estudo envolveu a aplicação do framework proposto para avaliar as implementações orientadas a aspectos e orientadas a objetos de alguns padrões de projeto de software.

O restante deste documento está estruturado da seguinte maneira. O capítulo 2 apresenta os conceitos do desenvolvimento de software orientado a aspectos. O capítulo 3 dá uma visão geral sobre medição de software. O capítulo 4 apresenta o framework de avaliação, dando ênfase à descrição do modelo de qualidade. As métricas são definidas no capítulo 5. Os dois estudos experimentais realizados com o uso do framework de avaliação são descritos no capítulo 6. O capítulo 7 fala brevemente de alguns trabalhos relacionados. Finalmente, o capítulo 8 apresenta algumas conclusões e possíveis trabalhos futuros. Os anexos A e B mostram todos os dados obtidos nos dois estudos experimentais realizados.