1
Introdução

O desenvolvimento de software orientado a aspectos (DSOA) [42] [43] [45] [66] é um paradigma recente que propõe mecanismos e abstrações para modularizar interesses transversais. Tais interesses são propriedades que naturalmente se espalham através dos módulos de um sistema e não são bem capturados por outros paradigmas, como o orientado a objetos (OO). O DSOA introduz uma nova unidade modular, chamado aspecto, para capturar o código espalhado de um interesse. Além disso, tal paradigma provê mecanismos para dar suporte à composição entre aspectos e outros módulos do sistema, como classes e interfaces. Alguns exemplos tradicionais de interesses transversais são: tratamento de exceções, persistência, distribuição, segurança, auditoria, entre outros. Os benefícios esperados da utilização de técnicas do DSOA são: superior separação de interesses, menor replicação de código, maior coesão dos módulos, menor acoplamento e, como conseqüência, aumento potencial de reutilização e evolução no desenvolvimento de sistemas de software complexos.

1.1.
Definição do Problema

Apesar dos benefícios prometidos com a utilização de DSOA, os próprios aspectos podem levar a complexidade adicional, ou ainda reduzir a qualidade das classes afetadas por eles [27] [28] [30] [32]. Na verdade, alcançar elevada qualidade em artefatos de software orientado a aspectos (OA) é um desafio por cinco razões principais. Primeiro, os desenvolvedores de software podem usar inadequadamente as construções das linguagens de programação OA, uma vez que estas exibem uma série de abstrações e mecanismos de decomposição e composição adicionais. Segundo, a separação de interesses alcançada por técnicas orientadas a aspectos pode ter impacto negativo em outros importantes atributos, como acoplamento e coesão [23] [27] [30]. Terceiro, a identificação de aspectos específicos do domínio e o projeto adequado de um sistema na presença destes

aspectos não são tarefas triviais, e requerem raciocínio adicional sobre decisões de projeto e implementação. Quarto, mesmo quando todos os aspectos são identificados, a separação completa de um determinado interesse transversal não é direta, como também não é trivial capturar todo o código que implementa tal comportamento. Finalmente, o projeto interno do próprio aspecto também pode conter problemas relacionados à separação de interesses [51].

Melhorar a qualidade no DSOA pode ser conseguido por meio de avaliações durante o ciclo de desenvolvimento. Estas avaliações são usualmente baseadas em métricas. Porém, a interpretação dos números gerados no processo de medição não é uma tarefa trivial e análises equivocadas são freqüentemente feitas. Além disso, fazer análise de qualidade em projeto e implementação sem qualquer automatização é um processo caro. Especialmente, no caso de DSOA, em que os aspectos exibem características particulares tais como (i) afetar vários módulos, incluindo outros aspectos, o que torna a análise de tais interações mais complicada e (ii) a propriedade desejável de inconsciência das classes em relação aos aspectos, o que torna ainda mais complicado entender como certas classes estão sendo afetadas por aspectos. Desta forma, um requisito fundamental para o processo de avaliação neste paradigma é o suporte adequado de ferramentas de software.

1.2. Limitações dos Trabalhos Relacionados

A Separação de Interesses (SI) sempre foi tida como de importância primária no processo de desenvolvimento, pois é um instrumento básico para se reduzir a complexidade de software [18]. Entretanto, ainda não existem abordagens que apresentam métodos de avaliação efetivos e suportados por ferramentas para as fases de projeto e implementação em DSOA. Também não se tem conhecimento de trabalhos que efetivamente apresentem boas práticas de desenvolvimento voltadas especificamente para o paradigma de aspectos. As únicas diretivas documentadas são explicitamente voltadas ao uso de construções de uma linguagem de programação em particular [34] [36], tais como AspectJ [66]. Além disso, a literatura inclui geralmente estudos de caso [30] [36] com foco apenas em SI para avaliar a qualidade de implementações orientadas a aspectos e, um dos motivos desta limitação, é ser difícil avaliar múltiplas características de

um software simultaneamente. Entretanto, a qualidade de software não depende exclusivamente de SI e vários outros importantes atributos da Engenharia de Software devem ser considerados, em especial coesão, acoplamento e tamanho.

Muitas ferramentas de software vêm sendo desenvolvidas para DSOA, mas a literatura ainda é escassa de abordagens de avaliação da qualidade. Um *framework* de avaliação bastante difundido na comunidade de DSOA e utilizado em diversos estudos de caso [30] [33] [60] é proposto por Sant'Anna *et al*. [57]. Este *framework* propõe um conjunto de métricas OA e um modelo para mapear os atributos mensuráveis do software em características de qualidade, como reusabilidade e manutenibilidade. Os atributos mensuráveis incluem coesão, acoplamento, tamanho e SI. Entretanto, a abordagem de Sant'Anna *et al*. não deixa claro como ocorre este mapeamento. Além disso, ela não apresenta um conjunto de atividades que possa ser automatizado para guiar o engenheiro de software no processo de avaliação. Em relação a ferramentas de suporte ao DSOA, estão surgindo propostas interessantes (como CME [15] e FEAT [56]) e apenas uma delas cobre o processo de medição [13]. Entretanto, tal ferramenta apóia somente a coletânea de dados associados com as métricas definidas. Nenhuma das ferramentas encontradas na literatura está associada a um método de avaliação que auxilie engenheiros de software na interpretação dos resultados de medições.

1.3. Solução Proposta

Esta dissertação de mestrado propõe um método para avaliação quantitativa de sistemas orientados a aspectos e uma ferramenta para automatizar as atividades definidas no método. O método proposto é fundamentado principalmente no princípio de separação de interesses e pode ser usado na avaliação de artefatos nas fases de projeto detalhado e implementação. Além disso, ele é organizado em etapas, baseado em métricas de software e apoiado por um conjunto de regras heurísticas. As métricas avaliam atributos bem conhecidos da Engenharia de Software como coesão, acoplamento, tamanho e SI. As regras heurísticas auxiliam na interpretação dos números e provêem suporte à identificação de problemas nos artefatos OA de projeto e implementação. Outro objetivo central das regras é

fornecer informações mais abstratas ao desenvolvedor, evitando que este trabalhe diretamente com o resultado das métricas.

Como o método propõe uma avaliação em iterações, uma ferramenta torna-se importante para aplicação efetiva do método em sistemas de médio a grande porte e aumento na confiabilidade dos resultados. Desta forma, este trabalho apresenta também uma ferramenta, chamada AJATO, que pode ser usada na avaliação de sistemas implementados em Java [17] e AspectJ [45]. O principal objetivo desta ferramenta é automatizar as atividades de medição e aplicação das regras definidas no método de avaliação. Para isso, ela disponibiliza um conjunto de métricas e regras heurísticas e oferece ainda mecanismos de extensão que tornam possível adicionar novos elementos. Tanto o método quanto a ferramenta foram definidos com base na experiência obtida em DSOA durante o mestrado no Laboratório de Engenharia de Software (LES) desta instituição de ensino.

Neste período, também foram desenvolvidos cinco estudos experimentais. Os experimentos contaram com a colaboração de equipes de outros grupos de pesquisa da Universidade de Lancaster, Universidade Federal do Rio Grande do Norte (UFRN), Universidade Federal de Pernambuco (UFPE) e Universidade Estadual de Campinas (UNICAMP). Estes estudos foram utilizados na avaliação e evolução dos três principais elementos da abordagem: método de avaliação, regras heurísticas e ferramenta. Em nossos estudos experimentais, a abordagem baseada em regras tem se mostrado útil para apontar problemas não triviais resultantes de uma análise equivocada das medições. Tanto problemas de SI, como de acoplamento e coesão podem ser identificados pelo método de avaliação. Entretanto, a abordagem é orientada a interesses e assume que problemas em outros atributos são relevantes quando estes estiverem relacionados a uma ineficiente SI.

1.4. Organização do Texto

O restante deste documento está estruturado da seguinte forma. No Capítulo 2, são detalhados os principais conceitos e abstrações definidos para o paradigma de desenvolvimento orientado a aspectos, sendo também apresentado AspectJ, uma linguagem que estende Java para realização deste paradigma. No Capítulo 3, o trabalho é contextualizado com o estado da arte em publicações e pesquisas

desta área, enfatizando principalmente qualidade e métricas de software, abordagens de avaliação e ferramentas de DSOA. No Capítulo 4, é apresentado o método de avaliação proposto nesta dissertação e três novas métricas de separação de interesses. As regras heurísticas são detalhadas no Capítulo 5, descrevendo suas contribuições em apontar problemas de SI, acoplamento e coesão que não são facilmente identificados por uma avaliação direta sobre as métricas. No Capítulo 6, é apresentada a ferramenta de medição e avaliação que dá suporte automatizado ao método, incluindo decisões arquiteturais e detalhes de implementação. Os cinco estudos experimentais que contribuíram com a avaliação e evolução da abordagem são explorados no Capítulo 7. Finalmente, o Capítulo 8 apresenta as principais contribuições deste trabalho e possíveis direções para trabalhos futuros.