# Uma Revisão Sistemática sobre Teste de Software Orientado a Aspectos

**Fabiano Cutigi Ferrari**[1]*, **José Carlos Maldonado**[1]

[1]Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo
Caixa Postal 668 – CEP 13560-970 – São Carlos - SP – Brasil

{ferrari,jcmaldon}@icmc.usp.br

***Abstract.*** *Aspect-Oriented Programming has brought up several benefits to the software development process. However, as well as other development methodologies, it has also brought up new challenges to the testing activity. This paper describes the results of a systematic review performed aiming at identifying works regarding software testing techniques and criteria applied to aspect-oriented software. The results show that a variety of works have proposed the definition of testing criteria based on traditional testing techniques. Moreover, we could also observe that most of these approaches lack validation.The systematic review will serve as a basis for new research related to aspect-oriented software testing, including the definition and evaluation of testing criteria, automated support tools and empirical studies.*

***Resumo.*** *A Programação Orientada a Aspectos trouxe benefícios para o desenvolvimento de software e, como toda nova metodologia de desenvolvimento, novos desafios para a atividade de teste. Neste artigo são apresentados os resultados de uma revisão sistemática cujo objetivo foi identificar os trabalhos que abordam a aplicação de técnicas e critérios de teste no contexto de software orientado a aspectos. Os resultados mostram que diversos trabalhos têm enfatizado a definição de critérios para as técnicas tradicionais de teste. Além disso, pode-se observar que existe pouca validação dos trabalhos propostos. Os resultados servirão de base para a condução de novos trabalhos relacionados, incluindo a definição e avaliação de critérios de teste, ferramentas de apoio e estudos experimentais.*

## 1. Introdução

A POA (Programação Orientada a Aspectos) surgiu no final da década de 90 como uma possível solução para algumas dificuldades enfrentadas no desenvolvimento de software, principalmente relacionadas à separação de interesses envolvidos no software. A idéia principal é possibilitar que interesses que se encontram espalhados ou entrelaçados sejam implementados tão separadamente quanto possível dos demais (Kiczales et al. 1997).

Os esforços iniciais dedicados à pesquisa envolvendo POA concentraram-se no estabelecimento dos conceitos e em como implementá-los nas tecnologias de apoio. Entretanto, pode-se observar que existe uma preocupação com a garantia da qualidade de software OA (Orientado a Aspectos), mais especificamente com a atividade de teste. A despeito dos benefícios trazidos pela POA para o desenvolvimento de software, sua simples adoção não evita que defeitos sejam introduzidos ao longo do processo de desenvolvimento (Alexander et al. 2004). Dependências implícitas e explícitas entre aspectos e módulos tradicionais (classes, no caso de programas orientados a objetos) resultam em sistemas com novos desafios para a atividade de teste, incluindo novas fontes de defeitos.

*Com auxílio financeiro da FAPESP – Fundação de Amparo à Pesquisa do Estado de São Paulo

Técnicas e critérios tradicionalmente aplicados em software desenvolvido sob os paradigmas procedimental e OO (Orientado a Objetos) têm sido investigados no contexto de software OA, e adaptações têm sido propostas. Além disso, novos critérios baseados em técnicas tradicionais como, por exemplo, teste estrutural e baseado em modelos de estados, têm sido propostos para tratar de características específicas de software OA.

Este trabalho apresenta os resultados de uma revisão sistemática sobre teste de software OA. Uma revisão sistemática consiste em um meio de identificação, avaliação e interpretação de todos os trabalhos de pesquisa relevantes e disponíveis sobre uma questão de pesquisa, tópico ou fenômeno de interesse (Kitchenham 2004). Apesar de demandar maior esforço do que uma revisão tradicional, uma revisão sistemática tende a evitar viés por parte dos revisores, apresentando também outras vantagens como a possibilidade de ser auditada e replicada.

O principal objetivo da revisão apresentada neste trabalho é a identificação de trabalhos que abordam a aplicação de técnicas e critérios de teste tradicionais ou novos no contexto de software OA. Além disso, a revisão visa também a identificar os tipos de defeitos OA[1] caracterizados nesses trabalhos, e quais desses trabalhos têm apresentado estudos experimentais para demonstrar a efetividade da abordagem proposta. Neste trabalho, enfatizam-se os resultados relativos às técnicas e critérios de teste abordados.

Ressalta-se que não foram identificados outros trabalhos na literatura que abordem revisões formais com o mesmo objetivo da revisão apresentada neste trabalho, fato que pôde ser confirmado durante a condução da revisão sistemática realizada. Naqvi et al. (2005) avaliam três abordagens de teste propostas na literatura em relação à capacidade de detectar tipos de defeitos caracterizados em uma taxonomia de defeitos candidata para software OA (Alexander et al. 2004). Entretanto, esse trabalho limita-se a essas três abordagens, embora diversas outras já haviam sido propostas até então. Em geral, os demais trabalhos envolvendo teste de software OA apresentam revisões limitadas a poucos trabalhos relacionados.

O restante deste trabalho está assim organizado: na Seção 2 são apresentados os principais pontos do planejamento e da condução da revisão sistemática realizada. As principais características dos trabalhos selecionados são apresentadas na Seção 3. Em seguida, na Seção 4, é apresentada uma análise dos dados coletados durante a leitura dos trabalhos. Por fim, na Seção 5, é apresentada a conclusão deste trabalho, em relação tanto à experiência adquirida na condução da revisão sistemática quanto aos resultados obtidos após a análise dos trabalhos recuperados.

## 2. Planejamento e Condução da Revisão

O planejamento da revisão sistemática foi realizado de acordo com o modelo de protocolo apresentado por Biolchini et al. (2005). Por motivo de limitação de espaço, somente os pontos mais importantes do plano e da condução da revisão são apresentados nesta seção. Mais detalhes, incluindo as estratégias de busca para cada fonte selecionada e os resultados individuais de cada fonte, são apresentados em um relatório técnico que se encontra em fase de elaboração (Ferrari and Maldonado 2006).

O plano foi revisado por um especialista, e as sugestões de ajuste foram discutidas com os revisores e implementadas no plano final. A seguir são apresentados os principais pontos do plano:

[1]Neste trabalho, "defeito OA" é utilizado como sinônimo de "defeito característico de software OA".

**Objetivo:** Identificar e analisar as técnicas e critérios de teste que têm sido aplicados no teste de software OA. A qualidade em si dos trabalhos não será um fator fundamental para a seleção. Devido à POA tratar-se de uma abordagem relativamente recente para desenvolvimento de software, poucos ainda são os trabalhos relacionados envolvendo teste. Pretende-se inicialmente identificar todos eles, e fatores de qualidade serão considerados futuramente.

**Questões de Pesquisa:** Uma questão de pesquisa principal e três questões secundárias foram elaboradas para atender aos objetivos propostos. Cada uma delas possui seu próprio critério de inclusão de trabalhos:

- *Questão Primária*: Quais técnicas e critérios de teste de software têm sido aplicados no teste de software OA?
  *Critério*: Aplicação de técnica/critério de teste em software OA.
- *Questão Secundária 1*: Dentre as técnicas e critérios de teste aplicados no teste de software OA, quais são específicas para esse tipo de software?
  *Critério*: Proposição de nova técnica/critério específicos para teste de software OA.
- *Questão Secundária 2*: Quais tipos de defeitos OA têm sido caracterizados nos trabalhos relacionados com o teste de software OA, incluindo taxonomias e modelos de defeitos?
  *Critério*: Caracterização de tipos de defeitos OA.
- Questão Secundária 3: Quais tipos de estudos experimentais vêm sendo realizados nos trabalhos relacionados com o teste de software OA como forma de validação das abordagens propostas?
  *Critério*: Realização de estudos experimentais para validar as técnicas e critérios investigados.

**Estratégia de Busca para Seleção dos Estudos Primários:** A estratégia de busca e seleção dos estudos primários é definida de acordo com as fontes de trabalhos e as línguas de redação selecionadas, e de acordo com as palavras-chave para a revisão:

- *Fontes*: bases de dados eletrônicas (IEEE, ACM e Springer), máquinas de busca eletrônica (Google e Scirus), anais de eventos relacionados e consulta a especialistas.
- *Língua dos trabalhos*: inglês e português. Inglês por essa ser a língua internacionalmente aceita para a redação de trabalhos científicos. Português porque sua exclusão automaticamente excluiria trabalhos relevantes de autoria do próprio grupo de pesquisa dos revisores. Esses trabalhos envolvem tanto a definição de critérios de teste de software OA quanto a implementação de ferramentas de apoio automatizado.
- *Palavras-chave*: *aspect-oriented* e *software testing*, incluindo termos simples e compostos relacionados.

**Seleção Preliminar:** Construir uma *string* de busca a partir dos termos relacionados às palavras-chave para ser submetida às máquinas de busca das fontes selecionadas. Realizar a leitura dos resumos de cada um dos trabalhos recuperados. Cada trabalho relevante para a revisão deve ser pré-selecionado para leitura completa. Ocorrendo falta de consenso sobre a relevância de um trabalho, ele deve ser colocado em espera, e sua inclusão ou exclusão será definida em reuniões entre os revisores.

**Seleção Final e Extração dos Resultados:** A leitura completa dos trabalhos pré-selecionados deve ser realizada por pelo menos um dos revisores, que fica encarregado de fazer uma síntese do trabalho, destacando a abordagem de teste apresentada e os conceitos subjacentes envolvidos.

### 2.1. Condução da Revisão

A revisão foi conduzida por um período de três meses (Maio/2006 a Julho/2006). Ao todo foram recuperados 260 trabalhos, para os quais foi realizada a leitura do resumo. Nesse passo, para os casos em que não se pôde definir a relevância dos trabalhos somente com base na leitura dos resumos, outras seções do texto foram analisadas e a decisão sobre a pré-seleção do trabalho foi tomada. Observa-se que não foi necessária a realização de reunião entre os revisores para a decisão de inclusão ou exclusão de trabalhos.

Inicialmente uma *string* de busca foi construída com a combinação lógica dos termos relacionados às palavras-chave. A *string* final ficou assim definida:

```
(aspect-oriented software OR aspect-oriented application OR aspect-oriented app OR
aspect-oriented program OR AOP) AND (testing OR testing criterion OR testing technique
OR fault OR defect OR error OR incorrect OR fault model OR failure)
```

Cada uma das fontes selecionadas (e suas respectivas máquinas de busca) possui particularidades para a construção de *strings*. Para cada uma das fontes, a *string* de busca foi adaptada ou particionada em *strings* menores, contendo os termos mais relevantes para as questões de pesquisa.

Na seleção final, 25 trabalhos foram identificados como relevantes para os objetivos da revisão. As principais características desses trabalhos são apresentadas na próxima seção. Alguns pontos que merecem destaque em relação à seleção final são:

- Foram selecionados trabalhos que têm como foco principal a proposição ou aplicação de técnicas e critérios de teste em software OA, ou que explicitamente tratam de tipos de defeitos OA.
- Trabalhos cujo foco principal diferem desse foco não foram incluídos. Dentre esses, estão trabalhos que apresentam tecnologias de apoio ao teste ou outras atividades relacionadas ao teste como, por exemplo, teste de regressão, depuração ou geração de casos de teste sem o apoio de uma técnica ou critério específico.

## 3. Visão Geral dos Trabalhos Analisados

Nesta seção são apresentadas as principais características dos trabalhos que foram selecionados. Por motivo de limitação de espaço, optou-se por apresentar as características dos trabalhos sem apresentar a referência de cada trabalho, o que geraria uma lista de referências extensa para um trabalho com número de páginas limitado. As referências completas são apresentadas no trabalho de Ferrari e Maldonado (2006).

Os trabalhos estão agrupados de acordo com a técnica de teste abordada. Nos 25 trabalhos selecionados, pode-se observar que as principais técnicas de teste têm sido exploradas. Dentre eles:

- Três envolvem as técnicas estrutural e baseada em defeitos;
- Um envolve as técnicas estrutural e funcional;
- Cinco envolvem a técnica estrutural;
- Três envolvem as técnicas estrutural e baseada em modelos;
- Sete envolvem a técnica baseada em modelos;
- Um envolve a técnica funcional;
- Cinco não envolvem alguma técnica em específico, mas estão relacionadas a pelo menos uma das questões de pesquisa.

Ressalta-se que para algumas abordagens de teste propostas, mais de um trabalho foi identificado. Alguns desses trabalhos consistem em relatórios técnicos que foram posteriormente publicados na forma de artigo em algum evento relacionado. Outros, por sua vez, consistem em trabalhos que foram estendidos ou complementados em trabalhos posteriores. Trabalhos que abordam mais de uma técnica de teste propõem a utilização dessas técnicas de forma complementar.

- **Teste Estrutural e Baseado em Defeitos**

Duas abordagens baseadas nas técnicas de teste estrutural e baseada defeitos foram identificadas em três trabalhos selecionados.

Uma das abordagens, descrita em dois trabalhos, tenta adequar os tipos de defeitos que podem ser encontrados à taxonomia de defeitos proposta por Alexander et al. (2004). Define-se um conjunto de critérios estruturais para atuar em diferentes níveis de granularidade, desde cobertura de comandos até a cobertura de execução de todos os comportamentos implementados nos aspectos. No contexto de teste baseado em defeitos, um conjunto de operadores de mutação é definido para atuar nas definições de escopo dos conjuntos de junção e nas definições de precedência de aspectos.

A outra abordagem é específica para tratar defeitos relacionados aos descritores de conjuntos de junção. A estratégia proposta consiste em identificar os pontos de junção selecionados não desejados, com apoio de teste estrutural baseado em fluxo de controle, e identificar pontos de junção negligenciados, com apoio de teste baseado em Análise de Mutantes. Os modelos subjacentes de cada técnica (grafos de fluxo de controle e operadores de mutação, respectivamente) são empregados como apoio à abordagem proposta.

- **Teste Estrutural e Funcional**

Uma abordagem de teste baseada nas técnicas estrutural e funcional é apresentada em um dos trabalhos selecionados. Consiste em uma estratégia de desenvolvimento de software OA a partir da refatoração de software OO. Nessa estratégia está incluído um procedimento de teste para garantir o correto comportamento do software independentemente da refatoração. A estratégia consiste de cinco passos, podendo-se destacar o projeto do conjunto de testes que seja sensível a exercitar as porções de código refatoradas. Uma taxonomia de defeitos também é proposta no trabalho, incluindo defeitos relacionados a declarações inter-tipos, definição de conjuntos de junção e definição e implementação de adendos. Critérios de teste estrutural e funcional tradicionais podem ser utilizados para derivar os requisitos de teste.

- **Teste Estrutural**

Duas linhas de trabalhos envolvendo o teste estrutural de software OA foram identificadas, sendo descritas em cinco trabalhos selecionados.

A primeira linha de trabalhos apresenta uma abordagem de teste estrutural de unidade de software OA escrito em AspectJ (Kiczales et al. 2001). Dois trabalhos descrevem essa abordagem, que é baseada nas primeiras versões da linguagem, de acordo com a estratégia corrente de combinação de aspectos e classes. As unidades consideradas são as classes e os aspectos que compõem a aplicação completa. Métodos e adendos são considerados módulos que compõem as unidades. A partir desses módulos, são definidos três níveis de teste: (i) intra-módulo; (ii) inter-módulo; e (iii) intra-classe ou intra-aspecto. Os trabalhos seguem a mesma linha do trabalho de Harrold and Rothermel (1994), que definem diferentes níveis de teste de software OO. Um conjunto de grafos de fluxo de controle acrescido de informações

sobre o fluxo de dados é definido, a partir dos quais são derivados pares definição-uso (def-uso) para cada um dos níveis de teste abordados.

A segunda linha de trabalhos explora a extração de informações dos *bytecodes* Java obtidos após a combinação dos aspectos com as classes base. Esses trabalhos abordam as fases de teste de unidade e integração, considerando os métodos e adendos como as unidades da aplicação. Três trabalhos descrevem a abordagem, dois envolvendo o teste de unidade e um envolvendo o teste de integração. Novamente, considera-se a implementação corrente da linguagem AspectJ (nesse caso, versões mais recentes da linguagem) para a extração de informações dos *bytecodes*. Para o teste de unidade, o grafo AODU (*Aspect-Oriented Def-Use*) é definido, incluindo nós transversais, que representam os pontos de junção capturados pelos aspectos. Um conjunto de critérios baseados em fluxo de controle e fluxo de dados é proposto, e uma ferramenta para apoiar a aplicação dos critérios foi implementada. Para o teste de integração, o grafo MADU (*Method-Advice Def-Use*) é definido, resultando da composição do grafo AODU de um método com os grafos AODU dos adendos que alteram seu comportamento. Com base nas diferentes interações de dados que podem ocorrer entre métodos e adendos, foram definidos quatro critérios envolvendo pares definição-uso inter-unidade de variáveis.

- **Teste Estrutural e Baseado em Modelos de Estados**

Duas linhas de trabalhos envolvendo o teste estrutural e baseado em modelos de estados foram identificadas, sendo descritas em três trabalhos selecionados.

A primeira linha, descrita em dois trabalhos, consiste em uma abordagem apoiada por um *framework* que é utilizado para gerar classes empacotadoras (*wrappers*) para os aspectos e para as classes afetadas por eles. Essas classes empacotadoras são submetidas para ferramentas que geram casos de teste considerando cobertura de comandos e de transições de estados. O *framework* também apóia a instrumentação e execução dos testes, calculando a cobertura de teste obtida. A principal diferença entre os dois trabalhos está nas partes de código enfatizadas nas abordagens. Enquanto em um deles o foco é no teste de métodos afetados por adendos, adendos e métodos inter-tipo declarados, no outro o foco é no teste de adendos, métodos inter-tipo declarados ou métodos de aspectos.

O terceiro trabalho apresenta uma abordagem de teste baseada em um modelo de estados intitulado ASM (*Aspectual State Model*). O ASM contém notações para representar as construções básicas de POA e os novos estados resultantes da atuação dos aspectos. Uma árvore de transições de estados é derivada do modelo de estados. Essa árvore é estendida de forma a incluir os grafos de fluxo de controle das operações que geram as transições de estados. Critérios baseados em fluxo de controle podem ser empregados em conjunto com os critérios de cobertura de transições de estados.

- **Teste Baseado em Modelos de Estados**

Quatro trabalhos apresentam abordagens para o teste baseado em modelos de estados. As abordagens são similares, e têm como base modelos de estados estendidos que incluem tanto estados referentes às classes base quanto estados resultantes da atuação dos aspectos sobre essas classes. O principal diferencial está no foco das abordagens. Enquanto três enfatizam o teste de uma classe e os aspectos que a afetam, a outra enfatiza o teste de aspectos que fazem a integração de uma classe base com classes utilizadas, por exemplo, em declarações inter-tipos. Todas as abordagens propõem procedimentos incrementais para geração de testes, nos quais parte-se do modelo de estados considerando somente as classes

base e evolui-se para o modelo incluindo os estados inseridos ou modificados pelos aspectos. A geração dos testes é realizada a partir de uma árvore de transições de estados derivada do modelo, de acordo com critérios de cobertura de transições de estados.

- **Teste Baseado em Modelos UML**

Apesar do modelo de estados fazer parte do conjunto de modelos da UML, alguns trabalhos têm utilizado outros modelos da UML como base para a derivação de requisitos de teste. Dentre os trabalhos selecionados, três deles utilizam diagramas de interação da UML.

Um desses trabalhos apresenta uma extensão da UML para permitir representar os pontos onde os aspectos modificam o comportamento de uma classe base. A partir do diagrama de seqüência estendido, um grafo de mensagens é construído. Uma árvore de transições é derivada do grafo de mensagens, e os casos de teste são gerados para percorrer caminhos dessa árvore.

Uma abordagem baseada em diagramas de colaboração é proposta em dois outros trabalhos, tendo como foco o teste da integração de um ou mais aspectos com um grupo de objetos que colaboram entre si. A abordagem é iterativa, iniciando com o teste dos cenários de colaboração sem considerar os aspectos. Na segunda etapa, um a um os aspectos são analisados e o diagrama de colaboração é modificado para representar as novas seqüências resultantes da combinação do aspecto. Para a geração dos casos de teste, uma árvore de mensagens é derivada do diagrama de colaboração, representando as possíveis seqüências de operações. A partir dessa árvore, os casos de teste são gerados de acordo com um conjunto de critérios de cobertura de transições de estados definido nesses trabalhos.

- **Outros Trabalhos**

Além dos trabalhos já apresentados, outros seis trabalhos foram selecionados por estarem relacionados a pelo menos uma das questões de pesquisa definidas.

Três trabalhos propõem estratégias de teste nas quais as técnicas tradicionais podem ser empregadas. Nesses trabalhos, critérios de teste em alto nível são definidos, envolvendo características específicas de software OA como, por exemplo, a execução de todos os adendos ou de todos os métodos afetados por adendos.

Um dos trabalhos apresenta um conjunto de pequenos padrões de teste de software OA, envolvendo teste funcional, uso de ferramentas de visualização, transferência de lógica dos adendos para outras classes e uso de objetos *mock* para simular as classes base. De acordo com os padrões apresentados, a abordagem é direcionada para o teste de adendos e conjuntos de junção. Além do teste funcional, outras técnicas podem ser aplicadas com o apoio dos padrões apresentados.

Quatro trabalhos têm como foco principal a caracterização de tipos de defeitos OA. Quatro potenciais fontes de defeitos em um programa OA que já passou pelo processo de combinação são apresentadas. Baseada nessas fontes de defeitos, uma taxonomia candidata de defeitos OA é proposta e estendida. Um desses trabalhos enfatiza a necessidade de um modelo de defeitos no qual os tipos de defeitos sejam detalhados sintática e semanticamente, e um modelo de estrutura candidata para descrição de tipo de defeitos é sugerido.

Defeitos decorrentes da combinação de aspectos em código reutilizado que já possui aspectos combinados são discutidos em um dos trabalhos cujo foco é a caracterização de defeitos OA. Esses aspectos são chamados aspectos estrangeiros, e sua possibilidade de reutilização depende de apoio da tecnologia de implementação.

## 4. Análise dos Dados Coletados

Na Tabela 1 é apresentado um sumário dos trabalhos selecionados. Os dados apresentados na tabela foram organizados de acordo com as questões de pesquisa definidas para a revisão. Os trabalhos estão enumerados na primeira coluna.

**Tabela 1. Sumário dos trabalhos selecionados.**

| Trabalho | Técnica | Técnica Específica | Critério Específico | Estudo Experimental | Defeitos OA |
|---|---|---|---|---|---|
| 1,2 | estrutural e baseada em defeitos | não | sim | estudo de caso | não |
| 3 | estrutural e baseada em defeitos | não | não | não | sim |
| 4 | funcional e estrutural | não | não | estudo de caso | sim |
| 5,6 | estrutural | não | não | não | não |
| 7,8,9 | estrutural | não | sim | não | não |
| 10 | estrutural e baseada em modelos de estados | não | sim | estudo de caso | não |
| 11 | estrutural e baseada em modelos de estados | não | não | não | não |
| 12 | baseada em modelos de estados e estrutural | não | não | não | não |
| 13 | baseada em modelos de estados | não | não | não | não |
| 14 | baseada em modelos de estados | não | sim | não | sim |
| 15 | baseada em modelos de estados | não | não | estudo de caso | sim |
| 16 | baseada em modelos de estados | não | sim | não | não |
| 17 | baseada em modelos UML | não | não | não | não |
| 18 | baseada em modelos UML | não | sim | não | não |
| 19 | baseada em modelos UML | não | sim | não | sim |
| 20 | funcional | não | não | não | não |
| 21 | não enfatizam | não | não | não | sim |
| 22 | não enfatizam | não | sim | não | sim |
| 23 | não enfatizam | não | não | não | sim |
| 24 | não enfatizam | não | sim | não | sim |
| 25 | não enfatizam | não | sim | estudo de caso | não |

As colunas “Técnica” e “Técnica Específica” informam, respectivamente, a técnica de teste utilizada nos trabalhos e se essa técnica é ou não específica para o teste de software OA. Como se pode observar, todos os trabalhos analisados aplicam as técnicas tradicionais de teste, seja de forma direta ou adequando-as para o contexto de software OA. Observa-se também que alguns trabalhos abordam mais de uma técnica de teste. Outros, por sua vez, concentram-se em propor estratégias de teste na qual o testador pode adotar a técnica de teste mais conveniente.

A coluna “Critério Específico” informa se no trabalho é proposto algum critério de teste específico para o teste de software OA. Nota-se que em 13 trabalhos houve a preocupação de se definirem novos critérios que enfatizem características particulares de

software OA. Nos outros trabalhos, os autores em geral sugerem o uso de critérios tradicionais adaptados para a contexto de software OA.

A coluna "Estudo Experimental" indica quais trabalhos apresentam algum tipo de estudo experimental para validar a abordagem proposta. De acordo com os dados coletados, nota-se que poucos trabalhos realizam algum tipo de estudo (apenas cinco), todos eles limitados a estudos de caso. Os demais trabalhos utilizam exemplos para demonstrar a aplicação das abordagens, sem conduzir estudos mais detalhados para comprovar sua efetividade.

Por fim, a coluna "Defeitos OA" contém a indicação de trabalhos que caracterizam tipos de defeitos OA. Nota-se que também existe uma preocupação em se caracterizar tipos de defeitos específicos introduzidos pela POA e tecnologias de apoio, o que pode ser visto em nove trabalhos selecionados.

Mais detalhes sobre os trabalhos, incluindo a descrição dos critérios de teste definidos e os tipos de defeitos OA caracterizados são apresentados no trabalho de Ferrari e Maldonado (2006).

## 5. Conclusão e Trabalhos Futuros

As conclusões obtidas com a realização da revisão sistemática apresentada neste trabalho podem ser exploradas sob duas perspectivas. A primeira em relação à condução da revisão em si, e a segunda em relação às abordagens apresentadas nos trabalhos selecionados.

A principal dificuldade enfrentada durante a condução da revisão está relacionada aos mecanismos de busca atualmente disponíveis. Além de não existir uma forma padrão de consultar as bases indexadas, trabalhos muito citados (relatório técnicos, por exemplo) não aparecem nessas bases, sendo recuperados somente em máquinas de busca convencionais.

Em relação à estratégia de busca adotada, as citações observadas nos trabalhos selecionados constituem indícios de que as buscas tiveram a amplitude desejada. Todas as citações encontradas nos trabalhos selecionados foram recuperadas com as *strings* construídas.

O trabalho de Alexander et al. (2004) foi o mais citado dentre os trabalhos selecionados. Dessa forma, nota-se que a decisão de incluir repositórios não indexados pode ser importante para uma revisão sistemática, na qual pretende-se recuperar trabalhos de reconhecida relevância para uma pesquisa específica. Trata-se de um relatório técnico que não foi publicado em repositórios indexados como IEEE e ACM, mas que após disponibilizado tem sido citado por quase todos os trabalhos relacionados ao teste de software OA. Por outro lado, observou-se que alguns dos trabalhos classificados como relevantes não são citados em boa parte dos demais, indicando que a existência de critérios sistemáticos para a inclusão e exclusão de trabalhos é efetivamente importante para evitar viés na revisão.

Em relação aos resultados obtidos na revisão, observa-se que alguns trabalhos têm enfatizado as técnicas tradicionais com o apoio de critérios específicos para software OA. Outros, por sua vez, sugerem a adaptação dos critérios tradicionais da referida técnica. Uma parcela considerável de trabalhos apresenta caracterizações de tipos de defeitos OA. Entretanto, poucos trabalhos têm enfatizado a realização de estudos experimentais como forma de validar os critérios propostos e os tipos de defeitos caracterizados.

Pode-se observar que as abordagens para teste de software OA em geral não envolvem o teste de todas as características de programas OA. Enquanto algumas são direcionadas para o teste de integração de adendos com métodos, outras são direcionadas para o teste de conjuntos de junção ou de comportamentos que concorrem por pontos de junção em comum.

A maioria das abordagens de teste de software OA é baseada em características da linguagem AspectJ. Entretanto, os conceitos presentes nas abordagens analisadas podem ser aplicados no teste de software OA em geral, independentemente da tecnologia de desenvolvimento adotada. Primeiro porque boa parte das iniciativas de implementação de tecnologias de apoio à POA têm como base a linguagem AspectJ. Segundo porque as tecnologias de apoio, de forma geral, implementam os elementos básicos da POA. Essas tecnologias fornecem um meio de modularizar comportamentos transversais e implementam um mecanismo de quantificação, que permite definir os pontos da execução do software em que esses comportamentos serão executados (Filman and Friedman 2000).

A elaboração de um relatório técnico contendo o planejamento completo, o material utilizado e os resultados obtidos na condução da revisão sistemática encontra-se em andamento. Nesse relatório, cada um dos trabalhos selecionados é apresentado em mais detalhes, incluindo os critérios de teste definidos e os tipos de defeitos OA caracterizados. Esses resultados serão utilizados como base para a condução de novos trabalhos relacionados ao teste de software OA, incluindo a definição de critérios de teste, mecanismos de apoio automatizado e realização de estudos experimentais.

## Referências


[Alexander et al. 2004] Alexander, R. T., Bieman, J. M., and Andrews, A. A. (2004). Towards the systematic testing of aspect-oriented programs. Tech. Report CS-04-105, Dept. of Computer Science, Colorado State University, Fort Collins/Colorado - USA.

[Biolchini et al. 2005] Biolchini, J., Mian, P. G., Natali, A. C. C., and Travassos, G. H. (2005). Systematic review in software engineering. Tech. Report RT-ES 679/05, Systems Engineering and Computer Science Dept., COPPE/UFRJ, Rio de Janeiro/RJ - Brazil.

[Ferrari and Maldonado 2006] Ferrari, F. C. and Maldonado, J. C. (2006). Teste de software orientado a aspectos: Uma revisão sistemática. Relatório técnico em preparação, Departamento de Sistemas de Computação, ICMC/USP, São Carlos/SP - Brasil.

[Filman and Friedman 2000] Filman, R. and Friedman, D. (2000). Aspect-oriented programming is quantification and obliviousness. In *Proceedings of the Workshop on Advanced Separation of Concerns – held in conjunction with OOPSLA'2000*, pages 21–35, Minneapolis - USA.

[Harrold and Rothermel 1994] Harrold, M. J. and Rothermel, G. (1994). Performing data flow testing on classes. In *Proceedings of the* $2^{nd}$ *ACM SIGSOFT Symposium on Foundations of Software Engineering*, pages 154–163, New York, NY. ACM Press.

[Kiczales et al. 2001] Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., and Griswold, W. G. (2001). Getting started with AspectJ. *Communications of the ACM*, 44(10):59–65.

[Kiczales et al. 1997] Kiczales, G., Irwin, J., Lamping, J., Loingtier, J.-M., Lopes, C., Maeda, C., and Menhdhekar, A. (1997). Aspect-oriented programming. In Akşit, M. and Matsuoka, S., editors, *Proceedings of the European Conference on Object-Oriented Programming*, volume 1241, pages 220–242. Springer-Verlag.

[Kitchenham 2004] Kitchenham, B. (2004). Procedures for performing systematic reviews. Joint Technical Report TR/SE-0401 (Keele) – 0400011T.1 (NICTA), Software Engineering Group - Department of Computer Science - Keele University and Empirical Software Engineering - National ICT Australia Ltd, Keele/Staffs-UK and Eversleigh-Australia.

[Naqvi et al. 2006] Naqvi, S. A. A., Ali, S., and Khan, M. U. (2006). An evaluation of aspect oriented testing techniques. In *Proceedings of the IEEE Symposium on Emerging Technologies*, pages 461–466, Islamabad - Pakistan. ACM Press.