Uma Revisão Sistemática sobre a atividade de Teste de Software no contexto de Métodos Ágeis

André Abe Vicente, Marcio Eduardo Delamaro e José Carlos Maldonado

Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo
Caixa Postal: 668 – 13560-970 – São Carlos – SP – Brazil
{avicente,delamaro,jcmaldon}@icmc.usp.br

Abstract. Software testing activity has a great importance on agile development methods. In this context, strategies for testing have been created and adapted, trying to eliminate non-productive aspects of testing, identifying best practices and, mainly, ways to automate the testing process. This papers presents a systematic review which main goal is to identify strategies, techniques and testing criteria applied in agile methods. We also investigated issues related to the importance of testing activity in these methods, testing tools used and experimental studies that demonstrate the implications of an agile testing approach. This study evinced an emphasis on the use of test-driven development strategy for unit and integration testing and acceptance testing (or business testing) with the customer, as well as a concern with experimental studies that can measure the benefits and difficulties of using such strategies, and the necessity of testing tools in the context of agile methods.

1 Introdução

Métodos de desenvolvimento tradicionais são centrados no processo, no qual acredita-se que sistemas são completamente especificáveis, previsíveis e podem ser feitos através de um planejamento meticuloso e extensivo [1], que gera uma grande quantidade de documentação.

Para enfrentar problemas com prazos e complexidade de processos tradicionais da engenharia de software, diversos métodos de desenvolvimento ágil estão sendo utilizados. Esses métodos possuem como principal objetivo a satisfação do cliente, preocupando-se com a entrega incremental de software desde as etapas iniciais de desenvolvimento, a minimização de produtos de trabalho de engenharia de software e a simplicidade global no desenvolvimento.

Métodos ágeis foram desenvolvidos para promover a entrega rápida de código útil por meio do desenvolvimento em pequenos ciclos iterativos dirigidos a características do produto. Além disso são utilizados períodos de reflexão e introspecção, tomada de decisão de forma colaborativa, incorporação de *feedback*, mudanças rápidas e contínua integração dessas mudanças ao repositório de código do sistema em desenvolvimento [1]. Os métodos ágeis mudam o foco da documentação do projeto, fortemente utilizada em métodos tradicionais, para técnicas que priorizam o desenvolvimento de código-fonte e testes.

Diversos métodos estão sendo utilizados em projetos ágeis [2,3]. Como exemplos mais conhecidos, citam-se o *eXtreme Programming* (XP), o *Scrum*, a família

Crystal, o *Feature Driven Development* (FDD), o *Adaptative Software Development* (ASD), o *Dynamic System Development Method* (DSDM) e o *Lean Software Development*. Todos esses métodos seguem princípios semelhantes. O que os diferencia são as suas práticas e a forma de condução do processo de desenvolvimento, sendo que os mais utilizados atualmente são o XP e o Scrum [4].

Em projetos tradicionais, o controle de qualidade é feito por meio de um planejamento detalhado e controle rígido, além de uma grande quantidade de testes que são executados de forma tardia no ciclo de vida do software. Em projetos ágeis, para satisfazer o cliente por meio da entrega rápida e contínua de software de qualidade, são utilizadas diversas práticas [5], como a presença do cliente, a programação em pares e a refatoração. Nesses projetos, existe uma preocupação com a melhoria constante do processo e do produto durante todo o ciclo de desenvolvimento do software. O progresso do projeto é avaliado diariamente por meio de um controle contínuo de requisitos, projeto e soluções, além de testes executados durante todo o ciclo de vida [1]. Além disso, a atividade de testes assume um papel importante no projeto do software [6] e também na forma de documentar o software de maneira não ambígua, sempre atualizada de acordo com o andamento do projeto e que pode ser executada rapidamente.

Esse artigo investiga e descreve como a atividade de teste de software pode ser aplicada dentro do contexto de métodos ágeis. Para caracterizar a atividade de testes ágeis, foi conduzida uma revisão sistemática (RS) para identificar trabalhos que descrevam estratégias, técnicas ou critérios de teste utilizados em projetos ágeis, ferramentas de apoio e resultados experimentais da aplicação de testes ágeis. Os estudos mais relevantes foram categorizados e analisados. Nesse contexto, a motivação desse trabalho é servir como base para equipes de desenvolvimento e pesquisadores da área de teste de software e métodos ágeis que necessitem conhecer as particularidades relacionadas à atividade de testes executada de forma ágil.

O restante do artigo esta organizado da seguinte forma: na Seção 2 são discutidas as motivações para a execução de uma revisão sistemática. Na Seção 3, é apresentado como os principais métodos de desenvolvimento ágil tratam a atividade de teste do ponto de vista de fases e práticas. Na Seção 4 são apresentadas as atividades executadas durante a condução desta revisão sistemática e, por fim, na Seção 5 são apresentadas as considerações finais em relação à condução da revisão sistemática e trabalhos futuros.

2 Revisão Sistemática

Devido à grande diversidade de tecnologias existentes no mercado, a escolha de quais aplicar em um projeto de software não é uma tarefa fácil para gerentes de projeto ou desenvolvedores. Nessa tarefa precisam ser considerados os problemas decorrentes dessas tecnologias e também como tirar proveito dos benefícios prometidos. No entanto, essa escolha pode ser dificultada pela falta de evidências relacionadas à tecnologia que possam confirmar sua capacidade de combinação, limites, qualidades, custos e riscos inerentes [7]. Nesse contexto, a Engenharia de Software Baseada em Evidência (ESBE) [8] tem como objetivo melhorar a tomada de decisão relacionada ao desenvolvimento de software e manutenção, integrando as melhores evidências atuais de pesquisas com experiência prática e valores humanos. A EBSE faz o uso de dois tipos de estudos complementares

[7]: estudos primários, que visam caracterizar uma determinada tecnologia em uso dentro de um contexto específico (experimentos, estudos de caso e *surveys*) e estudos secundários, que visam identificar, avaliar e interpretar resultados relevantes a um tópico de pesquisa, fenômeno de interesse ou questão de pesquisa.

A revisão sistemática, é um tipo de estudo secundário. Diferentemente de revisões tradicionais, a RS é um método de pesquisa explícito e rigoroso que procura identificar o conhecimento científico em uma determinada área por meio da coleta, combinação e avaliação crítica de descobertas de diversos estudos já realizados [9]. As razões mais comuns para condução de uma RS são [7]: **(i)** resumir evidências existentes relacionadas a um comportamento ou tecnologia;**(ii)** identificar questões de pesquisa em aberto com o objetivo de sugerir áreas para futuras investigações; **(iii)** fornecer um *framework* de conhecimento para definir apropriadamente novas atividades de pesquisa; **(iv)** examinar o nível que hipóteses teóricas suportam ou contradizem evidências experimentais ou até mesmo para apoiar a geração de novas hipóteses.

O processo de uma RS deve seguir os seguintes passos (Figura 1): (1) *Planejamento*: definição dos objetivos e de um protocolo de revisão que especifica a questão da pesquisa a ser conduzida e os métodos que serão utilizados para executar a revisão; (2) *Execução*: identificação de estudos primários (buscas), avaliação dos estudos primários (requer critérios de inclusão e exclusão explícitos) e seleção final; (3) *Análise dos Resultados*: extração e síntese dos dados. Além disso, a atividade de empacotamento deve ser executada durante todas as fases com o objetivo de armazenar os resultados intermediários e finais da RS.

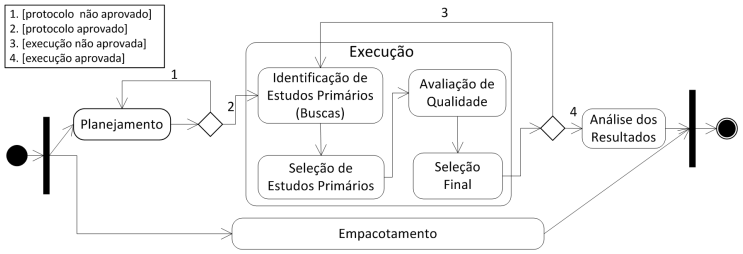

Figura 1. Processo de Condução de Revisão Sistemática [10]

A revisão sistemática deste trabalho tem como objetivo verificar como a atividade de testes vem sendo tratada dentro de métodos ágeis. Outros trabalhos relacionados também selecionaram estudos a respeito de métodos ágeis, dentre eles o trabalho de Abrantes e Travassos [11], que procurou identificar características de agilidade dentro de métodos ágeis e o trabalho de Dybå e Dings[12] que conduziu uma RS de estudos experimentais de métodos ágeis de desenvolvimento, na qual foram investigados os benefícios e limitações desses métodos, a qualidade das evidências providas por esses estudos, além de implicações para a pesquisa e prática de métodos ágeis. Também foram comparados estudos experimentais relacionados à prática de programação pareada [13].

3 Teste de Software no contexto de Métodos Ágeis

A atividade de teste dentro de métodos ágeis tem como objetivo evitar que a qualidade do produto e a condução do projeto seja afetada por processos menos formais de documentação e projeto em relação aos métodos tradicionais [14]. Além dos testes validarem novas funcionalidades e mudanças de forma contínua, eles também podem guiar o projeto e documentar o software [6]. A documentação do software utilizando testes, além de não conter ambiguidades como uma documentação descrita em linguagem natural, sempre estará atualizada e poderá ser executada tanto pelo desenvolvedor (*e.g.* com ferramentas de teste de unidade) como pelos clientes (*e.g.* com ferramentas de teste de aceitação).

Segundo Perry [15], a melhor forma de criar um processo ágil de teste de software é focar em eliminar aspectos de teste não produtivos e identificar e incorporar boas práticas de testes. Dentre as práticas de teste existentes em métodos ágeis, pode-se citar testes de unidade utilizando *Test Driven Development* (TDD) [16], testes de integração contínuos, testes de aceitação com o cliente e testes de regressão associados à prática de refatoração (melhoria do código). Todas essas práticas de testes devem ser executadas preferencialmente de forma automatizada, buscando a agilidade no processo de testes.

O TDD [6] é uma estratégia de desenvolvimento de software que requer que testes automatizados sejam escritos antes do desenvolvimento de código funcional em iterações pequenas e rápidas. Neste sentido o desenvolvimento dirigido a testes pode ser utilizado para explorar, projetar, desenvolver e testar o software. Estudos experimentais [17,18] mostram que o TDD diminui a quantidade de defeitos no software, precisando de menos tempo para a atividade de depuração e deixando os usuários mais satisfeitos com os produtos entregues. Os testes de aceitação com o cliente são testes em alto-nível que procuram validar junto ao cliente as funcionalidades do software. Essa técnica também pode incorporar a idéia de se desenvolver os casos de teste antes do código, chamado de Teste de Aceitação Automatizados (*Aceptance Test Driven Development*- ATDD) [19].

Diferentemente de métodos tradicionais, nos quais os testes ocorrem mais tarde dentro do processo de desenvolvimento, os testes ágeis devem ocorrer de forma frequente e procurando detectar defeitos o mais cedo possível, dentro de ciclos de desenvolvimento iterativos e curtos, com um constante *feedback* do cliente. Equipes de desenvolvimento ágeis podem parar de desenvolver novas funcionalidades durante um período, mudando o foco para a estabilidade do código. E também, corrigindo defeitos, refatorando código e executando testes de regressão para se assegurarem que não houve nenhuma alteração no código que tenha afetado o funcionamento do sistema em desenvolvimento [20]. Além disso, há uma preocupação em testar sob o ponto de vista do cliente por meio de testes de aceitação. Cada história do usuário (pequenos requisitos) deve ter um ou mais casos de teste de aceitação associados, que devem validar o software.

Em relação às fases de teste, métodos ágeis tratam-nas de maneiras diferente, sendo que as fases mais enfatizadas são as de teste de unidade e teste de aceitação. Como descrito anteriormente, os ciclos de desenvolvimento em projetos ágeis são iterativos e curtos e os testes são executados frequentemente, durante todo o ciclo de vida do software. A partir de dois estudos [2,21], é descrito na Tabela 1 um breve comparativo da atividade de verificação, validação e testes (VV&T) nos principais métodos ágeis.

Tabela 1. Técnicas e práticas de VV&T em Métodos Ágeis.

Método	Técnicas/Práticas relacionadas a VV&T
DSDM	- Testes integrados durante todo o ciclo de vida - Todo componente de software é testado pelos desenvolvedores e usuários assim que são desenvolvidos
XP	- Integração contínua e testes - *Test-Driven Development* para teste de unidade e integração - Testes de aceitação associados a histórias do usuário
Scrum	- *Builds* diários e testes (não especifica nenhuma técnica de teste) - Testes durante o *sprint* e Teste de sistema (após *sprint*)
Familia Crystal	- Testes são executados durante a iteração - Teste de regressão de funcionalidades automatizados - Nível de formalidade e documentação dos testes conforme o time e complexidade do projeto
ASD	- Teste faz parte da construção concorrente de componentes - Inspeções de código - Revisões para melhorar a qualidade do produto (Testes de Aceitação e Sistema)
FDD	- Inspeção do projeto e do código - Testes de unidade (após implementação)

4 Condução da Revisão Sistemática

Nesta seção serão descritas as fases de planejamento, execução e análise dos resultados dos trabalhos selecionados desta revisão sistemática.

4.1 Planejamento

Na fase de planejamento devem ser definidos os objetivos da pesquisa e a estratégia a ser utilizada para a busca dos estudos primários. Isso inclui a formulação das questões da busca, os termos a serem utilizados na busca e quais os recursos a serem procurados. Também são escolhidas as fontes a serem utilizadas na pesquisa e como os estudos (trabalhos) serão identificados por meio de critérios de seleção e exclusão [7]. Por fim é descrito como será feita a extração e síntese dos dados.

A revisão sistemática apresentada neste artigo teve como objetivo verificar como a atividade de teste vem sendo tratada dentro de métodos ágeis. Para isso, como questão primária procurou-se identificar e analisar quais estratégias, técnicas e critérios de teste são utilizados no contexto de tais métodos. Adicionalmente, foram protocoladas três questões secundárias: **(1)** qual a importância da atividade de teste em métodos ágeis, **(2)** quais ferramentas de apoio são utilizadas para testes automatizados em métodos ágeis e **(3)** quais os resultados de estudos experimentais para a atividade de teste em métodos ágeis que validam a abordagens ágeis de teste neste método.

A **população** selecionada foi o de métodos de desenvolvimento ágil, sendo que a intervenção considerada foi estratégias, técnicas e critérios de teste. Como **controle** foram utilizados artigos e trabalhos selecionados em revisões bibliográficas feitas anteriormente pelo grupo de pesquisa a respeito do tema. O **resultado** desejado são técnicas, critérios e ferramentas de teste utilizados em projetos ágeis. A **aplicação** desta revisão sistemática é servir como base para projetos de desenvolvimento ágil que necessitam especificar, documentar e executar a atividade de testes. As **fontes** selecionadas foram bases de dados eletrônicas

indexadas (*Springer*, *ACM Digital Library*, *IEEEXplore* e *Science Direct*). Nessa fase também foram escolhidas as palavras-chaves. Para a população foram escolhidos termos relacionados a métodos ágeis e para intervenção termos relacionadas a testes. A partir desses termos foi elaborada uma *string* geral de busca que foi replicada para todas as máquinas de busca selecionadas:

> **(**(agile methodology OR agile method OR agile process OR agile software development OR extreme programming OR xp OR scrum OR crystal clear OR crystal orange OR crystal red OR crystal blue OR dsdm OR fdd OR feature driven development or lean development) AND (testing)**) OR (**test-first OR test-driven development OR automated acceptance test OR executable acceptance test OR user acceptance test OR customer acceptance test OR fitnesse**)**

Para as questões primárias e secundárias foram definidos critérios de inclusão e exclusão. Como **critérios de inclusão** para a questão primária foram considerados trabalhos que fizessem referência a prática, técnica ou critério de teste utilizado em métodos ágeis. Para as questões secundárias foram considerados trabalhos relacionados a métodos ágeis que citam aspectos que fazem com que a fase de teste seja considerada importante dentro de projetos ágeis, propõem ou citam ferramentas de teste utilizadas e por fim trabalhos que realizam estudos experimentais para avaliar a atividade de testes.

Como **critérios de exclusão** primeiramente foram excluídos trabalhos que não relacionassem a atividade de teste ou métodos ágeis, não se encontrassem redigidos em inglês ou que não tivessem a versão completa disponível. Além disso, foram excluídos trabalhos que não atendessem aos critérios de inclusão descritos anteriormente. Na **estratégia de extração de informações**, para cada artigo selecionado foram extraídas informações gerais do artigo e informações relacionadas às questões primárias e secundárias da revisão.

4.2 Busca nas Fontes

Após a fase de planejamento, parte-se para a fase de execução que consiste primeiramente na realização de estudos nas fontes escolhidas durante o planejamento. As buscas foram feitas em bases de dados eletrônicas indexadas por meio da *string* de busca geral também definida durante o planejamento.

Foram recuperados um total de 473 referências, das quais 268 artigos (56,6%) foram da IEEE Xplore, 72 da ACM Digital Library (15,22%), 119 trabalhos da SpringerLink (25,16%), 14 da ScienceDirect (2,96%). Estes trabalhos foram empacotados e os seguintes dados das buscas foram documentados em uma planilha: **(i)** *string* de busca utilizada na fonte e data que a busca foi realizada **(ii)** listagem dos trabalhos retornados e seus dados específicos (identificador, autores, título, ano, evento ou revista que foi publicado).

4.3 Seleção Preliminar

Nesta etapa foram lidos os resumos dos trabalhos recuperados da busca nas fontes. A partir desta leitura, foi elaborada uma lista dos trabalhos pré-selecionados, que levou em conta a relevância do trabalho destacado no resumo (ou, se necessário, em algumas das seções do artigo, como introdução ou conclusão). Foram selecionados 74 trabalhos, dos quais 41 artigos (56%) foram da IEEE Xplore, 9 da ACM Digital Library (12%), 18 trabalhos da SpringerLink (24%), 6 da ScienceDirect (8%).

4.4 Seleção Final e Análise dos Resultados [1]

A leitura completa dos trabalhos identificados na seleção preliminar possibilitou a identificação de 29 trabalhos relevantes para os objetivos da revisão, sendo que os critérios de inclusão e exclusão foram aplicados conforme planejado.

Na Tabela 2 é detalhada a quantidade de trabalhos incluídos, excluídos e o total de trabalhos por fonte de busca, tanto da fase de seleção preliminar, como da fase final. Além disso, na Figura 2, são apresentadas estatísticas dos trabalhos selecionados na RS. Por fim, a partir dos estudos selecionados, pode-se analisar os dados extraídos de cada trabalho e concluir a revisão sistemática. Mais detalhes sobre os estudos selecionados e outras estatísticas desta revisão sistemática podem ser consultados em [22].

Tabela 2. Quantidade de trabalhos por categoria e por fonte.

Etapa	IEEE	ACM	Springer	Science Direct	Total
Prel.	Inc.: 41 Exc.: 227 Total: 268	Inc.: 9 Exc.: 63 Total: 72	Inc.: 18 Exc.: 101 Total: 119	Inc.: 6 Exc.: 8 Total: 14	Inc.: 102 Exc.: 371 Total: 473
Final	Sel.: 16 Exc.: 25 Total: 41	Sel.: 5 Exc.: 4 Total: 9	Sel.: 5 Exc.: 13 Total: 18	Sel.: 3 Exc.: 3 Total: 6	Inc.: 29 Exc.: 73 Total: 102

A importância da atividade de testes em métodos ágeis pode ser constatada pela maioria dos métodos ágeis que consideram a atividade de testes durante todo o seu ciclo de vida. O XP por exemplo, considera que os testes são tão importantes quanto a atividade de programação [23]. A utilização de testes durante o desenvolvimento facilita a compreensão do código-fonte, apoia a documentação do software e a manutenção do código-fonte (acaba com o medo de mudanças), além de resultar em um código-fonte de melhor qualidade.

Entre os 29 estudos selecionados, 20 trabalhos (69%) discutem a respeito da estratégia TDD. A mesma vem sendo aplicada em grandes projetos (16-18) [1] e também em diversos tipos de aplicação (13), inclusive sistemas embarcados (11). A maioria dos estudos selecionados descrevem ou citam diversos estudos experimentais tanto na academia quanto na indústria.

Foram selecionados experimentos controlados e quasi-controlados, e estudos de caso na academia e na indústria (Tabela 3). Os atributos avaliados nestes estudos são descritos na Tabela 4. Esses estudos mostram diversas melhorias por meio da utilização do TDD, como o código de melhor qualidade devido à menor quantidade de defeitos e à maior cobertura de código. Além disso apontaram maior reúso, melhoria do código, menos tempo de depuração e usuários mais satisifeitos. A maioria dos estudos que tratavam sobre a produtividade utilizando TDD apontaram uma melhoria neste aspecto, contrariando os estudos descritos no trabalho (13). Foram realizados experimentos em relação a métricas de código orientados a objeto (07, 15), demonstrando em alguns casos nenhuma diferença na qualidade e pouca diferença em termos de acoplamento e coesão. Um dos estudos também apontou uma complexidade ciclomática constante du-

[1] Os trabalhos selecionados serão referenciados com uma identificação única colocada entre parênteses (ex.:"(13)"). Mais detalhes sobre os estudos selecionados podem ser consultados em [22].

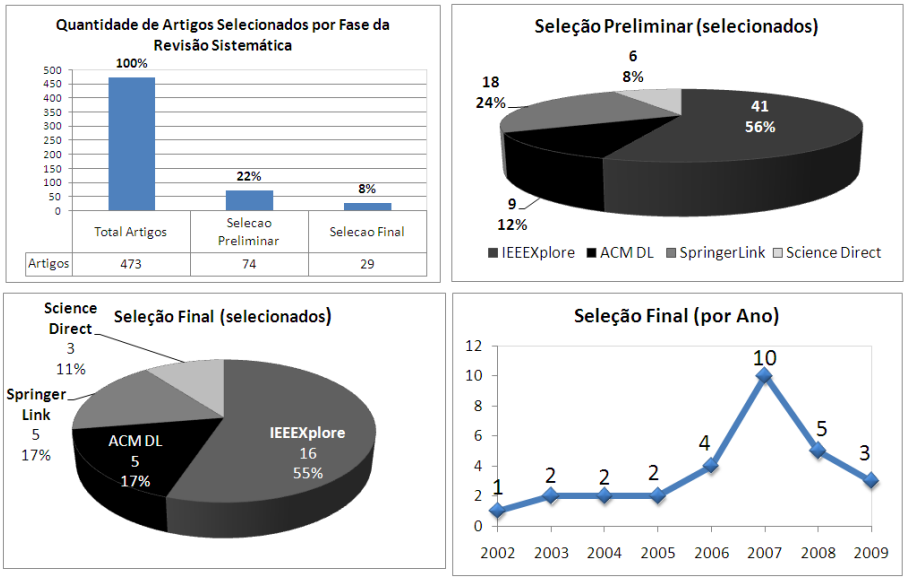

Figura 2. Estatísticas da seleção de trabalhos da revisão sistemática

rante as iterações utilizando-se TDD. Por fim, alguns estudos detectaram uma melhoria na qualidade, resultando em módulos de software menores, menos complexos e mais testados que módulos produzidos com testes tradicionais. Um dos estudos (13) faz uma análise de diversos experimentos da academia e indústria com TDD. Os resultados da academia são bastante controversos, o que se deve principalmente à inexperiência dos estudantes na utilização do TDD.

Os estudos (14, 27, 28) evidenciam a preocupação com o ensino de TDD em disciplinas iniciais de programação (10%), demonstrando bons resultados quanto à melhoria da aprendizagem dos estudantes de computação. Uma das maiores dificuldades para a utilização do TDD é a mudança do paradigma de testes tradicionais para o TDD. Nesse sentido, o treinamento de programadores para o TDD é essencial para a aplicação efetiva dessa estratégia de testes.

Os trabalhos (01,08,10,20,21,29) tratam de testes de aceitação com o cliente (21%). Esta técnica de teste, apesar de demandar um grande esforço por parte dos clientes e desenvolvedores (21), proporciona uma melhor qualidade e entendimento dos requisitos. Além disso, melhora a comunicação entre clientes e equipe de desenvolvimento, serve como indicador do progresso do desenvolvimento e melhora a identificação e correção de defeitos.

O custo para escrita e manutenção de códigos de teste é desafiador e pode ser otimizado pelo uso de ferramentas. Os testes devem ser fáceis de codificar, de ler e manter, além disso, devem ser acompanhados utilizando-se métricas. Nos estudos selecionados, foram citadas principalmente ferramentas de apoio à criação e execução de testes de unidade (como a ferramenta JUnit) e ferramentas de teste de aceitação (como o *framework* FIT e a ferramenta Selenium). Também foram detectadas algumas iniciativas de novas ferramentas de apoio a testes ágeis como a FitNesse (teste de aceitação) e o Zorro (gerenciamento do processo TDD).

Tabela 3. Tipos de Estudos Experimentais - TDD e ATDD (Seleção Final)

	Academia	**Industrial**
Exp. Controlado	(02, 05, 28)	(25)
Exp. Quasi-Controlado	(01, 08, 23)	(03, 04, 23)
Estudo de Caso	(14, 21, 22, 27)	(10, 15, 16, 17, 20, 24)
Diversos Estudos Exp.	(07, 12, 13)	(07, 12, 13)

Tabela 4. Resultados dos Estudos Experimentais - TDD (Seleção Final) [1]

	Melhoria	**Estável**	**Deterioração**
Acoplamento e Coesão		(07)	
Custos Financeiros			(26)
Cobertura	(03, 15, 23, 28)		
Complexidade Ciclomática	(16)		
Defeitos	(03, 12, 13, 16, 17, 19, 27)		
Entendimento do Programa	(04, 27)		
Maior Reúso	(04, 12)		
Mais tempo testando	(02)		
Melhoria de Código	(17, 22)		
Menos tempo Depuração	(06)		
Métricas OO		(15)	
Performance melhor alunos	(14)		
Produtividade	(02, 05, 18, 19, 24, 27)	(17)	(03, 13)
Qualidade em Geral	(07, 11, 18, 26)	(05, 25, 28)	
Usuários mais satisfeitos	(06)		

5 Considerações Finais e Trabalhos Futuros

A partir deste trabalho, procurou-se evidenciar como a atividade de teste vem sendo utilizada em projetos ágeis. Diversos estudos experimentais procuram investigar os benefícios e dificuldades para a aplicação de estratégias de teste ágil. Quanto a estratégias, técnicas ou critérios de teste utilizados, observou-se a aplicação do TDD para testes de unidade e testes de aceitação com o cliente. A atividade de teste é executada durante todo o ciclo de vida de um projeto ágil e além de validarem novas funcionalidades e mudanças de forma contínua, eles também podem guiar o projeto e documentar o software. A automação dos testes foi um ponto recorrente em diversos trabalhos, enfatizando a necessidade de apoio ferramental para que a atividade seja executada de forma eficiente.

Como contribuição, espera-se que esta revisão sistemática sirva como base para equipes de desenvolvimento e pesquisadores da área de teste de software e métodos ágeis que necessitem conhecer as particularidades relacionadas à atividade de testes executada de forma ágil. Além disso, os estudos selecionados desta revisão sistemática estão sendo utilizados para extração de termos relacionados a testes ágeis, para apoiar a criação de uma ontologia neste domínio.

Agradecimentos

Este trabalho conta com apoio financeiro do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

Referências

1. Nerur, S., Mahapatra, R., Mangalaraj, G.: Challenges of migrating to agile methodologies. Commun. ACM **48**(5) (2005) 72–78
2. Abrahamsson, P., Salo, O., Ronkainen, J., Warsta, J.: Agile software development methods - Review and analysis. Relatório técnico, VTT PUBLICATIONS (2002)
3. Abrahamsson, P., Warsta, J., Siponen, M.T., Ronkainen, J.: New directions on agile methods: a comparative analysis. In: ICSE '03: Proceedings of the 25th International Conference on Software Engineering. (2003) 244–254
4. Version One: Survey: The state of agile development. Relatório técnico (2008)
5. Huo, M., Verner, J., Zhu, L., Babar, M.A.: Software quality and agile methods. In: 28th COMPSAC Conference). (2004) 520–525
6. Janzen, D., Saiedian, H.: Does test-driven development really improve software design quality? IEEE Software **25**(2) (Mar-Abr 2008) 77–84 034.
7. Kitchenham, B.: Procedures for performing systematic reviews. Relatório técnico, Keele University and NICTA (2004)
8. Dybå, T., Kitchenham, B., Jorgensen, M.: Evidence-based software engineering for practitioners. IEEE Software **22**(1) (Jan.-Fev. 2005) 58–65
9. Mafra, S.N., Travassos, G.H.: Estudos primários e secundários apoiando a busca por evidência em engenharia de software. Relatório Técnico RT-ES 687, COPPE/UFRJ (2006)
10. Biolchini, J., Mian, P.G., Natali, A.C.C., Travassos, G.H.: Systematic review in software engineering. Relatório Técnico RT - 679/05, COPPE/UFRJ (2005)
11. Abrantes, J., Travassos, G.: Caracterização de Métodos Ágeis de Desenvolvimento de Software. In: Workshop de Desenvolvimento Rápido de Aplicações. (2007)
12. Dybå, T., Dings, T.: Empirical studies of agile software development: A systematic review. Inf. Softw. Technol. **50**(9-10) (2008) 833–859
13. Salleh, N.: A systematic review of pair programming research - initial results. In: New Zealand Computer Science Research Students Conference. (2008) 151–158
14. Simons, A.J.H.: Testing with guarantees and the failure of regression testing in extreme programming. **3556** (2005) 118–126
15. Perry, W.E.: Effective Methods for Software Testing. Wiley Publishing, Inc. (2006) 3rd Edition.
16. Beck, K.: Test Driven Development: By Example. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, EUA (2002)
17. George, B., Williams, L.A.: A structured experiment of test-driven development. Information & Software Technology **46**(5) (2004) 337–342
18. Jeffries, R., Melnik, G.: Guest editors' introduction: TDD–the art of fearless programming. IEEE Software **24**(3) (2007) 24–30
19. Melnik, G.: An Empirical Evaluation of the Impact of Test-Driven Development on Software Quality. Tese de Doutorado, University of Calgary, Canadá (2007)
20. Champion, S.: Keys to agile testing success (2008) Disponível em: http://blog.utest.com/?p=55 [18/05/2009].
21. Strode, D.E.: The agile methods: An analitical comparison of five agile methods and an investigation of their target environment. Dissertação de Mestrado, Massey University, New Zeland (2005)
22. Vicente, A.A., Delamaro, M.E.: Uma Revisão Sistemática de Teste de Software no contexto de Métodos Ágeis. Relatório Técnico (to be published) - Disponível em: http://www.labes.icmc.usp.br/avicente/revisao-testesageis/, ICMC/USP (2009)
23. Beck, K., Andres, C.: Extreme Programming Explained: Embrace Change (2nd Edition). Addison-Wesley, Boston (2004)