Test-Driven Development: Uma Revisão Sistemática

GUSTAVO BACULI BENATO*

Resumo

Test-Driven Development (TDD) é uma prática de desenvolvimento de software que ganhou notoriedade quando Kent Beck a definiu como uma parte essencial da Extreme Programming (XP). O presente estudo analisou experimentos e conclusões de estudos, previamente publicados, em relação aos efeitos do TDD na produtividade dos desenvolvedores e na qualidade do software produzido, contrastando o TDD com o Test-Last Development (TLD). Para isto, foi conduzida uma revisão bibliográfica sistemática considerando artigos publicados entre 2003 e 2020. Ao final do processo de revisão, aproximadamente 73% dos estudos analisados, consistiram em experimentos com TDD e em 27% deles, o principal tema era o TDD em sua essência, detalhando-o. A análise realizada mostra que 43% dos estudos apontaram um aumento considerável na qualidade do software, enquanto nenhum artigo apontou queda na qualidade. Em relação à produtividade, 28% dos estudos apontaram queda na produtividade e 47% foram inconclusivos. Via de regra, os estudos não apontaram melhorias significativas na produtividade quando o TDD foi utilizado. De acordo com a análise, o TDD promove maior qualidade, mesmo que alguns estudos apontem o contrário. Em relação à produtividade, o TDD é inconclusivo. Sendo assim, de acordo com os artigos analisados, não há uma posição final referente ao custo-benefício envolvido nesta prática, discutimos algumas possíveis razões para essa conclusão.

1 INTRODUÇÃO

Test-Driven Development (TDD) é uma prática de desenvolvimento de software onde testes unitários automatizados são escritos de forma incremental antes mesmo do código de produção ser desenvolvido (BECK, 2003). O TDD ganhou popularidade quando foi definido por Kent Beck como uma parte essencial da *Extreme Programming* (XP) (BECK, 2010), que é uma metodologia ágil de desenvolvimento de software focada na aplicação de técnicas de programação, comunicação clara e trabalho em equipe.

*Tecnólogo em Análise e Desenvolvimento de Sistemas pela Faculdade de Tecnologia - Universidade Estadual de Campinas. E-mail para contato: g198462@dac.unicamp.br.

Essa abordagem, também chamada de *Test-First*, traz uma visão completamente oposta à abordagem tradicional (*Test-Last*), onde o código de produção é escrito de acordo com as especificações do projeto e, normalmente, somente depois de grande parte do código de produção ser escrito, os casos de teste começam a ser desenvolvidos (HAMMOND; UMPHRESS, 2012). A abordagem tradicional será referenciada aqui como *Test-Last Development* - TLD.

Ao buscar entender os resultados que o TDD proporciona, o que encontra-se são análises e resultados divergentes em relação ao seu custo-benefício, sendo que em muitos casos são considerados inconclusivos, mesmo que sejam executados experimentos em ambientes diversos.

Portanto, esta revisão tem o objetivo de entender o atual cenário do TDD, considerando ambientes industriais e acadêmicos, com enfoque em seu custo-benefício, ou seja, busca-se analisar a qualidade do software desenvolvido com TDD e a produtividade de equipes que fizeram uso do TDD, para, assim, obter um comparativo entre o custo-benefício do desenvolvimento de software que faz uso do TDD e do TLD.

Neste estudo, apresenta-se uma revisão bibliográfica sistemática de artigos e estudos empíricos que foram publicados de 2003 a 2020. Adotou-se o ano de 2003 pois neste ano Beck publicou seu primeiro livro sobre o TDD (BECK, 2003), portanto este ano é o marco zero do TDD. Esta revisão se limita a agregar descobertas empíricas em duas construções de resultados. A primeira é a qualidade externa do código e a segunda, é a relação entre custo-benefício do TDD quando comparado com o *Test-Last Development* (TLD).

O processo de revisão sistemática utilizado neste estudo identificou inicialmente 281 artigos, sendo que 19 deles foram selecionados e discutidos em detalhes. Estes 19 artigos foram selecionados pois têm o TDD como seu foco de pesquisa e abordam um comparativo entre TDD e TLD, na maioria dos casos. Os demais artigos foram rejeitados porque não se encaixavam nos critérios de aceite ou satisfaziam algumas das condições de exclusão.

A maioria dos estudos selecionados apontam um aumento na qualidade do código e testes unitários quando o TDD foi utilizado. Porém, são inconclusivos, no que diz respeito à produtividade dos programadores durante o processo de desenvolvimento de software.

2 METODOLOGIA

O método de pesquisa utilizado neste estudo foi uma revisão bibliográfica sistemática (RBS), que consiste em um estudo empírico no qual uma hipótese ou pergunta de pesquisa é abordada para agrupar indicadores de estudos primários por meio de um processo sistemático de pesquisa e extração de dados.

2.1. Processo de pesquisa

O processo de pesquisa teve seu início a partir da definição do protocolo de pesquisa. Toda a pesquisa foi estruturada e documentada no software Parsifal (PARSIFAL. . . , s.d.), uma ferramenta

online projetada para auxiliar a realização de revisões sistemáticas da literatura no contexto da engenharia de software.

Após a definição do protocolo, foram elaboradas as perguntas de pesquisa, as quais exercem papel fundamental em uma revisão sistemática, visto que as mesmas guiam todo o processo. Abaixo estão as perguntas de pesquisa do presente estudo:

P1 - Os custos envolvidos na aplicação do *Test-Driven Development* compensam seus benefícios?

P2 - Quando o *Test-Driven Development* for aplicado, a quantidade de defeitos encontrados no software nas fases mais avançadas do desenvolvimento será menor?

P3 - Quando o *Test-Driven Development* for aplicado, muitos problemas serão detectados e removidos ainda nas fases iniciais, muito provavelmente, pelos próprios desenvolvedores?

Logo após a elaboração das perguntas de pesquisa, os estudos primários foram filtrados em repositórios online. Após a busca nos repositórios digitais, os artigos foram selecionados e classificados nas seguintes categorias

1. Experimento sobre TDD: aqui se encaixam artigos que relatam experimentos conduzidos e analisados pelo(s) próprio(s) autor(es);
2. Experimento de terceiros e experimento próprio: engloba artigos que analisam experimentos de outros autores e, também, descrevem experimento(s) do(s) próprio(s) autor(es);
3. Revisão Bibliográfica Sistemática sobre TDD: consiste em estudos que são revisões sistemáticas, gerando grande embasamento para o presente estudo;
4. *Test-Driven Development*: estudos que são descritivos e detalham a prática do TDD durante o processo de desenvolvimento de software.

3 ANÁLISE DOS RESULTADOS

Para cada estudo foram analisados dois pontos principais, que serviram como base para responder às perguntas de pesquisa: a qualidade do software e a produtividade dos desenvolvedores.

3.1. Qualidade

Em relação à variável qualidade, nenhum estudo apontou que o TDD diminua a qualidade do software quando comparado com o desenvolvimento TLD.

Entre os estudos analisados, 42,86% apontaram um aumento na qualidade (seja ela interna ou externa); 28,57% não apontaram nenhuma melhoria na qualidade; e 28,57% são inconclusivos

em relação à variável qualidade. Com o uso do TDD, a preocupação com a escrita de testes se torna maior, visto que é requerido a criação de testes unitários que desenvolvem gradualmente pequenas partes da funcionalidade até que um recurso seja totalmente implementado (VU et al., 2009). A maioria dos estudos selecionados, apontam que o TDD aumenta a qualidade de modo geral, conforme previa Kent Beck quando apresentou a metodologia XP, que faz uso do TDD (BECK, 2010).

Pode-se observar, em (GEORGE; WILLIAMS, 2003) os autores mostram que o desenvolvimento de software com TDD, aparentemente, mesmo considerando as limitações do estudo, produz código com qualidade superior quando comparado com o código desenvolvido com uma prática mais tradicional, por exemplo, o TLD, utilizando um conjunto de casos de teste de caixa preta.

3.2. Produtividade

Analisando os estudos selecionados, levando em consideração a variável produtividade, pode-se notar que o TDD, de modo geral, não apresenta aumento na produtividade dos desenvolvedores de software, assim como é retratado no experimento de (VU et al., 2009) onde o grupo *Test-Last*, segundo os autores do experimento, parece ser mais produtivo que o *Test-First* (TDD).

Nesta revisão, somente um estudo apontou que quando o TDD foi aplicado a produtividade aumenta (GUPTA; JALOTE, 2007), enquanto que 21,43% foram classificados como "Sem Aumento". Seguindo a análise, 28,57% dos estudos apontaram que quando o TDD foi utilizado a produtividade diminui e, por fim, 42,86% dos estudos são inconclusivos.

4 CONCLUSÃO

Esta revisão bibliográfica sistemática foi realizada com o objetivo de identificar, selecionar e analisar estudos empíricos que abordam o uso da prática TDD no desenvolvimento de software, sempre tendo como foco avaliar os efeitos originados na qualidade do software desenvolvido e na produtividade dos desenvolvedores.

A maior parte dos estudos presentes nesta revisão (42,86%) indicam uma melhoria na qualidade do software quando o TDD foi utilizado, sendo que nenhum estudo apontou uma diminuição na qualidade.

É importante ressaltar que os resultados da análise desses artigos mostraram que, em quase metade deles, não houve um resultado claro sobre a relação entre custo e benefício do TDD, sendo assim, seus resultados foram inconclusivos. Apenas 7% dos estudos apontaram o uso do TDD como positivo de modo geral, ou seja, neste estudo a qualidade aumentou e a produtividade também, não havendo necessidade de empenhar mais tempo durante o desenvolvimento do experimento. Estes mesmos 7%, representam, também, os estudos que descrevem o TDD

como uma prática negativa, ou seja, não aumenta a qualidade e eleva o tempo gasto durante o desenvolvimento do software.

Esta revisão buscou entender a relação de custo-benefício do TDD, mas não chegou a ser conclusiva, sendo assim, este estudo não pode afirmar ou negar que o uso da prática realmente traz melhorias compensando seus custos. Todavia, pode-se afirmar que o TDD traz mais qualidade para o software e, principalmente, para os testes unitários que são desenvolvidos pelo próprio programador.

Por fim, os resultados finais dessa pesquisa foram submetidos para publicação. O artigo "Avaliando o Custo-Benefício do Test-Driven Development:Uma Revisão Bibliográfica Sistemática" foi enviado para a Revista Brasileira de Computação Aplicada (RBCA).

REFERÊNCIAS

BECK, K. **Extreme Programming Explained: Embrace Change**. [S.l.]: Addison-Wesley, 2010.

BECK, K. **Test-Driven Development: By Example**. [S.l.]: Addison-Wesley, 2003.

GEORGE, B.; WILLIAMS, L. An Initial Investigation of Test Driven Development in Industry. In: PROCEEDINGS of the 2003 ACM Symposium on Applied Computing. Melbourne, Florida: Association for Computing Machinery, 2003. (SAC '03), p. 1135–1139. ISBN 1581136242. DOI: 10.1145/952532.952753. Disponível em: <https://doi.org/10.1145/952532.952753>.

GUPTA, A.; JALOTE, P. An Experimental Evaluation of the Effectiveness and Efficiency of the Test Driven Development. In: FIRST International Symposium on Empirical Software Engineering and Measurement (ESEM 2007). [S.l.: s.n.], 2007. p. 285–294.

HAMMOND, S.; UMPHRESS, D. Test Driven Development: The State of the Practice. In: PROCEEDINGS of the 50th Annual Southeast Regional Conference. Tuscaloosa, Alabama: Association for Computing Machinery, 2012. (ACM-SE '12), p. 158–163. ISBN 9781450312035. DOI: 10.1145/2184512.2184550. Disponível em: <https://doi.org/10.1145/2184512.2184550>.

PARSIFAL. [S.l.: s.n.]. https://parsif.al/.

VU, J. H. et al. Evaluating Test-Driven Development in an Industry-Sponsored Capstone Project. In: 2009 Sixth International Conference on Information Technology: New Generations. [S.l.: s.n.], 2009. p. 229–234.