Teste de Software: o que e como é ensinado?

Fabiane Barreto Vavassori Benitti[1], Edson Lucas Albano[1,2]

[1]Mestrado em Computação Aplicada – Universidade do Vale do Itajaí (UNIVALI)
Caixa Postal 360 – 88302-202 – Itajaí – SC – Brazil
[1,2]Universidade Paranaense (UNIPAR)
Av. Júlio Assis Cavalheiro, 2000 - 85601-000 – Francisco Beltrão – RS – Brazil

fabiane@univali.br, lucas.albano@unipar.br

Abstract. *The quality of software is increasingly demanded by the market and, in this context, the activities related to software testing stand out. Thus, this paper explores what should be taught about software testing, and identifies teaching practices that have shown good results. This paper initially presents a documentary research, involving curriculum guidelines and teaching plans, to point out what should be taught in the area. Subsequently, we present a systematic mapping involving resources and techniques for teaching software testing. The results show a list of concepts, 5 approaches to teaching and 4 computational resources.*

Resumo. *A qualidade do software está cada vez mais sendo exigida pelo mercado e, neste contexto, ganham destaque as atividades relacionadas ao teste de software. Assim, este artigo explora o que deve ser ensinado sobre teste de software, bem como identifica práticas de ensino que tenham apresentado bons resultados. Para tanto, inicialmente é apresentada uma pesquisa documental, envolvendo diretrizes curriculares e planos de ensino, para apontar o que deve ser ensinado na área. Posteriormente, é apresentado um mapeamento sistemático envolvendo recursos e técnicas para o ensino de teste de software. Os resultados apontam uma lista de conceitos, bem como apresentam 5 abordagens de ensino e 4 recursos computacionais.*

1. Introdução

Inicialmente, a atividade de teste era encarada simplesmente como a tarefa de navegar pelo código e corrigir problemas já conhecidos. Tais tarefas eram realizadas pelos próprios desenvolvedores, não existindo recursos dedicados exclusivamente a essa atividade, de maneira que os testes eram realizados tardiamente. Apesar de essa situação estar associada a uma má prática de desenvolvimento, ela continua presente em muitas organizações (BARTIÉ, 2002).

Para que as equipes desenvolvedoras melhorem estas práticas, tendo em vista a importância do teste na garantia de qualidade do software, existe a necessidade das universidades enfatizarem o ensino do teste de software aos alunos dos cursos de computação. Apesar de muitas pessoas entenderem que uma introdução breve aos testes na ementa de engenharia de software seria suficiente, esta abordagem é inadequada (CHEN, POON; 2004). Os esforços alocados para o ensino de testes é

desproporcionalmente inferior ao esforço e recursos que precisam ser empregados para uma eficiente atividade de testes em projetos de software (SHEPARD, LAMB, KELLY; 2001). Para Jones e Chatmon (2001), Chen e Poon (2004) e Elbaum et Al. (2007), os currículos dos cursos de Ciência da Computação destinam um nível muito baixo de atenção aos testes de software. Beizer (1990) destaca que embora os testes consumam mais da metade da vida profissional de um programador, menos do que 5% da educação de um programador são dedicados a atividade de teste.

A falta de atividades práticas também é um problema identificado no ensino de teste de software nos cursos de graduação. Como na programação, o teste exige experiência prática para complementar os princípios do ensino (CARRINGTON, 1997), além de desenvolver atitudes e habilidades necessárias para a sua aplicação na vida profissional (JONES, CHATMON, 2001). As capacitações tradicionais geralmente não enfatizam o exercício prático de elaboração de casos de testes num contexto variado, e nem a execução dos casos de testes, algo necessário para que os profissionais, efetivamente, aprendam a usar estes conceitos em situações reais (KANER, PADMANABHAN, 2007).

A pouca importância aos testes extrapola as salas de aula e se reflete diretamente nas empresas de desenvolvimento de software, que consideram os testes como uma atividade secundária. Essa visão errônea, aliada ao fato de que os testes consomem tempo e recursos para sua execução, fazem com que os testes sejam suprimidos, principalmente quando se visa cumprir um prazo de entrega mal planejado. Porém, softwares mal testados provocam prejuízos enormes às organizações (BARTIÉ, 2002).

Pesquisas apontam a existência de uma grande diferença entre o estado da arte e o estado da prática de teste de software, havendo poucas evidências relatadas da efetividade das práticas de teste de software sugeridas pela literatura técnica nas empresas de software (DIAS NETO et al, 2006; BERTOLINO, 2004). Na pesquisa realizada por Dias Neto et al (2006) foi evidenciado que, embora as organizações admitam que aplicam testes de software, quase metade (48,5%) das práticas de teste de software são consideradas não-aplicadas e não-importantes. Dentre as razões apontadas para os resultados obtidos, destaca-se a falta de conhecimento e a falta de recursos humanos para a implantação de técnicas e práticas de testes.

Estes problemas referentes ao ensino de teste de software e a necessidade de ampliar a adoção de boas práticas de teste em empresas de desenvolvimento de software, motivam a investigação / discussão de métodos eficazes de ensino de teste que promovam a aprendizagem dos alunos nas habilidades de teste de software. Assim, este artigo apresenta uma pesquisa documental (seção 2) para identificar “o que” ensinar e, posteriormente, na seção 3, é descrito um mapeamento sistemático visando apontar “como” tem sido ensinado o assunto teste de software.

2. Pesquisa documental

A pesquisa documental assemelha-se muito com a pesquisa bibliográfica, diferenciando-se apenas na natureza das fontes utilizadas. Para Gil (1999) a pesquisa documental é elaborada a partir de materiais que não receberam tratamento analítico, ou seja, caracteriza-se pela busca de informações em documentos que não receberam nenhum tratamento científico (como por exemplo, relatórios, reportagens, revistas, cartas, etc); já

a pesquisa bibliográfica envolve material já publicado, constituído principalmente de livros e artigos de periódicos, ou seja, vale-se de documentos de domínio científico.

2.1 Diretrizes curriculares

Para determinar quais assuntos abordar referente a teste de software nos cursos superiores relacionados à informática, esta pesquisa documental envolveu currículos de referência nacionais (CEEinf, 1999; SBC, 2003; SBC, 2005) e internacionais (IEEE, 2004; ACM; AIS; IEEE-CS, 2006).

As diretrizes do CEEInf (1999) propõem uma estrutura curricular abstrata que divide as disciplinas em quatro grandes áreas: formação básica, tecnológica, completar e humanística. O teste de software é sugerido em duas disciplinas: Engenharia de Software e Interface Homem-Máquina, as quais se encontram dentro da área de formação tecnológica. Na disciplina de engenharia de software o teste é citado como um tópico que deve ser abordado, mas o documento não apresenta os conteúdos, deixando em aberto esta questão. Porém, define que o estudo da engenharia de software pode dar origem a várias disciplinas com diferentes ênfases.

O currículo de referência da SBC (2005) para cursos de graduação em informática, tem uma estrutura semelhante ao CEEInf (1999) e, assim como o SBC (2003), faz referência a testes de software nas disciplinas de engenharia de software, gestão de projetos e gestão da qualidade de software, citando o teste como um tópico a ser abordado dentro de cada disciplina, mas deixando em aberto os conteúdos e nível de profundidade.

ACM, AIS e IEEE-CS (2006) fazem uma proposta curricular para cinco cursos superiores para os quais o teste de software é abordado em uma disciplina de Verificação e Validação de Software. O documento apresenta um conceito para as atividades de V&V, mas não detalha os conteúdos, deixando este aspecto em aberto. Porém, classifica a importância das atividades de V&V para cada um dos cinco cursos usando uma métrica de 0 (zero) para mínimo e 5 (cinco) para máximo. Os valores expostos foram os seguintes: i) Ciência da Computação: 1 a 2; ii) Sistemas de Informação: 1 a 2; iii) Engenharia da Computação: 1 a 3; iv) Tecnologia da Informação: 1 a 2; e v) Engenharia de Software: 4 a 5.

Já o IEEE (2004) define uma estrutura de conteúdos para um curso de Engenharia de Software, especificando, além de uma estrutura de tópicos, os conceitos básicos de cada um, além disso, também apresenta a distribuição dos tópicos sugeridos dentro da taxonomia de Bloom, conforme Tabela 1.

Tabela 1 – Tópicos para ensino do teste de software

Divisão de Tópicos	Conteúdo	Nível de Taxonomia (*)
Fundamentos de Teste de Software	Terminologia relacionada à testes	C
	Questões chave	AP
	Relacionamento do Teste com outras atividades	C
Níveis de teste	Onde se aplicam os Testes	AP
	Objetivos do Teste	AP
Técnicas de Teste	Testes baseados na intuição e experiência	AP
	Baseados nas especificações	AP
	Baseados no código	AP
	Baseados em falhas	AP
	Baseados na natureza da aplicação	AP
	Baseados no uso	AP

	Seleção em combinação de técnicas	AP
Métricas relacionadas a teste	Avaliação do programa em teste	AN
	Avaliação do teste realizado	AN
Processo de teste	Questões quanto ao gerenciamento	C
	Atividades de teste	AP

*Taxonomia de Bloom - C: Compreender, AP: Aplicar e AN: Analisar
Fonte: Adaptado de IEEE, 2004.

Observa-se que as diretrizes curriculares nacionais, de forma geral, apenas apontam o teste de software como um componente curricular, mas deixam a cargo das instituições determinarem os tópicos e profundidade dos conteúdos com base na literatura e na contextualização do curso. Já as diretrizes internacionais apresentam uma abordagem mais detalhada para o tema, sugerindo assuntos e utilizando a taxonomia de Bloom para determinar a profundidade dos conteúdos.

Tendo em vista que as diretrizes nacionais não detalham os conteúdos sobre teste de software, o próximo tópico analisa alguns planos de ensino de instituições brasileiras, visando identificar o conteúdo ensinado relacionado a teste de software.

2.2 Planos de ensino envolvendo teste de software

Esta pesquisa documental considera os planos de ensino[1] das disciplinas de cursos de graduação voltados à informática, que apresentem alguma referência ao teste de software. O objetivo é conhecer os conteúdos sobre teste de software que estão sendo ensinados nestes cursos.

Como o número de cursos é bastante grande, optou-se por observar os planos de ensino de instituições que, além de cursos de graduação relacionados, também possuam pós-graduação stricto-sensu na área, apresentando conceito mínimo 5 pela CAPES, conforme avaliação do triênio 2007-2009 (CAPES, 2010). As instituições filtradas pelo critério de corte, somam um total de 9 instituições, as quais, possuem juntas 18 cursos de graduação voltados para área da informática.

Todos os cursos analisados apresentam o teste de software em alguma das disciplinas curriculares. Ao todo foram encontradas 28 disciplinas que abrangem teste de software, destas, apenas 11 disciplinas específicas, ou seja, voltadas exclusivamente para o teste de software. As 17 restantes são disciplinas voltadas à engenharia de software ou análise e projeto de sistemas, que possuem o teste de software como um tópico de seu conteúdo - o que atende às diretrizes nacionais.

O fato de serem encontradas 11 disciplinas específicas para teste de software entre as 28 que abordam o tema, pode ser um indicador de que os cursos superiores analisados estão percebendo a necessidade de tratar de forma mais aprofundada o ensino de testes. No caso, dos 18 cursos envolvidos na pesquisa, apenas 7 não possuem uma disciplina específica sobre testes de software. Os conteúdos encontrados nos planos de ensino dessas disciplinas são apresentados de forma sintetizada na Tabela 2, agrupando os conteúdos obtidos dos planos de ensino em cinco grupos: fundamentos, níveis, técnicas, tipos e processo de teste (grupos adaptados do Swebok (IEEE, 2004)).

[1] Os planos de ensino foram obtidos durante o ano letivo de 2010 nos sites das Universidades ou mediante contato com a coordenação do curso.

Tabela 2 – Síntese dos conteúdos sobre teste de software presentes nos planos de ensino

Universidades*	Fundamentos	Níveis	Técnicas	Tipos	Processo
PUC-RIO	Princípios de teste de programas. Verificação e validação.	Teste de unidade, teste integração. Testes de aceitação, alfa e beta.	Técnicas de teste: Caixa branca e caixa preta. Geração de casos de teste. Verificação e validação em modelos UML. Inspeção de software.	Teste de interface humanas. Teste de aplicações Web. Teste regressão.	Desenvolvimento orientado a testes. Planos de teste. Gerenciamento do processo de teste. Registro e acompanhamento de problemas. Ferramentas e Automação dos testes.
UFMG	Princípios: objetivos e métodos.	Teste de integração e aceitação.		Teste de regressão	Desenho de testes. Realização.
UFPE	Motivação.		Técnicas de teste e validação de sistemas		Ferramentas de teste e validação de sistemas
UNICAMP	Conceito V & V. Testes: objetivos, fases e terminologia.	Fases de teste: teste de unidade (*drivers* e *stubs*), de integração, de validação e de sistemas. Escopo de testes. V&V nos modelos de ciclo de vida.	Técnicas de teste: noção de critério e cobertura. Teste caixa branca. Teste caixa preta. Testes baseados em modelos da UML. Teste baseado em código. Revisões Técnicas. Análise Estática.	Teste de interface. Teste de requisitos de qualidade. Teste de regressão. Teste OO.	Processo de teste. Documentação dos testes. Automação de testes. Aspectos de gerenciamento de testes. Atividades da garantia de qualidade de software (SQA).
USP e USP/São Carlos	Terminologia e conceitos básicos de teste.	Teste no ciclo de vida: fases de teste.	Técnicas de teste de software: teste funcional, estrutural e baseado em erros. Estratégias de Análise estática - inspeções, walkthroughs.	Teste de regressão. Teste OO.	Estratégias de teste de software. Planejamento de teste. Ferramentas de teste de software. Ferramentas para análise estática.

* Incluídos apenas os planos das universidades que apresentaram conteúdo detalhado. Nesta síntese foi agrupado o conteúdo de diferentes disciplinas ministradas em uma mesma universidade. Em algumas universidades "testes" era apenas um tópico da ementa, sem acesso a detalhamento e, portanto, não contemplado nesta tabela.

Esta síntese de conteúdos (Tabela 2), embora com o foco restrito, pode colaborar na definição de conteúdos a serem abordados na área de teste de software.

3. Mapeamento Sistemático

Segundo Kitchenham e Charters (2007) uma revisão sistemática deve ser realizada seguindo as etapas de planejamento da revisão, condução da revisão e documentação da revisão, podendo estes passos ser observados também para um mapeamento sistemático. Assim, as seções seguintes detalham cada etapa realizada.

3.1 Planejamento

O planejamento da revisão sistemática deve primeiramente justificar sua realização, definindo a originalidade do trabalho. Em seguida define-se o protocolo de revisão, documento onde as questões de pesquisa são detalhadas (KITCHENHAM; CHARTERS 2007). Assim, por não se ter encontrado nenhuma revisão sistemática envolvendo

estudos referentes ao ensino de teste de software, ressalta-se a necessidade de realizar a pesquisa, definindo-se o protocolo da revisão sistemática conforme Tabela 3.

Tabela 3 – Protocolo de revisão adotado neste trabalho

Objetivo	Identificar recursos e técnicas para o ensino de teste de software.
Pergunta de pesquisa	De que maneira o teste de software está sendo ensinado em cursos superiores e empresariais? Os limites da revisão são estabelecidos identificando qual é o escopo considerado para que a revisão seja focada em responder as perguntas de pesquisa. Além disso, foi adotado um conjunto de critérios denominado PICO - População, Intervenção, Comparação e Resultados (KITCHENHAM; CHARTERS, 2007). Desta forma, o escopo para a revisão foi estabelecido conforme segue: **População** – Cursos superiores e empresariais envolvendo teste de software. **Intervenção** – Uso de recursos e técnicas inovadoras para ensino de teste de software. **Comparação** - Métodos tradicionais de ensino de teste de software. **Resultados** - Melhorias obtidas no ensino/aprendizagem
Estratégia	**String de busca:** abstract:((learn OR learning OR teach OR teaching OR education) AND ("testing software" OR "software test")) **Fontes de pesquisa (Bases de dados):** ACM Digital Library, CiteSeerX, GoogleScholar, IEEE Explore, ScienceDirect, SpringerLink.
Critérios de Seleção	•Os termos de busca devem estar contidos no título ou no abstract •Serão considerados trabalhos na língua inglesa •Serão considerados apenas trabalhos que passaram por revisão por pares
Critérios de Exclusão	•(CE1) O estudo não está relacionado com um método de ensino de teste de software •(CE2) O estudo não é direcionado a graduação ou treinamento profissional •(CE3.a) O estudo não apresenta uma avaliação do método ou (CE3.b) a avaliação não está detalhada no estudo.
Estratégia para extração dos dados	1.Leitura do título e abstract para verificar a adequação aos critérios de seleção e exclusão; 2.Caso o estudo fosse incluído no passo 1, foi realizada a leitura do artigo completo (novamente confrontando com os critérios de inclusão e exclusão); 3.Caso o estudo atendesse aos critérios, foi realizado o preenchimento de uma planilha contemplando os principais pontos de interesse.

3.2 Execução

A execução consiste na busca e seleção dos estudos, conforme o protocolo definido, bem como avaliação da qualidade destes estudos, extração e síntese dos dados (KITCHENHAM; CHARTERS, 2007). Como pode ser observado na Tabela 4, a busca nas bases de dados retornaram 103 estudos. Desconsiderando duplicidades e links quebrados, restaram 100. A tabela também informa o total de estudos considerados na pesquisa e os critérios de exclusão aplicados (conforme os itens listados na Tabela 1).

Tabela 4 – Quantidade artigos por base de dados

Base de dados	String formatada	Total de estudos	CE1	CE3a	CE3b	Estudos incluídos
ACM Digital Library	((Abstract:learn OR Abstract:learning OR Abstract:teach OR Abstract:teaching OR Abstract:education) AND (Abstract:"testing software" OR Abstract:"software test"))	5	3	2	0	0
CiteSeerX	abstract:((learn OR learning OR teach OR teaching OR education) AND ("testing software" OR "software test"))	47	31	10	2	4
GoogleScholar	(notítulo:learn OR notítulo:learning OR notítulo:teach OR notítulo:teaching OR notítulo:education) (notítulo:"testing software" OR notítulo:"software test")	35	18	8	2	4

IEEE Explore	("Abstract":(learn OR learning OR teach OR teaching OR education) AND "Abstract":("testing software" OR "software test"))	11	8	2	0	1
ScienceDirect	tak((learn OR learning OR teach OR teaching OR education) AND ("testing software" OR "software test")) [All Sources(Computer Science)]	3	2	1	0	0
SpringerLink	'ab:((learn or learning or teach or teaching or education) and ("testing software" or "software test"))'	2	2	0	0	0
	Total:	103	64	23	4	9

* GoogleScholar: 1 artigo foi ignorado por estar duplicado e 2 não forneceram download
* Não houve excluídos por CE2

A grande maioria dos estudos não tratava de um método de ensino de testes (CE1). Estes trabalhos envolviam teste de software e neles constavam os termos de pesquisa como "ensino" ou "aprendizagem", mas em contextos diferentes do que se procurava, como por exemplo, teste de ambientes virtuais de ensino-aprendizagem.

Dos estudos mais diretamente ligados ao tema foram encontrados ao todo 36 artigos. Porém, destes, 23 não apresentavam uma avaliação do método proposto (CE3a) ou essa avaliação não era detalhada (CE3b). Restando assim os 9 estudos escolhidos como resultado deste mapeamento.

3.3 Análise e Resultados

Conforme Kitchenham e Charters (2007), a síntese dados envolve resumir os resultados dos estudos incluídos, podendo ser descritiva e, quando possível, também quantitativo (envolvendo análise estatística). Neste mapeamento, o pequeno número de estudos obtidos impossibilita uma análise estatística mais precisa. Albano (2012) apresenta um resumo descritivo dos estudos incluídos.

Para entender as diversas propostas para ensino de teste de software apresentadas pelos artigos selecionados, os mesmos foram inicialmente classificados conforme o método de ensino proposto. A síntese destes estudos apresentou três métodos de ensino distintos: Módulos Educacionais (2 estudos), Tutorial Baseado na Web (2 estudos) e Abordagem de ensino (5 estudos).

Os cinco artigos classificados como abordagem de ensino ((Chen; Poon, 2004), (Hilburn; Towhidnejad, 2000), (Janzen; Saiedian, 2008), (Mao, 2008), (Wang; et al., 2010)) são estudos que propõem uma metodologia para o ensino de teste. Alguns desses destacam-se por trazer para sala de aula recursos existentes no processo de desenvolvimento de software visando apoiar o ensino. Um exemplo é o processo TSP (*Team Software Process*) baseado no modelo de qualidade em V (HILBURN; TOWHIDNEJAD, 2000), outro propõe uma abordagem chamada TDL (*Test-Driven Learning*) que sugere usar TDD (*Test-Driven Development*) para ensinar testes de unidade (JANZEN; SAIEDIAN, 2008).

Dois estudos ((Elbaum et al, 2007), (Kaner; Padmanabhan, 2007)) envolvem algum tipo de ferramenta de ensino desenvolvida em forma de aplicação para internet. Essas propostas foram classificadas como Tutoriais Baseados na Web. Os Módulos de Ensino, por sua vez, também poderiam ser considerados aplicações web, mas, por terem uma estrutura padronizada e definida, optou-se por usar essa nomenclatura. Em um dos artigos os módulos de ensino são aplicados para ensinar teste de mutação (BARBOSA; MALDONADO, 2006), mas o artigo deixa claro que os módulos de ensino podem ser

construídos para outros tópicos, como ocorre no segundo artigo, em que o método é aplicado para ensino de conceitos de validação e verificação (BARBOSA; SOUZA; MALDONADO, 2008).

Como resultado da análise dos métodos de ensino, observa-se que a utilização de ferramentas de apoio ao ensino de testes, como é o caso dos módulos educacionais e aplicações web, constituem grande parte das propostas encontradas nos artigos selecionados, demonstrando que esses recursos computacionais contribuem para o processo de ensino e, por isso, devem ser considerados. Mesmo os trabalhos classificados como abordagens de ensino, consideram o uso de recursos computacionais para disponibilização de materiais, como por exemplo, o estudo apresentado por Kaner, e Padmanabhan (2007). Eles elaboram um conjunto de slides, palestras, exemplos, exercícios e exames, fazendo uso da internet para disponibilizar os materiais.

O mapeamento mostrou que poucos trabalhos na área preocuparam-se em fazer uma avaliação do método ou técnica de ensino proposta, apenas 9 em 36. Nestes, as formas de avaliação foram bastante diferenciadas. Em um caso foi usado grupo de controle e experimental para avaliação de conhecimentos, mas não considerou os conhecimentos anteriores dos participantes. Três artigos avaliaram apenas o sucesso dos participantes em realizar atividades com a técnica proposta, mas sem comparação com um processo padrão de ensino. Outros dois estudos consideraram apenas a opinião dos participantes quanto ao uso do método de ensino obtida com um questionário de opinião. Apenas três trabalhos fizeram avaliação com pré e pós-teste, aplicados a um grupo experimental e outro de controle, além do questionário de opinião.

Quanto aos resultados apresentados pelas avaliações aplicadas aos estudos, de maneira geral, todos avaliaram positivamente suas propostas. No entanto, 4 dos 9 estudos deixaram claro que os resultados obtidos são preliminares, e que seriam necessários experimentos mais sistematizados, que avaliassem de forma quantitativa, para um resultado mais preciso. O estudo de Kaner e Padmanabhan (2007) foi o único que apresentou um ponto negativo: observou que seu método de ensino se mostrou eficiente para assimilação dos conceitos, mas os alunos demonstraram dificuldades em aplicar os conhecimentos em situações novas.

4. Conclusões

A partir da pesquisa documental realizada, considerando principalmente a diretriz da IEEE (2004) e os planos de ensino, pode-se responder a pergunta “o que ensinar sobre teste de software?”:

- Fundamentos: Motivação para estudar a área (importância). Envolver conceitos como objetivos, princípios e terminologias de teste de software.
- Níveis de teste: V&V nas fases de ciclo de vida. Abordar, por exemplo, teste de unidade, integração e aceitação (alfa e beta).
- Técnicas de testes: Envolver técnicas de caixa preta e caixa branca, bem como geração de casos de teste, testes baseados em modelos UML, testes baseados em código e testes baseados em erros. Análise estática - inspeções, walkthroughs.
- Tipos: Focar em teste de interface, teste de aplicações Web, teste de requisitos de qualidade, teste de regressão e teste OO.

- Processo de teste: abordar atividades de um processo de teste (envolvendo gerenciamento, planejamento e execução), artefatos e ferramentas de automação de teste, bem como atividades da garantia de qualidade de software (SQA). Métricas relacionadas a teste é apontado pelo IEEE (2004).

Com base nas análises apresentadas após a execução do mapeamento sistemático, busca-se responder a pergunta "Como é ensinado teste de software?" Concluiu-se que os métodos de ensino de teste de software adotados atualmente exploram ferramentas de apoio ao ensino baseadas na web e propostas para definir processos de ensino mais específicos e eficazes. Também fica evidente a quantidade restrita de estudos nesta área, bem como a escassez de avaliações empíricas. Contudo, se deve considerar que os estudos apresentados são resultantes de uma *string* de busca específica, em um conjunto de 6 fontes de dados, limitado a busca por termos em inglês. A ampliação dos termos, línguas e fontes de dados pode revelar novos recursos.

A pesquisa nos planos de ensino aponta que o ensino de teste de software, como um tópico na disciplina de engenharia de software, vem sendo considerado insuficiente pela maioria das universidades consideradas (visto que as mesmas já possuem disciplinas específicas). Outro ponto percebido, e que merece destaque, é a maior ênfase destinada ao ensino de validação, pouco se encontrou sobre verificação.

Como pesquisa futura, sugere-se uma comparação entre o ensino de teste nas Universidades e as necessidades apontadas pelo mercado. Além disso, ampliar as fontes de pesquisa (planos de ensino de outras universidades e outras bases de dados) também pode contribuir para resultados mais aprofundados.

Referências

ACM; AIS; IEEE-CS. (2006) "Computing Curricula 2005." Disponível em: <www.acm.org/education/curric_vols/CC2005-March06Final.pdf>. Acesso em: 05 jun. 2011.

Albano, E. L. (2012). "Planejamento, Construção e Avaliação de Objetos de Aprendizagem para Apoio ao Ensino de Teste de Software". Qualificação de Mestrado – Universidade do Vale do Itajaí, Itajaí, 2012.

Barbosa, E. F. and Maldonado, J. C. (2006) "Establishing a Mutation Testing Educational Module based on IMA-CID". In: Second Workshop on Mutation Analysis, p.14, 7-10.

Barbosa, E. F.; Souza, S. R. S. and Maldonado, J. C. (2008) "An Experience on Applying Learning Mechanisms for Teaching Inspection and Software Testing". In: 21st Conference on Software Engineering Education and Training, April.

Bartié, A. (2002) Garantia da qualidade de software: adquirindo maturidade. Rio de Janeiro: Campus.

Beizer, B. (1990) Software Testing Techniques. 2 ed. New York: Van Nostrand Reinhold.

Bertolino, A. (2004) "The (Im)maturity level of software testing." In: ACM SIGSOFT Software Engineering Notes. New York, v. 29, n. 5, p. 1-4, set.

CAPES (2010). "Relatório de Avaliação 2007-2009 Trienal 2010." Disponível em: <http://trienal.capes.gov.br/wp-content/uploads/2011/01/CIÊNCIA-DA-COMPUTAÇÃO-RELATÓRIO-DE-AVALIAÇÃO-FINAL-jan11.pdf>. Acesso em: 05 maio 2011.

Carrington, D. (1997) “Teaching Software Testing”. In: Proceedings of the 2nd Australasian conference on Computer science education, p.59-64, July.

CEEInf. (1999) “Diretrizes Curriculares de Cursos da Área de Computação e Informática.” Disponível em ftp://ftp.inf.ufrgs.br/pub/mec/diretrizes.doc. Acesso em: 12 maio de 2009.

Chen, T. Y. and Poon, P. L. (2004) “Experience With Teaching Black-Box Testing in a Computer Science/Software Engineering Curriculum”. In: IEEE Transactions on Education, vol. 47, no. 1, feb.

Dias Neto, A. C.; Natali, A. C. C.; Rocha, A. R. and Travassos, G. H. (2006) “Caracterização do estado da prática das atividades de teste em um cenário de desenvolvimento de software brasileiro”. In: Simpósio Brasileiro de Qualidade de Software, 5., 2006, Vila Velha: SBC, p. 27-41.

Elbaum, S.; Person, S.; Dokulil, J. and Jorde, M. (2007) “Bug hunt: making early software testing lessons engaging and affordable”. In: 29th International Conference on Software Engineering (ICSE'07).

Gil, A. C. (1999) Métodos e Técnicas de Pesquisa Social. São Paulo: Atlas.

Hilburn, Th. B. and Towhidnejad, M. (2000) “Software Quality: A Curriculum Postscript?” In: 31 SIGCSE Technical Symposium on Computer Science Education.

Janzen, D. S. and Saiedian, H. (2008) “Test-Driven Learning in Early Programming Courses”. In: 39th SIGCSE Technical Symposium on Computer Science Education.

Jones, E. L. and Chatmon, C. L. (2001) “A perspective on teaching software testing”. In: Consortium For Computing In Small Colleges, Proceedings… pp. 92-100.

IEEE Computer Society. (2004) “SWEBOK - Guide to the Software Engineering Body of Knowledge” Disponível em http://www.computer.org/portal/web/swebok/. Acesso em: 11 março de 2012.

Kaner, C. and Padmanabhan, S. (2007) “Practice and Transfer of Learning in the Teaching of Software Testing”. In: Conference on Software Engineering Education & Training, 20 Proceedings… pp. 157 – 166.

Kitchenham, B. and Charters, S. (2007) “Guidelines for performing systematic literature reviews in software engineering (version 2.3)”. Technical report, Keele University and University of Durham.

Mao, C. (2008) “Towards a Question-Driven Teaching Method for Software Testing Course”. In: International Conference on Computer Science and Software Engineering, p. 645 – 648.

SBC - Sociedade Brasileira de Computação. (2003) Currículo de Referência da SBC para Cursos de Graduação em Computação e Informática. Disponível em: http://www.sbc.org.br/. Acesso em: 12 maio de 2009.

SBC - Sociedade Brasileira da Computação. (2005) Currículo de Referência da SBC para Cursos de Graduação em Bacharelado em Ciência da Computação e Engenharia da Computação. Disponível em: http://www.sbc.org.br/. Acesso em: 12 maio de 2009.

Shepard, T.; Lamb, M. and Kelly, D. (2001) “More Testing Should Be Taught”. Communications of the ACM, v. 44, n. 6, Jun.

Wang, M.; Jia, H.; Sugumaran, V.; Ran, W.and Liao, J. (2010) “A Web-Based Learning System for Software Test Professionals”. In: *IEEE Transactions on Education*.