Uma Experiência no Ensino de Inspeção e Teste de Software

Ellen F. Barbosa[1], **Simone R. S. Souza**[1], **André L. S. Domingues**[1], **Alessandra Chan**[1], **Erika N. Höhn**[1], **José C. Maldonado**[1]

[1]Departamento de Sistemas de Computação – ICMC/-USP
São Carlos (SP) – CEP 13560-970, Cx. Postal 668

{francine,srocio,alsd,alechan,hohn,jcmaldon}@icmc.usp.br

Abstract. *Inspection and testing are fundamental activities regarding software development, resulting in great demand for human resources in the area. However, although some initiatives can be identified, academia has not sufficiently answered such demand, mainly in Brazil. At ICMC-USP, two courses for teaching inspection and testing are offered to the undergraduate students – SCE-221: Verification, Validation and Software Testing and SCE-702: Testing and Software Inspection. This work aims at describing our learning experience conducted in the scope of these courses. The adopted approach involves the use of an educational module for inspection and testing allied to the conduction of an experimental study addressing the practical use of concepts, techniques and criteria explored during the semester. The learning effectiveness is measured in terms of the application of the V&V techniques, considering the average number of test cases generated, the coverage degree obtained and the number of defects revealed by the students.*

Resumo. *A percepção de que inspeção e teste são atividades indispensáveis no desenvolvimento de software tem provocado um crescimento significativo na demanda por pessoal capacitado na área. Entretanto, apesar de algumas iniciativas serem identificadas, o meio acadêmico não tem conseguido atender satisfatoriamente à demanda existente, principalmente no Brasil. No caso do ICMC-USP, duas disciplinas voltadas ao ensino de inspeção e teste são oferecidas aos alunos de graduação – SCE-221: Verificação, Validação e Teste de Software e SCE-702: Teste e Inspeção de Software. Este trabalho visa a descrever a experiência de ensino conduzida no contexto dessas disciplinas. A proposta adotada envolve a utilização de um módulo educacional na área de inspeção e teste aliada à condução de um estudo experimental abordando a aplicação prática dos conceitos, técnicas e critérios vistos durante as aulas. A efetividade do aprendizado alcançado é medida com respeito ao uso das técnicas de V&V, sendo considerados o número médio de casos de teste gerados, o grau de cobertura obtido e a quantidade de defeitos identificados pelos alunos.*

1. Introdução

Entre as áreas de conhecimento tidas como essenciais dentro da Engenharia de Software, as atividades de inspeção e teste de software desempenham um papel fundamental na temática de qualidade, sendo fundamentais para a avaliação e a melhoria dos produtos de software desenvolvidos [Myers et al. 2004]. Ressalta-se, entretanto, que apesar dos avanços e pesquisas na área, a utilização sistemática de inspeção e teste em ambientes reais de desenvolvimento ainda não é uma prática comum, principalmente no

Brasil. No contexto nacional, pesquisas realizadas pelo Ministério da Ciência e Tecnologia [DSI/CGSA 2002] indicaram que apenas parte das empresas desenvolvedoras de software no país utilizam algum tipo de inspeção e teste como método para detecção de erros. De modo geral, os resultados obtidos demonstram a dificuldade da indústria em incorporar e aplicar em seu processo de desenvolvimento tanto recursos já disponíveis comercialmente como novas tecnologias investigadas no ambiente acadêmico.

Por outro lado, a percepção de que inspeção e teste são atividades indispensáveis dentro do processo de desenvolvimento tem provocado um crescimento significativo na demanda por pessoal capacitado, sobretudo no ambiente industrial. No entanto, apesar de algumas iniciativas serem identificadas, o meio acadêmico não tem conseguido atender satisfatoriamente à demanda existente. No Brasil, embora exista uma grande quantidade de cursos de Computação/Informática no país, poucos deles possuem uma disciplina especificamente voltada ao ensino dos conceitos relacionados às atividades de inspeção e teste. No caso do Instituto de Ciências Matemáticas e de Computação (ICMC-USP/São Carlos), duas disciplinas abordando aspectos teóricos e práticos de inspeção e teste de software são oferecidas aos alunos de graduação dos cursos de Bacharelado em Ciência de Computação e Bacharelado em Engenharia de Computação – *SCE-221: Verificação, Validação e Teste de Software* e *SCE-702: Teste e Inspeção de Software*, respectivamente.

Nesse cenário, o estabelecimento de mecanismos de apoio ao processo de ensino e aprendizado em inspeção e teste constitui um dos aspectos fundamentais a ser investigado a fim de favorecer o domínio e a disseminação do conhecimento na área. Em linhas gerais, a idéia é fornecer meios para a formação e a capacitação de pessoal tanto em conceitos já consolidados em alguns setores da indústria quanto em tecnologias emergentes, estimulando o processo de transferência tecnológica entre universidade e indústria, e contribuindo para a melhoria da qualidade dos produtos de software desenvolvidos.

O presente trabalho insere-se nessa perspectiva e tem como objetivo descrever a experiência de ensino em inspeção e teste de software conduzida no ICMC-USP, no contexto das disciplinas SCE-221 e SCE-702. A proposta de ensino adotada envolve dois aspectos. O primeiro deles é relacionado com a utilização de um módulo educacional na área de inspeção e teste, desenvolvido a partir da aplicação de uma abordagem integrada para a modelagem de conteúdos educacionais [Barbosa and Maldonado 2006b, Barbosa and Maldonado 2006a]. O módulo é explorado como um instrumento de apoio ao ensino dos aspectos teóricos relacionados.

O segundo aspecto envolve a replicação e condução de um estudo experimental abordando questões práticas da aplicação de inspeção e teste. O estudo experimental foi definido com base em um pacote de laboratório desenvolvido para replicação de experimentos visando investigar a efetividade de técnicas de V&V na detecção de defeitos [Basili and Selby 1987, Maldonado et al. 2006b]. Embora essa perspectiva também possa ser explorada a partir do estudo realizado, o foco deste trabalho consiste em investigar a replicação do estudo experimental como instrumento de apoio tanto no ensino de aspectos práticos de inspeção e teste como na avaliação da efetividade do aprendizado alcançado. Tal efetividade é medida a partir da aplicação das técnicas de V&V, sendo considerados o número médio de casos de teste gerados, o grau de cobertura obtido e a quantidade de defeitos identificados pelos alunos. O uso do pacote de laboratório no contexto de ensino também é avaliado.

O artigo está organizado como se segue. Na Seção 2 é dada uma visão geral dos aspectos de desenvolvimento de módulos educacionais. Na Seção 3 é apresentada a proposta de ensino adotada, com ênfase no desenvolvimento do material didático e na replicação e condução do estudo experimental. Os resultados obtidos são analisados discutidos na Seção 4. Por fim, na Seção 5 são sintetizadas as contribuições deste trabalho e apresentadas as perspectivas futuras de pesquisa.

2. Aspectos do Desenvolvimento de Módulos Educacionais

Entre as linhas que têm sido objeto de pesquisa no contexto de Ensino e Aprendizado destaca-se o desenvolvimento de módulos educacionais – unidades concisas de estudo, compostas por conteúdos teóricos integrados a atividades práticas e avaliações, cuja disponibilização aos aprendizes é apoiada por recursos tecnológicos e computacionais [Barbosa and Maldonado 2006b, Barbosa and Maldonado 2006a].

A Figura 1 ilustra um módulo educacional em termos de seus componentes principais: (1) material didático associado, envolvendo os conteúdos referentes ao domínio de conhecimento que se deseja ensinar; e (2) ambientes educacionais e mecanismos de apoio, representando a infra-estrutura necessária à utilização do módulo.

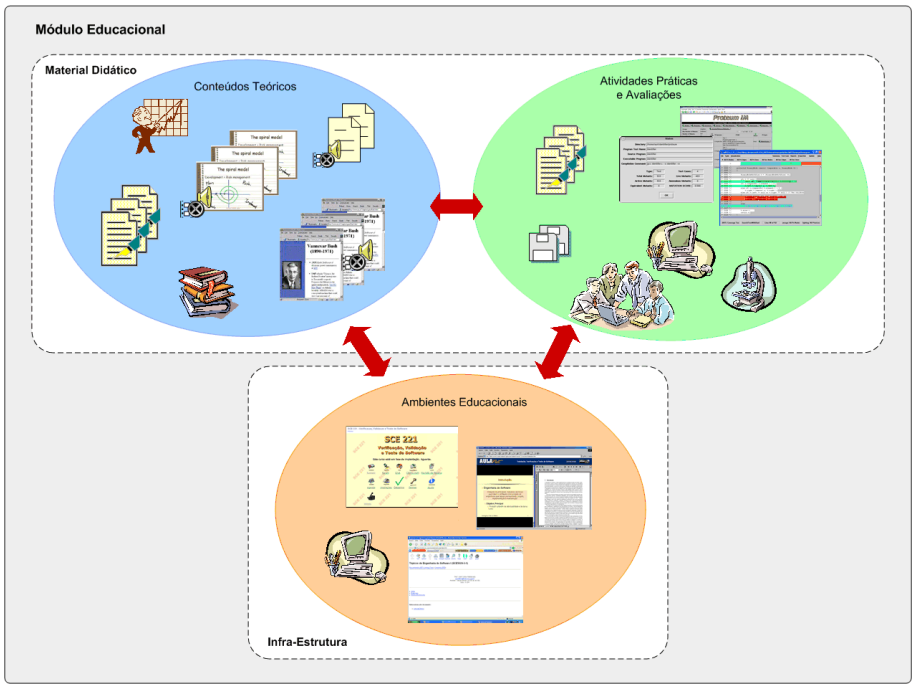

Figura 1. Componentes de um Módulo Educacional.

De acordo com a figura, o material didático é constituído por conteúdos teóricos e conteúdos práticos. Conteúdos teóricos podem ser vistos como livros, artigos, informações e referências na web, transparências, anotações de aula, áudio e vídeo associados, entre outros. Conteúdos práticos são caracterizados em função das atividades e avaliações conduzidas e de seus produtos resultantes (documentos, códigos-fonte, programas executáveis, discussões, estudos experimentais). Ferramentas específicas pertinentes ao domínio de conhecimento bem como os resultados obtidos a partir de sua utilização também são considerados conteúdos práticos.

Conteúdos teóricos e práticos são integrados e disponibilizados aos aprendizes por meio de ambientes e sistemas educacionais. Recursos computacionais e tecnológicos, tais como mecanismos para captura de aulas presenciais e mecanismos de apoio à comunicação e à colaboração, também são caracterizados como parte da infra-estrutura necessária à integração do material didático e à conseqüente disponibilização do módulo educacional.

No decorrer das atividades associadas ao trabalho de Barbosa [Barbosa and Maldonado 2006b, Barbosa and Maldonado 2006a] foram investigados e definidos mecanismos de apoio ao desenvolvimento de módulos educacionais. Duas linhas de atuação foram caracterizadas e são brevemente apresentadas a seguir: (1) Modelagem de Conteúdos Educacionais e (2) Padronização de Processos para a Elaboração de Módulos Educacionais.

2.1. Modelagem de Conteúdos Educacionais

No que se refere à modelagem de conteúdos educacionais, foi estabelecida a abordagem *AIM–CID* (Abordagem Integrada de Modelagem – Conceitual, Instrucional e Didática), caracterizando um conjunto de modelos genéricos para a representação de conteúdos educacionais [Barbosa and Maldonado 2006a]. Cada modelo aborda aspectos distintos da atividade de modelagem. A Figura 2 sintetiza os principais aspectos da abordagem.

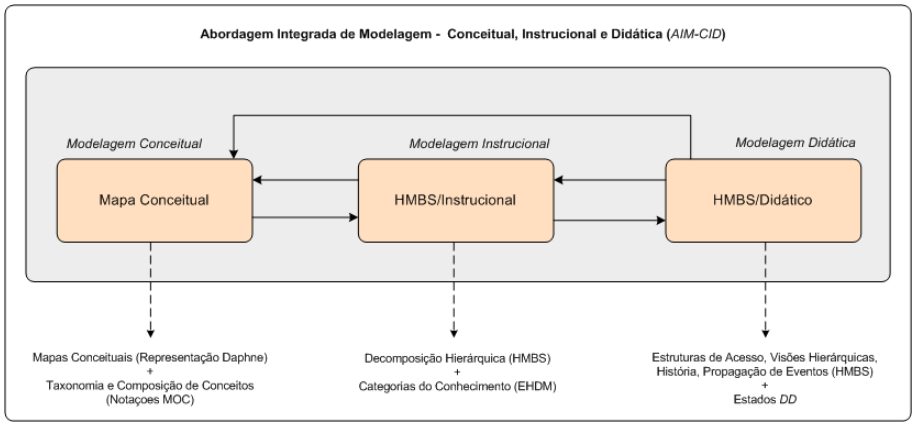

Figura 2. *AIM–CID*: Abordagem para Modelagem de Conteúdos Educacionais.

O modelo conceitual consiste em uma descrição de alto-nível do domínio de conhecimento, na qual são representados os conceitos associados e seus inter-relacionamentos. Basicamente, a construção do modelo utiliza as idéias e regras da técnica de Mapas Conceituais [Novak 1990].

O modelo instrucional é responsável pela definição de informações adicionais relativas aos conceitos previamente identificados. Fatos, princípios, procedimentos, exemplos e exercícios ilustram algumas dessas informações adicionais. Como suporte à construção desse modelo foi adotado o modelo HMBS (*Hypertext Model Based on Statecharts*)[1] [Turine et al. 1997] considerando, nesse nível, somente os aspectos de decomposição hierárquica fornecidos pelo mesmo. O HMBS estendido no nível instrucional foi denominado *HMBS/Instrucional*.

[1]O HMBS consiste em um modelo para a especificação da estrutura e da semântica navegacional de hiperdocumentos, utilizando a técnica *Statecharts* como modelo de especificação formal subjacente.

O modelo didático é responsável pelo estabelecimento de relações de precedência, definindo seqüências de apresentação entre os objetos caracterizados nos modelos conceitual e instrucional. No nível didático, também optou-se pela utilização do HMBS, que foi estendido a fim de possibilitar a representação de "especificações abertas" sobre os aspectos de navegação. Em uma especificação aberta são representadas todas as possíveis seqüências de apresentação entre os objetos modelados. Essa característica é essencial para gerar conteúdos diferenciados, nos quais os tópicos e as seqüências de apresentação são estabelecidos de acordo com aspectos particulares como, por exemplo, aspectos pedagógicos, preferência dos instrutores, perfil dos aprendizes, entre outros. O modelo HMBS estendido no nível didático foi denominado *HMBS/Didático*.

2.2. Processo Padrão para Elaboração de Módulos Educacionais

Similar ao desenvolvimento de software, módulos educacionais requerem o estabelecimento de um processo de desenvolvimento sistemático a fim de produzir produtos com qualidade, passíveis de reúso e evolução. Adicionalmente, aspectos específicos relacionados ao ensino e aprendizado também devem ser considerados – domínio do conhecimento, perfil dos aprendizes, objetivos do curso, estratégias pedagógicas adotadas, entre outros. Nesse sentido foi definido um Processo Padrão para Elaboração de Módulos Educacionais (Figura 3) [Barbosa and Maldonado 2006b]. O processo foi estabelecido com base na norma ISO/IEC 12207, adaptada ao contexto de produção dos módulos por meio da inclusão de práticas específicas de projeto instrucional [Dick et al. 2001] e modelagem de conteúdos educacionais [Barbosa and Maldonado 2006a]. Questões associadas ao desenvolvimento cooperativo e distribuído [Maidantchik and Rocha 2002] também foram preliminarmente investigadas.

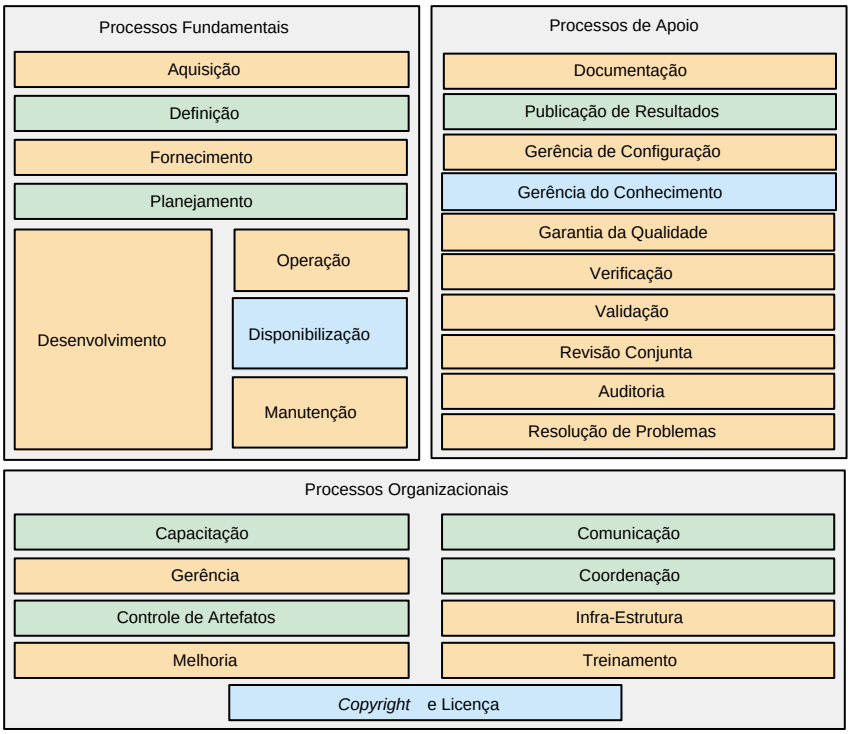

Figura 3. Estrutura Geral do Processo Padrão para Módulos Educacionais.

As atividades do ciclo de vida de um módulo educacional foram estabelecidas por meio de três classes de processos: (1) processos fundamentais; (2) processos de apoio; e (3) processos organizacionais. Comparados aos processos fundamentais da norma ISO/IEC 12207, os processos fundamentais para módulos educacionais diferem, sobretudo, com respeito à inclusão do Processo de Disponibilização, que trata das atividades e tarefas do instrutor/mediador do módulo. Já os processos de apoio diferem da norma pela inclusão do Processo de Gerência do Conhecimento, responsável por tratar os aspectos pertinentes ao gerenciamento e controle do conhecimento sob o qual o módulo está fundamentado. Quanto aos processos organizacionais, estes se diferenciam basicamente pela definição do Processo de *Copyright* e Licença, no qual são abordados aspectos pertinentes aos direitos autorais e aos termos de uso e distribuição do módulo.

Além da padronização de processos para o desenvolvimento de módulos educacionais, aspectos associados à especialização e instanciação do processo padrão definido também foram abordados. Detalhes podem ser encontrados em [Barbosa and Maldonado 2006b].

3. Proposta Adotada nas Disciplinas SCE-221 e SCE-702

As disciplinas SCE-221 e SCE-702 são optativas para os cursos de Bacharelado em Ciência da Computação e Engenharia de Computação, respectivamente. São disciplinas semestrais de 45 horas-aula, correspondendo a três créditos, com três horas semanais, em um total de 15 semanas. Ambas têm como objetivo fornecer uma visão geral da área de Verificação e Validação de Software (V&V), com ênfase em estratégias, técnicas e critérios de teste e inspeção de software e ferramentas associadas que podem ser aplicados na construção de software. Entre os tópicos abordados na ementa destacam-se: qualidade de software; estratégias de V&V – análise estática e dinâmica; terminologia e conceitos básicos de inspeção; métodos para inspeção – inspeção em documento de requisitos e em código-fonte; terminologia e conceitos básicos de teste; técnicas de teste – técnica funcional, estrutural e baseada em erros; ferramentas de teste; e estudos empíricos.

No primeiro semestre de 2006, 52 alunos matricularam-se na disciplina SCE-221 e 36 na disciplina SCE-702. As aulas foram expositivas, adotando-se recursos audiovisuais e utilizando-se o módulo educacional desenvolvido a partir do trabalho de Barbosa [Barbosa and Maldonado 2006b, Barbosa and Maldonado 2006a]. Ao término de cada aula eram propostos exercícios de fixação referentes ao tópico discutido. Como instrumento de avaliação foram adotados: (1) uma prova individual escrita, abordando os conteúdos apresentados; e (2) um trabalho prático, realizado em grupo, abordando a condução de um estudo experimental envolvendo as técnicas e critérios de V&V discutidos em sala de aula.

3.1. Desenvolvimento e Aplicação do Módulo de Inspeção e Teste de Software

Os mecanismos de apoio sintetizados nas Seção 2 foram aplicados inicialmente no contexto de teste de software, propiciando a reestruturação dos materiais didáticos pertinentes. Em linhas gerais, os aspectos de modelagem foram utilizados na estruturação e organização do conhecimento sobre teste, sendo que os processos definidos serviram como base para o desenvolvimento sistemático de um módulo educacional nesse domínio. Conceitos, fatos, princípios, procedimentos, exemplos, informações complementares e exercícios pertinentes a esse domínio foram estruturados, modelados e implementados

na forma de um conjunto de transparências, ao qual foram integradas páginas HTML, documentos-texto, ferramentas específicas do domínio e ferramentas educacionais. A transparência principal do módulo educacional sobre teste, ilustrando a integração entre seus diversos componentes, é apresentada na Figura 4.

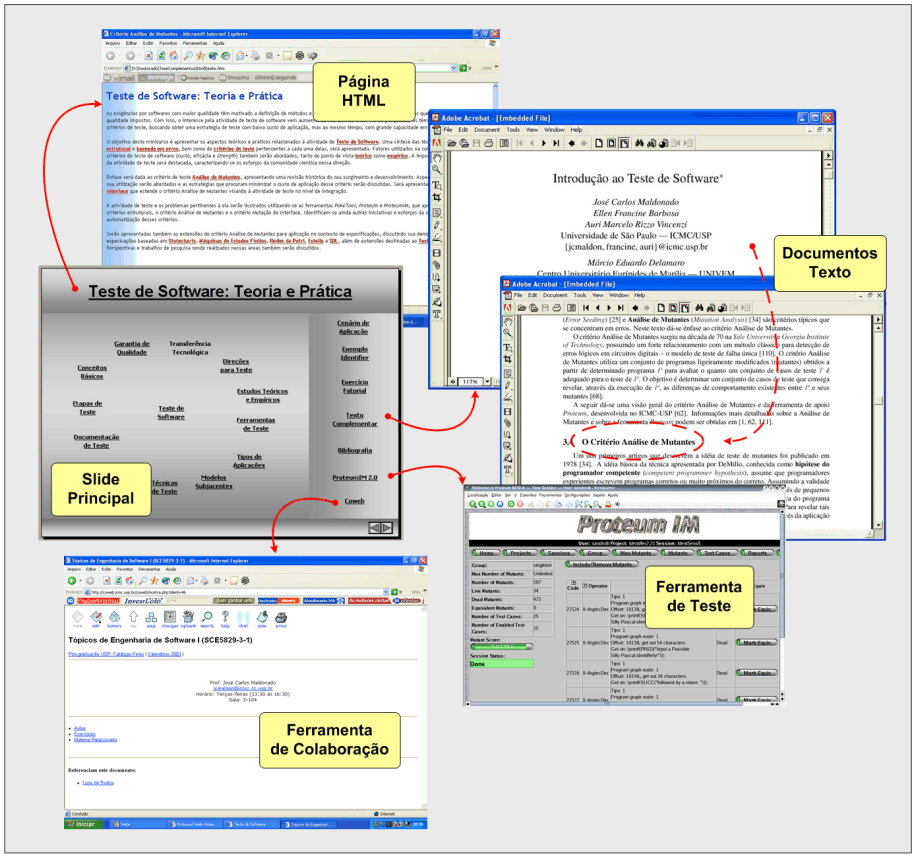

Figura 4. Integração entre Componentes do Módulo Educacional.

Além da incorporação de ferramentas educacionais ao módulo (*CoWeb* [Guzdial et al. 2002]), ressalta-se a integração de ferramentas de teste (*PokeTool* [Chaim 1991] e *PROTEUM* [Delamaro et al. 2001], as quais atuaram como mecanismos de apoio à condução das atividades instrucionais propostas, possibilitando a aplicação prática dos conceitos e demais informações teóricas associadas. O módulo desenvolvido foi disponibilizado, a princípio, em palestras e minicursos. Avaliações informais referentes às características técnicas e pedagógicas foram conduzidas, possibilitando a identificação de problemas e melhorias associadas.

Para a adoção do módulo nas disciplinas SCE-221 e SCE-702 foi necessária sua evolução, sendo modelados e incorporados ao mesmo conceitos e informações referentes à atividade de inspeção de software. Ressalta-se que o módulo foi elaborado de modo a apresentar características de especificação e implementação abertas [Barbosa and Maldonado 2006a] – o usuário, no caso o instrutor/professor, tem total liberdade para decidir, em tempo de execução, quais tópicos e assuntos devem ser tratados e em que seqüência apresentá-los; a navegação no material depende da decisão do professor no momento da exposição do mesmo. Com isso, a idéia é permitir que o módulo seja

personalizado em função de aspectos tais como tempo de apresentação, objetivos e metas de aprendizado, público-alvo, entre outros.

É importante observar que o módulo educacional foi explorado no contexto das disciplinas como um instrumento de apoio ao ensino dos aspectos teóricos relacionados às atividades de inspeção e teste de software. A efetividade do aprendizado alcançado a partir de sua utilização foi avaliada, de maneira indireta, por meio dos resultados obtidos com a condução do estudo experimental, descrito na próxima seção.

3.2. Descrição do Estudo Experimental Conduzido

Estudos experimentais são importantes em Engenharia de Software a fim de se obter resultados objetivos em relação à compreensão, controle, prognóstico e melhoria dos processos de desenvolvimento de software [Maldonado et al. 2006a, Wohlin et al. 2000]. A disponibilidade de evidências sobre efetividade, custo, adequação e riscos de tecnologias disponíveis ou que estão sendo objeto de pesquisa é essencial tanto para tomada de decisão sobre quais tecnologias usar como para direcionar futuras pesquisas.

Um aspecto relevante em experimentação é a necessidade de replicar um estudo experimental. Replicação em diferentes contextos é uma característica importante para qualquer tipo de estudo em Engenharia de Software, constituindo um mecanismo para se obter credibilidade e aprendizado em relação aos resultados obtidos. Replicações são encorajadas pela disponibilidade de pacotes de laboratório, os quais contém todas as informações do experimento, desde a definição até a análise dos dados. Um pacote de laboratório descreve um experimento em termos específicos, permitindo replicação, oportunidades de variação e um cenário para combinação de resultados em diferentes tipos de tratamentos experimentais [Wohlin et al. 2000, Dória 2001]. Os pacotes de laboratório apóiam replicações futuras com diferentes objetivos: (1) confirmar/negar resultados originais; (2) completar a experiência original; e/ou (3) organizar o objeto do estudo para um contexto experimental específico.

Várias iniciativas de condução de estudos experimentais em diversas áreas da computação podem ser identificadas. No caso de V&V, observa-se um esforço significativo no desenvolvimento de estudos experimentais a fim de investigar quais técnicas e critérios de inspeção e teste são mais adequados e efetivos na detecção de defeitos em produtos de software, e de que forma as mesmas podem ser utilizadas de maneira complementar [Basili et al. 1999, Basili et al. 1996, Basili and Selby 1987, Maldonado et al. 2006b, Maldonado et al. 2006a]. Ressalta-se que a condução de estudos experimentais no domínio de inspeção e teste de software pode representar uma base significativa não somente em termos do conhecimento gerado, mas também no que se refere aos processos de ensino e treinamento, e à avaliação da efetividade do aprendizado obtido em relação às técnicas envolvidas. Este aspecto é investigado neste trabalho.

No escopo das disciplinas SCE-221 e SCE-702 foi definido e conduzido um estudo experimental a fim de avaliar o conhecimento adquirido pelos alunos com relação às técnicas e critérios estudados, tanto para inspeção como para teste de software. No desenvolvimento do estudo experimental utilizou-se um pacote de laboratório elaborado e disponibilizado para replicação em experimentos envolvendo técnicas de V&V [Basili and Selby 1987, Maldonado et al. 2006b]. A partir do pacote de laboratório foram extraídas informações com respeito aos programas selecionados, defeitos existen-

tes, formulários para coleta de dados e procedimentos adotados na condução do estudo. $Exp1$ refere-se ao experimento aplicado aos 36 alunos da disciplina SCE-702 (9 grupos), enquanto $Exp2$ refere-se ao experimento conduzido com os 52 alunos da disciplina SCE-221 (13 grupos).

Com relação à inspeção foi utilizada a técnica de leitura de código, conhecida como *code reading* [Myers 1978]. No caso de teste de software, foram utilizados os critérios: (1) da técnica funcional – critério particionamento em classes de equivalência [Myers et al. 2004]; (2) da técnica estrutural – critérios todos-nós [Myers et al. 2004], todos-arcos [Myers et al. 2004] e todos-potenciais-usos [Maldonado 1991]; e (3) da técnica baseada em erros – critério análise de mutantes [DeMillo et al. 1978]. Além disso, como apoio à aplicação dos critérios estruturais e baseados em erros foram utilizadas as ferramentas de teste *PokeTool* [Chaim 1991] e *PROTEUM* [Delamaro et al. 2001], respectivamente.

Também foi considerada na condução do experimento a idéia de teste incremental – estratégia de teste em que pontos positivos de técnicas distintas são combinados em um processo evolutivo de teste. Desse modo, em $Exp1$ foram aplicadas, isoladamente, a técnica de inspeção *code reading* ($T1$) e o critério baseado em erros análise de mutantes ($T2$). A terceira técnica refere-se ao teste incremental ($T3$) que, no caso, consistiu na combinação dos critérios funcionais e estruturais, nesta ordem: particionamento em classes de equivalência, todos-nós, todos-arcos e todos-potenciais-usos. Em $Exp2$ foram aplicadas a técnica de inspeção *code reading* ($T1$) e uma seqüência de critérios estruturais ($T2$): todos-nós, todos-arcos e todos-potenciais-usos. O teste incremental ($T3$), por sua vez, consistiu na combinação de critérios funcionais e baseados em erro, ou seja, particionamento em classes de equivalência e análise de mutantes.

Da mesma forma que os experimentos de Basili e Selby [Basili and Selby 1987] e Maldonado et al. [Maldonado et al. 2006b], a definição do estudo experimental combinou as três técnicas de V&V estabelecidas com três programas distintos – *cmdline*, *nametbl* e *ntree*. Todos os programas continham defeitos. Uma breve descrição dos programas selecionados, bem como informações a respeito do número de linhas de código (LOC) e quantidade de defeitos existentes são apresentadas na Tabela 1.

Tabela 1. Descrição dos Programas Utilizados no Estudo Experimental.

Programa	Descrição	LOC	Defeitos
cmdline	Analisa uma linha de comando para correção sintática e semântica.	268	9
nametbl	Lê comandos de um arquivo e os processa para testar novas funções. As funções implementam uma tabela de símbolos para uma determinada linguagem.	270	8
ntree	Lê comandos de um arquivo e os processa para testar novas funções. As funções implementam uma árvore na qual cada nó pode ter qualquer número de nós filhos.	244	8

Foi realizada uma permutação entre os grupos envolvidos, de modo que cada grupo aplicasse cada técnica uma única vez, sempre considerando programas diferentes. Assim, cada grupo trabalhou com as três técnicas e com os três programas, garantindo-se que todas as técnicas fossem aplicadas a todos os programas. A título de ilustração, a

definição de $Exp1$ relacionando os grupos, técnicas e programas utilizados é apresentada na tabela 2.

Tabela 2. Definição do Experimento 1.

	ntree			cmdline			nametbl		
Teams	T1	T2	T3	T1	T2	T3	T1	T2	T3
1	X	-	-	-	X	-	-	-	X
2	X	-	-	-	X	-	-	-	X
3	X	-	-	-	X	-	-	-	X
4	-	X	-	-	-	X	X	-	-
5	-	X	-	-	-	X	X	-	-
6	-	X	-	-	-	X	X	-	-
7	-	-	X	X	-	-	-	X	-
8	-	-	X	X	-	-	-	X	-
9	-	-	X	X	-	-	-	X	-

É importante observar que as variações na ordem de aplicação das técnicas adotadas em $Exp1$ e $Exp2$ foram definidas com base em um pacote de laboratório desenvolvido para replicação de experimentos visando investigar a efetividade de técnicas de V&V na detecção de defeitos [Basili and Selby 1987, Maldonado et al. 2006b]. Embora essa perspectiva também possa ser explorada a partir do estudo realizado, este não é o foco do trabalho. De fato, neste artigo a idéia é investigar a replicação do estudo experimental como instrumento de apoio tanto no ensino de aspectos práticos de inspeção e teste como na avaliação da efetividade do aprendizado alcançado.

4. Análise dos Resultados Obtidos

Para a condução do estudo experimental, os grupos receberam instruções e formulários específicos a serem preenchidos para cada técnica. No contexto de inspeção, foi solicitado dos alunos a observação de falhas e isolamento dos defeitos existentes nos programas. Já no contexto da atividade de teste, além da detecção dos defeitos, também foram necessárias a geração/seleção de casos de teste, a avaliação de cobertura dos casos de teste em relação às técnicas utilizadas e a identificação de elementos não-executáveis [2]. É importante observar que tais aspectos caracterizam os parâmetros de avaliação a partir dos quais a efetividade de aprendizado dos alunos foi medida.

A Tabela 3 ilustra, para cada grupo de $Exp1$, o total de defeitos encontrados e a quantidade válida e inválida desses defeitos em relação à aplicação da técnica $T1$ (*code reading*). Tomando-se como exemplo o grupo $G1$ e o programa *nametbl* observa-se que dos 8 defeitos existentes no programa, 3 deles foram identificados corretamente por $G1$. Esta análise foi conduzida para todas técnicas de ambos os experimentos.

As figuras 5 e 6 apresentam a quantidade média de defeitos identificados pelos grupos em relação aos defeitos existentes em cada programa, considerando as técnicas $T1$, $T2$ e $T3$ para os experimentos $Exp1$ e $Exp2$, respectivamente. A análise considera somente os defeitos válidos. Considerando o programa *cmdline* e $Exp1$ tem-se que, dos

[2] Um elemento é dito "não executável" se não existe um valor que possa ser atribuído às variáveis de entrada do programa em teste que leve à execução desse elemento [Frankl 1987].

Tabela 3. Defeitos Encontrados x Defeitos Válidos para a Técnica *Code Reading* – $Exp1$.

Grupo	Programa	Def. Existentes	Def. Encontrados	Qtde. Inválida	Qtde. Válida
G1	nametbl	8	5	2	3
G2	nametbl	8	3	1	2
G3	nametbl	8	7	0	7
G4	cmdline	9	7	2	5
G5	cmdline	9	3	0	3
G6	cmdline	9	5	3	2
G7	ntree	8	4	3	1
G8	ntree	8	2	1	1
G9	ntree	8	6	1	5

9 defeitos existentes, os grupos responsáveis pela aplicação de $T1$, $T2$ e $T3$ identificaram, em média, 3.33, 5.33 e 3.33 defeitos, respectivamente. É importante destacar que tais valores referem-se à aplicação isolada de cada uma das técnicas. Analisando-se as técnicas em conjunto, notou-se que para este caso (programa *cmdline* e $Exp1$) todos os 9 defeitos existentes foram identificados pelos grupos. Em outro caso, para o programa *nametbl* e $Exp1$, dos 8 defeitos existentes, somente um deles não foi identificado. Esta mesma análise foi feita para os demais programas, em ambos os experimentos. De fato, observou-se que a maioria dos defeitos existentes nos programas foi revelada ao se considerar a aplicação conjunta das técnicas.

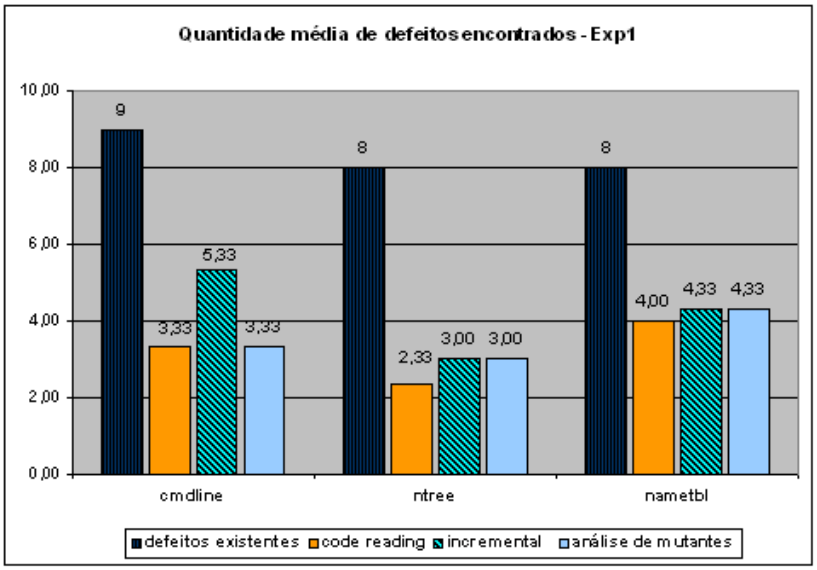

Figura 5. Quantidade Média de Defeitos Identificados – $Exp1$.

A Tabela 4 apresenta características dos programas utilizados com respeito aos requisitos de teste que precisam ser exercitados pelos diferentes critérios de teste envolvidos nos experimentos. Por exemplo, o critério estrutural todos-nós requer que todos os nós do programa em teste sejam exercitados pelos casos de teste. Assim, 91 nós são requeridos para o programa *cmdline*; 81 para *ntree*; e 89 para *nametbl*. De maneira análoga, arcos, associações-potencial-uso e mutantes referem-se aos elementos requeridos pelos critérios todos-arcos, todos-potenciais-usos e análise de mutantes, respectivamente. Elementos não-executáveis – associações não-executáveis e mutantes equivalentes – também

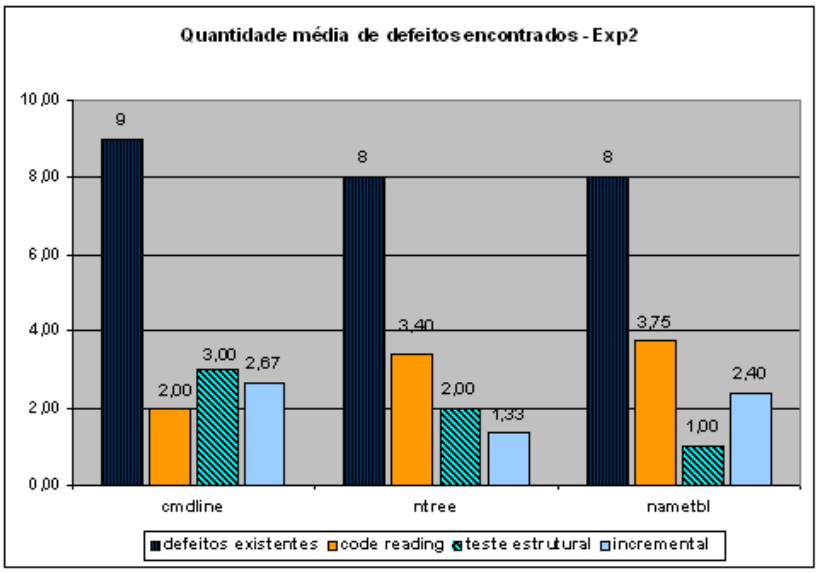

Figura 6. Quantidade Média de Defeitos Identificados – $Exp2$**.**

são ilustrados na tabela. Requisitos de teste e elementos não-executáveis são considerados na geração de casos de casos e na determinação da cobertura dos critérios.

Tabela 4. Características dos Programas Utilizados.

Dados	**cmdline**	**ntree**	**nametbl**
Nós	91	81	89
Arcos	47	31	40
Associações potencial-uso	614	180	184
Associações não-executáveis	102	16	5
Mutantes	2086	1371	1388
Mutantes equivalentes	643	347	309

A Tabela 5 apresenta o número médio de casos de teste gerados pelos grupos a fim de satisfazer os requisitos de teste associados às técnicas $T2$ e $T3$ em ambos os experimentos. A geração de casos de teste foi feita manualmente, tendo como base a especificação dos programas e os elementos requeridos (requisitos de teste) de cada técnica. Por exemplo, para satisfazer a técnica $T2$ (análise de mutantes) no teste do programa *cmdline* foram gerados, em média, 217.7 casos de teste pelos três grupos de $Exp1$ responsáveis pela aplicação da técnica ao programa em questão. No entanto, analisando-se os dados de cada grupo separadamente, observou-se que um dos grupos gerou 510 casos de teste, enquanto os demais grupos geraram 106 e 37 casos de teste. Desconsiderando-se o valor 510, teríamos uma média de 71.5 casos de teste gerados para satisfazer o critério análise de mutantes. Este valor é bastante próximo da média obtida considerando-se os grupos de $Exp2$ e a aplicação da técnica $T3$ (particionamento em classes de equivalência e análise de mutantes) no teste do mesmo programa – 76.5. O aumento médio observado pode ser justificado pelo fato de que em $Exp1$ o critério análise de mutantes foi aplicado isoladamente, ao passo que em $Exp2$ sua aplicação foi realizada em conjunto com outro critério de teste. Tais observações dão indícios de que, embora o grupo responsável pela geração de 510 casos de teste tenha conseguido aplicar corretamente a técnica, este poderia ter tido uma maior eficiência e desempenho no que se refere à geração dos casos de teste. De

fato, o número de casos de teste gerado é um aspecto relevantes a ser considerado no que diz respeito ao custo de aplicação das técnicas de teste em geral. Para os demais grupos não foram observadas discrepâncias na quantidade de casos de teste gerada.

Tabela 5. Número Médio de Casos de Teste Gerados para as Técnicas $T2$ e $T3$.

Experimento	Técnica	cmdline	ntree	nametbl
$Exp1$	Análise de Mutantes – $T2$	217.7	23.0	18.3
	Incremental – $T3$	38.0	14.7	17.0
$Exp2$	Estrutural – $T2$	36.8	10.0	18.7
	Incremental – $T3$	76.5	26.5	27.2

Por fim, as figuras 7 e 8 apresentam as coberturas médias obtidas pelos grupos em relação à aplicação das técnicas $T2$ e $T3$, considerando os experimentos $Exp1$ e $Exp2$, respectivamente. Juntamente com as coberturas obtidas são apresentadas também as coberturas máximas que poderiam ser alcançadas para cada programa e cada técnica. Como exemplo, considere o programa *cmdline* e $Exp1$. Enquanto as coberturas máximas possíveis para as técnicas $T2$ e $T3$ seriam, respectivamente, 100.0% e 83.3%, os grupos obtiveram, em média, uma cobertura de 97.0% para $T2$ e 83.2% para $T3$. Analisando-se as tabelas é possível observar que os grupos conseguiram atingir coberturas adequadas e, na maioria dos casos, muito próximas da cobertura máxima possível. Além disso, ressalta-se que nessa fase os grupos também tiveram que identificar associações não-executáveis e mutantes equivalentes, tarefas realizadas manualmente e consideradas complexas, principalmente para "testadores iniciantes". De fato, não era esperado que os grupos conseguissem determinar com precisão todos os elementos não-executáveis, sobretudo devido à falta de experiência em relação às técnicas utilizadas e à complexidade dos programas envolvidos. Apesar disso, observou-se que os grupos entenderam como realizar essa tarefa, conseguindo atingir o objetivo em relação à avaliação da cobertura para os critérios de teste.

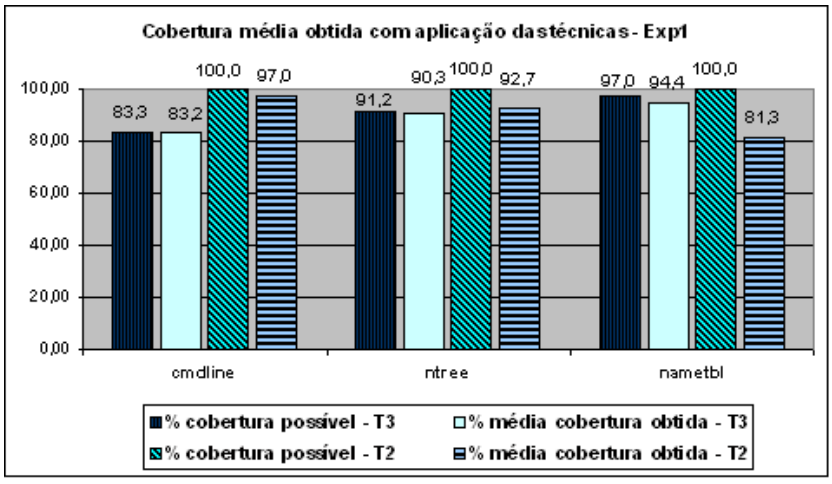

Figura 7. Comparação das Coberturas Obtidas para as Técnicas $T2$ e $T3$ – $Exp1$.

5. Conclusão e Trabalhos Futuros

Este trabalho descreveu uma experiência de ensino em inspeção e teste de software conduzida no contexto de duas disciplinas de graduação oferecidas no ICMC-USP. A proposta

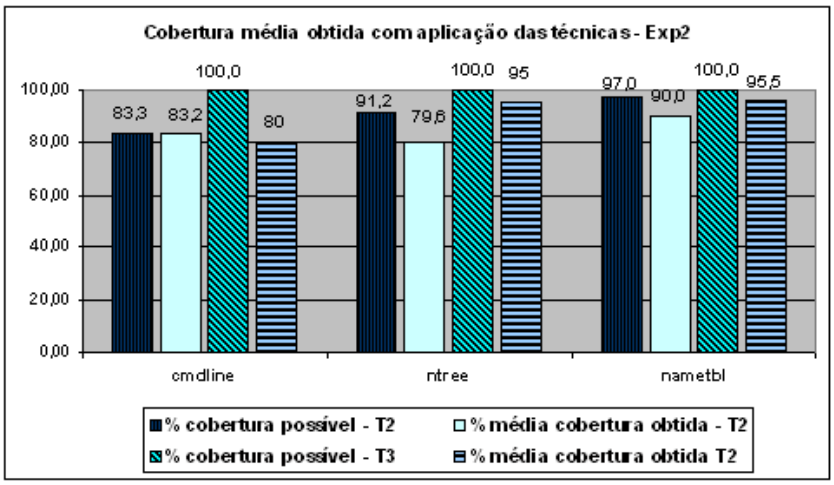

Figura 8. Comparação das Coberturas Obtidas para as Técnicas $T2$ e $T3$ – $Exp2$.

de ensino adotada envolveu: (1) o uso de um módulo educacional, abordando o ensino de aspectos teóricos de inspeção e teste; e (2) a replicação e condução de um estudo experimental, abrangendo a aplicação prática dos conceitos vistos durante as aulas.

Procurou-se avaliar a efetividade de aprendizado alcançado em função dos resultados obtidos com a condução do estudo experimental. De modo geral observou-se que o aprendizado em relação à aplicação das técnicas de inspeção e teste foi satisfatório tanto no que se refere à geração de casos de teste como ao grau de cobertura alcançado e à quantidade de defeitos identificados pelos alunos.

Ressalta-se que os resultados poderiam ser melhorados com o treinamento no uso das ferramentas de teste sendo conduzido durante as aulas, no decorrer do semestre. De fato, durante o semestre o treinamento foi centrado em técnicas e critérios de inspeção e teste. Apenas durante o experimento os alunos tiveram contato direto com as ferramentas de teste (*PokeTool* e *PROTEUM*), sendo observado certo grau de dificuldade em sua utilização.

A experiência de ensino discutida neste trabalho apresentou-se bastante interessante e positiva, em especial quanto à adoção de pacotes de laboratório e estudos experimentais como instrumentos de ensino e avaliação de aprendizado. É importante destacar que o módulo educacional desenvolvido também atuou como mecanismo de apoio ao ensino, sendo que a efetividade do aprendizado alcançado a partir de sua utilização foi avaliada, de maneira indireta, por meio dos resultados obtidos com a condução do estudo experimental. Ainda assim, ressalta-se a necessidade de aplicação e avaliação da proposta em outras turmas, com características diferenciadas.

Por fim, ressalta-se que o trabalho conduzido insere-se no contexto do projeto *QualiPSo (Quality Platform for Open Source Software Proposal)* – projeto de cooperação internacional, apoiado pela comunidade européia, envolvendo representantes de instituições acadêmicas e da indústria. O projeto tem como principal objetivo a definição e a implementação de tecnologias, procedimentos e políticas que apóiem as atuais práticas para o desenvolvimento de software livre. A médio prazo, pretende-se estabelecer um cenário para o desenvolvimento de módulos educacionais livres (*open learning*

materials), sobretudo no contexto de inspeção e teste, de forma a estimular o processo de transferência tecnológica das técnicas de V&V entre universidade e indústria.

6. Agradecimentos

Os autores agradecem o apoio da FAPESP, CNPQ e CAPES.

Referências

Barbosa, E. and Maldonado, J. (2006a). An integrated content modeling approach for educational modules. In *IFIP 19th World Computer Congress – International Conference on Education for the 21st Century*, pages 17–26, Santiago, Chile.

Barbosa, E. and Maldonado, J. (2006b). Towards the establishment of a standard process for developing educational modules. In *36th Annual Frontiers in Education Conference (FIE 2006)*, San Diego, CA. CD-ROM.

Basili, V., Green, S., Laitenberger, O., Shull, F., Sorumgaard, S., and Zelkowitz, M. (1996). The Empirical Investigation of Perspective Based Reading. *Empirical Software Engineering - An International Journal*, 1(2):133–164.

Basili, V. and Selby, R. W. (1987). Comparing the Effectiveness of Software Testing Strategies. *IEEE Transactions on Software Engineering*, SE-13(12):1278–1296.

Basili, V. R., Shull, F., and Lanubille, F. (1999). Building Knowledge Through Families of Experiments. *IEEE Transactions on Software Engineering*, 25(4):456–473.

Chaim, M. L. (1991). PokeTool – Uma ferramenta para suporte ao teste estrutural de programas baseado em análise de fluxo de dados. Master's thesis, DCA/FEEC/UNICAMP, Campinas, SP.

Delamaro, M. E., Maldonado, J. C., and Mathur, A. P. (2001). Interface mutation: An approach for integration testing. *IEEE Transactions on Software Engineering*, 27(3):228–247.

DeMillo, R. A., Lipton, R. J., and Sayward, F. G. (1978). Hints on test data selection: Help for the practicing programmer. *IEEE Computer*, 11(4):34–43.

Dick, W., Carey, L., and Carey, J. O. (2001). *The Systematic Design of Instruction*. Longman, 5 edition.

Dória, E. S. (2001). Replicação de experimentos em engenharia de software. Master's thesis, ICMC-USP, São Carlos, SP.

DSI/CGSA (2002). Qualidade e produtividade no setor de software brasileiro: 2001. Revista do SEPIN/MCT. Brasília, DF.

Frankl, F. G. (1987). *The Use of Data Flow Information for the Selection and Evaluation of Software Test Data*. PhD thesis, University of New York, NY.

Guzdial, M., Rick, J., and Kehoe, C. (2002). Beyond adoption to invention: Teacher-created collaborative activities in higher education. *Journal of the Learning Sciences*.

Maidantchik, C. L. L. and Rocha, A. R. (2002). Managing a worldwide software process. In *Workshop on Global Software Development – International Conference on Software Engineering (ICSE 2002)*, Orlando, FL.

Maldonado, J. C. (1991). *Critérios Potenciais Usos: Uma Contribuição ao Teste Estrutural de Software*. PhD thesis, DCA/FEEC/UNICAMP, Campinas, SP.

Maldonado, J. C., Carver, J., Shull, F., Fabbri, S. C. P. F., Dória, E., Martimiano, L. A. F., Mendonça, M., and Basili, V. R. (2006a). Perspective-based reading: A replicated experiment focused on individual reviewer effectiveness. *Empirical Software Engineering*, 11(1):119–142.

Maldonado, J. C., Fabbri, S. C. P. F., Mendonça, M., Dória, E., Martimiano, L. A. F., and Carver, J. (2006b). Comparing code reading and testing criteria. In *ISESE 2006 - International Symposium on Empirical Software Engineering*, pages 42–44, Rio de Janeiro.

Myers, G. J. (1978). A Controlled Experiment in Program Testing and Code Walkthough Inspections. *Communications of the ACM*, pages 760–768.

Myers, G. J., Sandler, C., Badgett, T., and Thomas, T. M. (2004). *The Art of Software Testing*. John Wiley & Sons, 2nd. edition.

Novak, J. D. (1990). Concept mapping: A useful tool for science education. *Journal of Research in Science Teaching*, 27:937–949.

Turine, M. A. S., Oliveira, M. C. F., and Masiero, P. C. (1997). Designing structured hypertext with HMBS. In *VIII International ACM Hypertext Conference (Hypertext 97)*, pages 241–256, Southampton, UK.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslen, A. (2000). *Experimentation in Software Engineering: an Introduction*. Kluwer Academic Publisher.