Informática Pública vol. 8 (2): 97-107, 2007

Avaliação da Ferramenta "TesteX"

Katia Romero Felizardo[1]
Marilia Amaral[2]
Roberto Vedoato[3]

Palavras-chave
Ferramenta de teste - engenharia de software.

Resumo

Este artigo tem como objetivo apresentar avaliação de uma ferramenta de apoio à aplicação de testes, denominada *"TesteX"*. Essa ferramenta foi desenvolvida para apoiar o experimento *"ProgParTeste-Exper"*. Este visa a contribuir para avaliar a utilização das práticas do *XP*, programação em pares e testes, como metodologia de ensino de programação para alunos de primeiro ano em curso de graduação em Sistemas de Informação. O principal objetivo da avaliação foi obter *"feedback"* sobre a ferramenta, sua viabilidade de uso, facilidade, dificuldades, vantagens e desvantagens.

1. Introdução

Uma das principais preocupações da engenharia de s*oftware* é apresentar modelos que descrevam processos de *software* que possam ajudar o desenvolvedor a executar suas tarefas de forma a conseguir maior qualidade e menor custo. Segundo Silva & Videira [7], a engenharia aprimora práticas de desenvolvimento de *software* bem organizadas e documentadas desde os finais da década de 60.

A partir daí, os métodos, técnicas e ferramentas evoluíram. Mesmo assim, ainda é comum encontrar *software* entregue fora do prazo, com custo além do planejado e que não atende às necessidades dos clientes. Fowler [3] aponta o excesso de formalidade existente nos modelos de processo propostos nos últimos 30 anos como uma das causas desse problema.

Nesse contexto, surgiu um novo segmento na engenharia de s*oftware* [1], [2], [3], [6], que sugere a utilização de processos mais simplificados, focados nas pessoas que os compõem, principalmente o programador. Esses novos processos são conhecidos como "processos ágeis".

O *Extreme Programming (XP)* é um exemplo desse novo conceito de desenvolvimento de *software*. Como destacado por Highsmith [5] *XP* consiste numa série de práticas e regras que permitem o desenvolvimento de *software* de forma dinâmica e ágil, com um mínimo de documentação.

O uso de métodos ágeis, como o *XP*, despertou interesse tanto de empresas como de universidades [4]. Nesse contexto, foi realizado um experimento acadêmico, denominado "*ProgParTeste-Exper*", que objetivou avaliar as vantagens/desvantagens da utilização das práticas do *XP*, programação em pares e testes, como metodologia de ensino de programação para alunos de primeiro ano de curso de graduação em Sistemas de Informação.

Para a condução dos testes, foi criada a ferramenta *"TesteX"*, que será apresentada neste artigo, organizado da seguinte forma: na Seção 2, apresenta-se um resumo dos conceitos básicos do método ágil *XP*; na Seção 3, o experimento *"ProgParTeste-Exper"* para o qual a ferramenta *"TesteX"* foi desenvolvida; na Seção 4, a ferramenta *"TesteX"*; na 5, é descrito o estudo de caso conduzido para avaliar a ferramenta; na 6, são expostos os resultados obtidos com a condução do estudo de caso e na 7 apresentam-se as considerações finais.

[1] E-mail: katia@ffalm.com.br
[2] E-mail: marilia@ffalm.com.br
[3] E-mail: robertovedoato@ffalm.com.br

2. EXTREME PROGRAMMING (XP)

Os métodos ágeis de desenvolvimento de *software*, como é o caso do *XP,* sugerem a utilização de processos mais simplificados, onde a ênfase não é a documentação, nem o processo ou a estratégia de previsibilidade. Esses métodos enfatizam as pessoas, ou seja, os aspectos humanos envolvidos no desenvolvimento do *software*; evitam redundâncias e excessos de documentação e optam pela adaptação às mudanças.

Comunicação, simplicidade, *feedback* e coragem são os quatro valores fundamentais do *XP*. Com base nesses quatro valores foi definido um conjunto composto de doze práticas, apresentadas a seguir.

1. **Jogo de Planejamento** *(Planning Game)*: deve-se determinar rapidamente o escopo das próximas versões, considerando a prioridade de negócio e as estimativas técnicas.
2. **Pequenas Versões** *(Small releases)*: o time coloca rapidamente um sistema simples em produção. Esse mesmo sistema deve ser atualizado freqüentemente e novas versões devem ser entregues em prazo curto, ou seja, em poucos dias ou semanas.
3. **Metáfora** *(Metaphor)*: a metáfora consiste em uma descrição simples do sistema, sendo que esta é utilizada tanto como guia para o desenvolvimento, como mecanismo de comunicação com o cliente.
4. **Projeto simples** *(Simple design)*: baseia-se em projetar o sistema da forma mais simplificada. Deve-se satisfazer somente os requisitos atuais e remover qualquer complexidade extra assim que identificada.
5. **Testes** *(Testing):* o desenvolvimento é continuado somente depois que os testes são escritos pelos programadores. Isso se deve ao fato do desenvolvimento atender aos requisitos dos testes. Os clientes também escrevem testes para validar se os requisitos estão finalizados.
6. **Refatoração** *(Refactoring)*: para Fowler [3], refatoração é uma técnica empregada na reestruturação do código e tem como principal objetivo fazer com que um programa fique mais reutilizável e fácil de entender, mas que preserve seu comportamento externo.
7. **Programação em pares** *(Pair Programming):* esta prática sugere que todo código seja sempre implementado por duas pessoas juntas, diante do mesmo computador, revezando-se no teclado.
8. **Propriedade coletiva** *(Colletive ownership)*: o código é de propriedade coletiva; assim, pertence ao time e cada integrante pode alterar qualquer código quando desejar.
9. **Integração contínua** *(Continnuous integration)*: o sistema é integrado e construído várias vezes por dia, pois logo que uma nova parte do código fica pronta é incorporada ao sistema.
10. **Semana de 40 horas** *(40-hour week)*: para manter o bem-estar do time, o mesmo não deve trabalhar mais do que 40 horas na semana, pois programadores exaustos cometem mais erros.
11. **Cliente dedicado** *(On-site customer)*: os programadores devem ter o cliente disponível o tempo todo, pois é o cliente quem determina os requisitos, atribui as prioridades e responde as dúvidas que os programadores possam ter.
12. **Padronização de Código** *(Coding standards)*: todos os programadores devem escrever o código da mesma forma, seguindo regras comuns que assegurem clareza e comunicação.

Vale destacar a preocupação de Goldman [4] sobre os artigos que têm sido publicados recentemente na literatura abordando o tema *"Ensinar XP e suas práticas"*. Para o autor, um cuidado especial deve ser tomado antes de ensinar ou aplicar práticas isoladas do *XP*, pois seu sucesso é proporcionado claramente pelo uso combinado das doze práticas. Aplicar somente um subconjunto delas pode, em alguns casos, produzir resultados desastrosos.

Ainda para o autor, existem somente duas práticas individuais que produzem bons resultados, mesmo quando destacadas das outras. São a prática de programação em pares e a de testes.

Baseado em Goldman [4] é que a prática de programação em pares e a de testes foram selecionadas para uso como metodologia de ensino em uma disciplina de programação. Na seqüência, será relatado o experimento no qual foram utilizadas as duas práticas.

3. O Experimento

O principal elemento motivador para a realização do experimento "*ProgParTeste-Exper*" foi o fato da disciplina de programação apresentar alto índice de reprovação. Assim, o objetivo foi avaliar a utilização de nova metodologia de ensino para reduzir esse índice através do incentivo do estudo/programação em par e aplicação de testes. Nesse contexto, foi realizado um experimento com oito alunos. Este teve como objetivos específicos:

- despertar o interesse dos alunos "calouros" para a prática de programação;
- avaliar se o aluno mais experiente do par ajuda o menos experiente a desenvolver suas habilidades;
- avaliar se o aluno mais experiente da par melhora suas habilidades de programação;
- desenvolver nos alunos o perfil de trabalho em equipe.

O experimento foi qualificado como pesquisa aplicada, pois objetivou gerar conhecimentos para aplicação prática. Estes servem à solução de problemas específicos da área de sistemas de informação e envolvem verdades e interesses da referida área.

O experimento teve uma abordagem tanto qualitativa como quantitativa.

Qualitativa porque considerou uma relação dinâmica entre o mundo real e o sujeito. Houve a interpretação dos fenômenos e das ações envolvidos e a atribuição de significados desses à realização da pesquisa, o que é básico no processo de pesquisa qualitativa. Como o processo e seu significado são os focos principais de abordagem e também deste experimento, pode-se considerar que a pesquisa realizada teve uma forma de abordagem qualitativa.

Porém, por considerar também elementos quantificáveis, ou seja, por traduzir em números opiniões e informações, diz-se que esta pesquisa teve uma vertente de análise quantitativa. Assim, fez-se necessário o uso de recursos e de técnicas estatísticas para abordar alguns dos resultados obtidos.

As principais atividades definidas para o desenvolvimento do experimento foram:

1. desenvolvimento de ferramenta para auxiliar a atividade de teste;
2. divulgação do projeto para os alunos do Curso de Sistemas de Informação;
3. seleção dos alunos participantes; o pré-requisito exigido foi que o participante estivesse matriculado no Curso de Sistemas de Informação e que tivesse conhecimentos básicos em programação;
4. definição do material a ser utilizado na pesquisa; divisão de tópicos e assuntos abordados;
5. desenvolvimento do material; preparação dos exemplos para treinamento dos alunos;
6. preparação dos exercícios que serão desenvolvidos;
7. treinamento dos participantes, com exemplos práticos de programação;
8. desenvolvimento dos problemas propostos pelos membros do grupo;
9. análise dos resultados obtidos com o desenvolvimento dos exemplos práticos de programação.

A primeira atividade relevante para a condução do experimento foi implementação de uma ferramenta para auxiliar a atividade de teste. Essa ferramenta será apresentada na seqüência sob dois aspectos: os funcionais e de implementação e os aspectos operacionais.

4. A Ferramenta "TesteX"

Quanto aos aspectos funcionais, a ferramenta *"TesteX"* dá suporte à atividade de teste durante as aulas práticas da disciplina de programação. O objetivo da ferramenta é proporcionar ao aluno um retorno rápido sobre o acerto da implementação do exercício.

Quanto aos aspectos de implementação, utilizou-se a linguagem "C", com armazenamento de dados em arquivos texto. No caso da linguagem de programação, foi escolhida a "C" porque é a utilizada na disciplina de programação. Quanto ao armazenamento dos dados, escolheu-se a utilização de arquivos-texto, pois os alunos do primeiro ano ainda não estudaram banco de dados. O objetivo foi utilizar os conceitos estudados em sala de aula para desenvolver a ferramenta.

A Figura 1 apresenta o modelo utilizado pelos professores da disciplina de programação para escrita dos enunciados que compõem as listas de exercícios. Como determinado pela "Prática 1", os testes são escritos antes da implementação. O primeiro conjunto de testes é escrito pelo professor responsável pela disciplina, mas o aluno pode, a qualquer momento, gerar seus próprios requisitos de teste.

Faça um programa (menor.cpp) que leia 3 números inteiros e e imprima o menor deles.

NÚMERO DE RESPOSTAS **2**

ENTRADA DE DADOS

n1: 1
n2: 2
n3: 3
n1: 2
n2: 3
n3: 4

GABARITO (Menor número)

1
2

Figura 1: Enunciado do exercício

Cada enunciado é subdividido em 3 partes: *"Número de Respostas"*, *"Entrada de Dados"* e *"Gabarito"*. Sabe-se que um caso de teste é composto pela entrada e a correspondente saída esperada. Os resultados da execução são examinados para verificar se o programa operou de acordo com o desejado. Assim, o *"Número de Respostas"* define o total de casos de testes, a *"Entrada de Dados"* representa o conjunto de dados necessários para execução do programa e o *"Gabarito"* define os resultados de cada uma das execuções.

Além de elaborar a lista de exercícios no formato especificado na Figura 1, é responsabilidade do professor gerar um arquivo *".TXT"* que contenha o conjunto de respostas esperadas para cada exercício. Assim, para o exemplo *"menor.cpp"* foi gerado o arquivo *"GABARITO.TXT"*. Este arquivo será utilizado pela ferramenta *"TesteX"* e é exibido na Figura 2. Como se pode observar, é gravada apenas uma resposta por linha.

Figura 2: Gabarito

De posse dos casos de testes, cada dupla implementa uma solução para o problema proposto.

```
int  n1,n2,n3,menor;
printf("\nDigite o numero de respostas: ");
scanf("%d",&numResp);
n1=1;
while(numResp)
{
  n2=n1+1;
  n3=n2+1;
  if(n1<n2 && n1<n3)
   menor=n1;
  else if (n2<n3)
   menor=n2;
  else menor =n3;
  Grava(menor); ←
  numResp--;
  n1++;
}
```

Figura 3: Implementação da solução

A Figura 3 ilustra uma possível solução para identificar, dentre três números inteiros, o menor deles. A linha de código em negrito enfatiza o uso da função *"Grava()"*, previamente implementada e disponibilizada aos alunos. Essa função apóia a gravação das respostas geradas pelos programas em arquivo. Assim, a função é utilizada sem gerar preocupação nos estudantes com relação a manipulação de arquivos.

A Figura 4 exibe o arquivo produzido pela execução do código da Figura 3. Vale destacar que todo arquivo gravado pela função *"Grava()"* é criado no diretório *"C:\EX\"* e recebe o nome de *"RESPOSTA.TXT"*.

O próximo passo consiste em verificar se o programa operou de acordo com o esperado. Para isso, é utilizada a ferramenta *"TesteX"*.

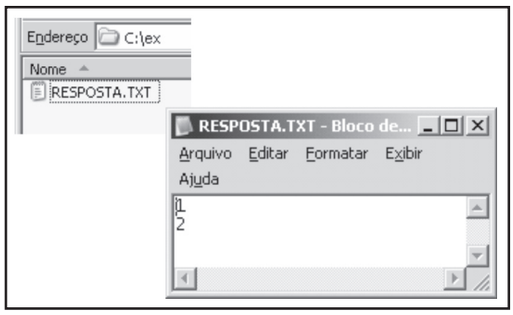

Figura 4: Gravação das respostas em arquivo

A ferramenta pode ser executada de três maneiras diferentes. A Figura 5a representa a primeira possibilidade que é através do *"Menu de opções"*. Para isto, basta selecionar o item *"TesteX.exe"*, como destacado.

A segunda possibilidade, Figura 5b, é através da criação de um atalho, disponível no "*desktop*", para o executável da ferramenta.

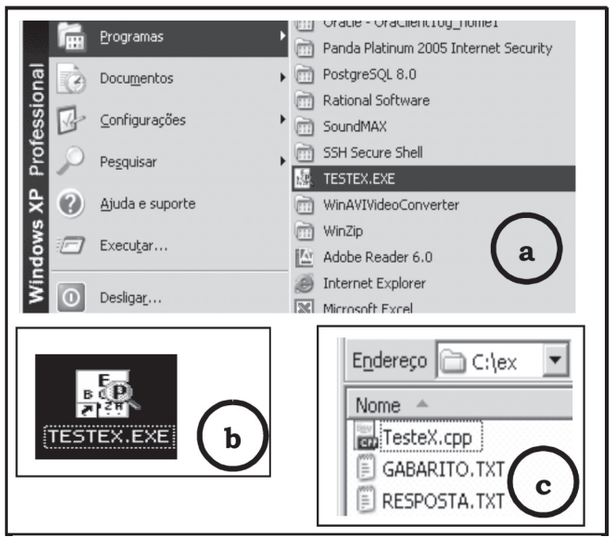

Figura 5: Possibilidades de Execução da ferramenta

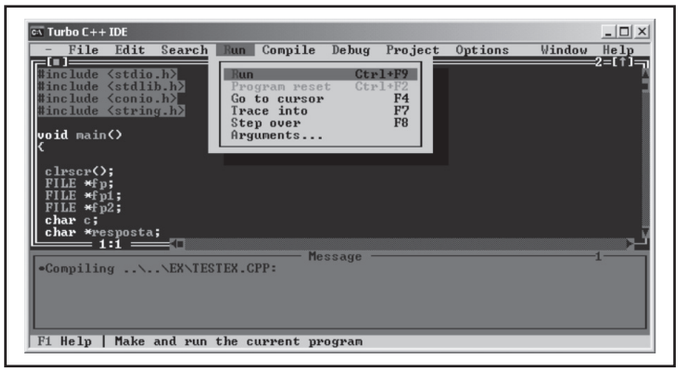

Figura 6: Execução da ferramenta pelo TurboC 3.0

Ainda é possível executar o próprio código fonte da ferramenta (Figura 5c). Para realizar esse procedimento, é necessário abrir o código *"TesteX.cpp"*, Figura 6, e escolher a opção de menu *"Run"*, indicada pela seta.

Para realizar o teste, a ferramenta basicamente compara o conteúdo do arquivo denominado *"GABARITO.TXT"*, com o conteúdo do arquivo *"RESPOSTA.TXT"*, Figura 7. Para que a ferramenta execute essa conferência é necessário que os dois arquivos estejam em um mesmo diretório, previamente determinado. Nesse caso, o diretório escolhido foi o *"C:\EX\"*.

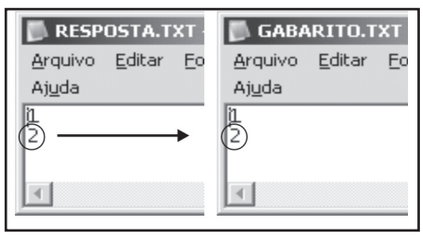

Figura 7: Conferência: resposta x gabarito (respostas corretas)

Finalizada a conferência, o *"TesteX"* exibe a lista de erros e acertos em dois formatos distintos: em tela e em arquivo. A Figura 8 demonstra o primeiro formato. Para o exemplo em uso, detectou-se acerto em todas as saídas produzidas. A mensagem padrão exibida neste caso é *"OK CONFERE",* como pode ser visto no realce da Figura 8.

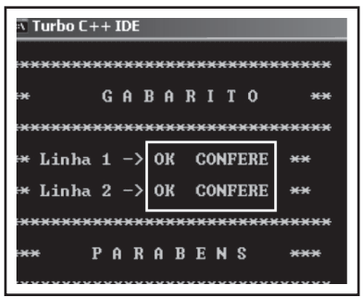

Figura 8: Resultado da correção exibida em tela (respostas corretas)

A Figura 9 demonstra o segundo formato. Para cada sessão de teste finalizada o *"TesteX"* gera um arquivo denominado *"CONFERE.TXT"* no diretório padrão, *"C:\EX\"*, utilizado pela ferramenta.

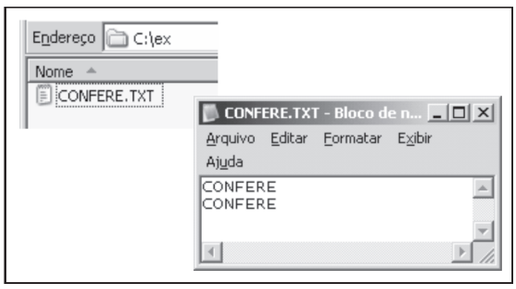

Figura 9: Resultado da correção exibida em arquivo (respostas corretas)

As figuras 10, 11 e 12 representam uma sessão de teste que detectou valores de saída (Figura 10a) diferentes dos esperados (Figura 10b).

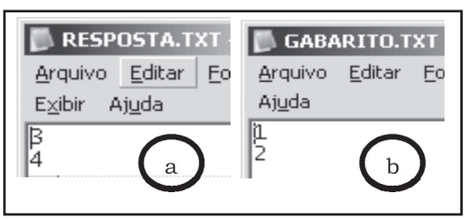

Figura 10: Conferência: resposta x gabarito (respostas incorretas)

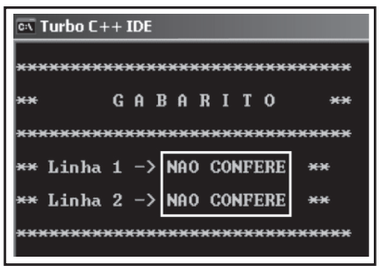

Figura 11: Resultado da correção exibida em arquivo (respostas incorretas)

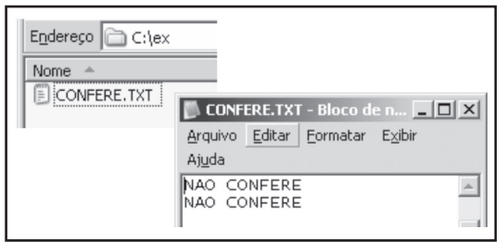

Figura 12: Resultado da correção exibida em arquivo (respostas incorretas)

Nesse caso, a mensagem padrão exibida é *"NÃO CONFERE"* (Figura 11).

5 Estudo de Caso: Avaliação da Ferramenta "TesteX"

O estudo de caso que será apresentado neste artigo relata a avaliação da ferramenta *"TexteX"*, descrita na Seção 4. O objetivo principal desse estudo foi obter uma primeira avaliação da ferramenta, sua viabilidade de uso, facilidade, dificuldades, vantagens e desvantagens.

O estudo foi conduzido com oito alunos de curso de graduação em sistemas de informação, em cinco etapas, sendo que a primeira foi monitorada e acompanhada e as demais, realizadas remotamente. Apesar da relevância da pesquisa, sabe-se que a primeira edição do experimento aqui relatado não utilizou uma população expressiva. Por isso, os resultados obtidos ainda não podem ser considerados como finais.

O projeto experimental do estudo de caso está apresentado na Tabela 1. A primeira etapa, também chamada de *"Treinamento",* foi conduzida de maneira controlada, com presença dos professores, treinamento na ferramenta, controle de tempo para resolução dos exercícios e possibilidade de esclarecimento de dúvidas. As demais, nas quais os participantes controlaram o próprio tempo e horário de execução, utilizaram o conhecimento adquirido na primeira etapa e não existia a possibilidade de esclarecimento de dúvidas.

Nas etapas posteriores ao treinamento, o objetivo foi avaliar como seria a utilização da ferramenta, como ela se comportaria, quais as dificuldades encontradas, assim como as vantagens e desvantagens.

Considerando que todos os participantes cursam a disciplina de "Programação I", considerou-se que eles possuíam o conhecimento teórico necessário para resolver os exercícios (C1- Comandos de Entrada e Saída, C2- Estrutura Condicional, C3- Estrutura de Repetição, C4- Estruturas de Dados Homogêneas: Vetores e Matrizes), de forma que o treinamento restringiu-se a explicações de como utilizar a *"TesteX"*.

Tabela 1 – Projeto Experimental

TREINAMENTO					**EXECUÇÃO**			
Etapa 1					**Etapa 2**	**Etapa 3**	**Etapa 4**	**Etapa 5**
C1	C2	C3	C4	C5	C1	C2	C3	C4
Conhecimento em: C1 = Entrada e Saída de Dados; C2- Estrutura de Seleção; C3- Estrutura de Repetição; C4- Estrutura Homogênea de Dados: Vetores e Matrizes.								

Para condução do treinamento os alunos foram divididos em duplas. Inicialmente, para todos foi ministrada uma sessão teórica sobre a ferramenta, com duração de 1h30min.

Ao término da primeira etapa, considerou-se que os alunos tinham adquirido experiência na ferramenta de forma que, para as quatro seguintes, não foi aplicado nenhum treinamento adicional.

O Projeto Experimental das etapas seguintes ao treinamento foi determinado da seguinte maneira: todos os participantes faziam parte de uma dupla e resolviam os mesmos problemas. A fase de execução

do estudo de caso foi realizada ao longo de quatro semanas.

Ao término das cinco etapas, foi possível concluir alguns pontos sobre a ferramenta, os quais estão descritos a seguir.

6. Principais resultados obtidos

Como dito anteriormente, um dos objetivos desse estudo de caso foi verificar a viabilidade de utilização do *"TesteX"* Assim, após o término da primeira etapa, os participantes foram questionados, informalmente, a esse respeito. A maioria considerou que a ferramenta havia sido útil para realização dos exercícios e que era fácil de ser utilizada. Vale ressaltar que todos os participantes estavam muito motivados com o estudo de caso.

Durante a primeira etapa, monitorada, não surgiram muitas dúvidas. Em relação à *"TesteX",* foi constatado que a duração do treinamento foi suficiente, pois a ferramenta não exige muita aprendizagem para seu uso. Assim, pôde-se perceber que é viável utilizar a ferramenta.

Durante as demais etapas, constatou-se que a ferramenta apóia a atividade de teste e tem como uma de suas vantagens o fato de exibir os relatórios para conferência em dois formatos, em tela e em arquivos. Finalmente, ela permite que o aluno tenha retorno rápido sobre sua produtividade em termos de acertos e erros.

Os estudantes aprovaram a implementação da ferramenta na mesma linguagem que utilizavam na disciplina de "Programação I", pois assim se sentiam familiarizados com ela e motivados a utilizá-la.

Uma de suas desvantagens é o uso de diretório e nome de arquivos pré-fixados. Devido a isso, é permitido ao usuário trabalhar apenas com um programa por vez, pois os arquivos gerados são nomeados, a cada execução, com nomes pré-definidos e armazenados em diretório padrão, como por exemplo, o *"c:\ex\Confere.txt"*.

É importante ressaltar que, com relação ao contexto deste artigo, o objetivo principal foi avaliar a ferramenta *"TesteX"* por meio de um estudo de caso. Mesmo assim, serão relatadas brevemente algumas considerações sobre a aplicação da "Prática 1" (Teste) como metodologia de ensino, já que essa prática está diretamente relacionada com o uso de *"TesteX"*.

Durante a condução do experimento, percebeu-se que apenas 50% dos alunos escreveram programas mais refinados para atender aos requisitos pré-estabelecidos pelos testes, ou seja, esses alunos se preocuparam em tratar situações que não "poderiam" acontecer, proporcionando assim uma melhoria no código fonte.

Também foi possível observar que o enunciado de exercícios contendo casos de teste estimulou os alunos a questionarem o professor sobre detalhes do que devia ser feito, antes de iniciarem a implementação da solução.

Para finalizar, vale destacar que o uso de testes em uma população de alunos cursando o primeiro ano, mesmo sendo uma população ainda não expressiva, proporcionou uma conscientização da importância da qualidade no desenvolvimento de *software,* desde o contato inicial desse aluno com a programação.

7. Considerações finais

Apresentou-se neste artigo o relato de um estudo de caso que foi realizado com o objetivo de fazer uma avaliação prévia da ferramenta "*TesteX"*. Considerando-se que essa avaliação era essencialmente qualitativa, ou seja, para avaliar a viabilidade de uso da ferramenta em um processo de utilização real, as principais considerações a fazer são: o uso da *"TesteX"* facilitou a realização das atividades de teste como um todo e contribuiu para a redução de erros em código fonte, pois viabilizou a aplicação continuada dos testes.

Keyword

Software Testing Tool - Software Engineering

Abstract

This article aims to present a support tool to tests application, named "TesteX". The tool was developed to support the experiment "ProgParTeste-Exper." This experiment intends to contribute to the evaluation of the usage of XP practices, pair programming and testing, as teaching methodology of programming for students of the first year of a graduation course in Information Systems. The main goal of the evaluation was to obtain a feedback about the tool, its usage viability, facility, difficulties, advantages and disadvantages.

Referências Bibliográficas

[1] BECK, K. **Extreme Programming Explained**: Embrace change. Reading, Massachusetts: Ed. Addison-Wesley, 1999.

[2] COCKBURN, A. **Agile software development.** Boston: Addison Wesley, 2002.

[3] FOWLER, M. **The New Methodology**, 2003. Disponível em: <http://www.martinfowler.com/articles/newMethodology.html>. Acessado em: 30 set, 2006.

[4] GOLDMAN, A.; Kon, F; Silva, P. J. S.; Yoder J. W. **Being Extreme in the Classroom: Experiences in Teaching XP.** In Journal of the Brazilian Computer Society, volume 10, number 2, pp. 1-17. Nov. 2004.

[5] HIGHSMITH, J. **Does Agility Work? Software Development**. Canadá, v.10, n.6, p.30-36, jun. 2002.

[6] SCHWABER, K.; Beedle, M. **Agile Software Development with SCRUM**. Prentice-Hall, 2002.

[7] SILVA, A. M. R.; VIDEIRA, C. A. E. **UML, Metodologias e Ferramentas Case**. Lisboa: Centro Atlântico, 2001.

Sobre os autores

Katia Romero Felizardo[1]

Graduada em Tecnologia em Processamento de Dados pelo Centro de Estudos Superiores de Londrina (1999) e Mestre em Ciência da Computação pela Universidade Federal de São Carlos (2003). Atualmente, é professora titular da Universidade Estadual do Norte do Paraná (UENP) e professora assistente da Universidade Estadual de Londrina e da Universidade do Oeste Paulista. Tem experiência na área de ciência da computação, com ênfase em engenharia de *software*, atuando principalmente nos seguintes temas: validação, verificação e teste.

Marilia Amaral[1]

Graduada em Ciência da Computação pela Universidade Estadual de Londrina (1997) e Mestre em Ciência da Computação pela Universidade Federal do Rio Grande do Sul (2002), está concluindo o doutorado no Programa de Pós- Graduação em Engenharia e Gestão do Conhecimento da Universidade Federal de Santa Catarina. Atualmente, é professora titular da Universidade Estadual do Norte do

Paraná (UENP) e pesquisadora da Universidade Federal de Santa Catarina. Tem experiência na área de ciência da computação, com ênfase em inteligência artificial, atuando principalmente nos seguintes temas: inteligência artificial, informática na educação, modelagem de conhecimento e desenvolvimento de sistemas *web*.

Roberto Vedoato[1]

Graduado em Ciência da Computação pela Universidade Estadual de Londrina (1995), Especialista em Ciência da Computação pela Universidade Estadual de Londrina (1998) e Mestre em Ciência da Computação pela Universidade Federal do Rio Grande do Sul (2003). Atualmente, é Professor da Universidade Estadual do Norte do Paraná (UENP). Tem experiência na área de ciência da computação, com ênfase em interação humano - computador e engenharia de *software*. Atua, principalmente, nos seguintes temas: casos de uso, cenários, abordagem orientada a objetivos.

[1] Departamento de Informática
UENP- Universidade Estadual do Norte do Paraná – FFALM - Faculdades Luiz Meneghel
Rodovia Br 369 Km 54 - CEP 86360-000 – Bandeirantes/PR
http://www.ffalm.edu.br/
katia@ffalm.br, marilia@ffalm.br, robertovedoato@ffalm.br