Ortoprog - Verificação Automática de Exercícios Computacionais Via Web

Alexandre Gonçalves Silva, Roberto de Alencar Lotufo

FEEC – Universidade Estadual de Campinas (Unicamp)
Caixa Postal 6101 – 13081-970 – Campinas – SP – Brasil

{alexgs,lotufo}@dca.fee.unicamp.br

Abstract. *This work describes the* Ortoprog *platform of submission, and computational exercises automatic verification. The goal is to assist a discipline which lacks of practical complementation of the subjects seen in class. Exercises are proposed and the students receive a remote automatic feedback, in any time, in relation to accuracy of their solutions. Moreover, illustrative reports with numeric outputs, images, graphics, run time, at last, a results history, from each submission, is maintained and can be accessed after the end of specific activity. From teaching side, there is dynamism in making activities available, selecting tests, resubmiting deliveries automatic from new tests, and organizing deliveries. A delivered programs structural analysis, and results comparation becomes facilitated by this organization.*

Resumo. *Este trabalho descreve a plataforma* Ortoprog *de submissão e verificação automática de exercícios computacionais. O objetivo é assistir uma disciplina que careça de complementação prática dos assuntos vistos em aula. Exercícios são então propostos e os alunos recebem um* feedback *automático remotamente, a qualquer tempo, em relação a exatidão de suas soluções. Além disto, relatórios ilustrados com saídas numéricas, imagens, gráficos, tempo de execução, enfim, um histórico de resultados, a cada submissão, é mantido e pode ser acessado após o término de uma dada atividade. Do lado do docente, há dinamismo de disponibilização de atividades, seleção de testes, re-submissão automática das entregas a partir de novos testes e organização das entregas. Uma análise estrutural dos programas entregues e comparação de resultados torna-se facilitada por esta organização.*

1. Introdução

Em certas disciplinas, é visível a importância da experimentação dos assuntos vistos em aula, de forma que o aluno sinta as dificuldades e reforce o conteúdo teórico. Esta é uma prática entre professores da área de computação. Porém, conforme a complexidade e quantidade dos exercícios surgem dificuldades como agilização da organização dos exercícios entregues e a verificação da correção dos mesmos. Normalmente pede-se um relatório impresso, ou arquivos anexados em *e-mail* ou em disquete. No entanto, havendo computadores e acesso a Internet para todos, torna-se interessante este uso com o intuito de facilitar o trabalho do professor e, principalmente, de possibilitar uma interação mais dinâmica entre aluno-sistema [Pardo 2002, Hendrich 2002] e entre alunos a fim de atingir um objetivo comum que é a solução de um dado problema. Para tanto, é essencial que haja também uma ferramenta de troca de mensagens como, por exemplo, a lista de discussão *Mailman*[1]. A plataforma *Ortoprog* surgiu da necessidade de um sistema de verificação automática de computação científica – vista a variedade de soluções de código obtida nas atividades de programação de algumas disciplinas da Unicamp – e também da observação do funcionamento do sistema *SuSy* [Kowaltowski 2003]. A diferença fundamental está na forma como os resultados do programa do aluno são comparados com um dado gabarito de saídas. Enquanto o *SuSy* realiza basicamente comparação de arquivos textos criados a partir da saída padrão dos programas (uso do *diff* do *Unix*), este trabalho consiste na comparação, em memória, de estruturas Python[2] [Lutz 1998], uma

1 http://www.list.org
2 http://www.python.org

linguagem interpretada, orientada a objetos, bastante genérica, de altíssimo nível (*very high-level*), de rápida prototipação (RAD), convidativa para iniciantes em programação, entre outras qualidades.

2. Plataforma *Ortoprog* de Verificação Automática

A plataforma *Ortoprog*, cujos casos de usos essenciais são mostrados na Figura 1, é desenvolvida em CGI[3]/Python e se divide basicamente em três modalidades:

(i) ***Administração*** – Inclusão de atividades, seleção de testes, definição de prazos e, se houver necessidade, re-submissão automática dos exercícios entregues com novos testes. Além disto, permite visualização do histórico de submissões com indicações de entrega no prazo, ilustrações de acertos, erros e tempo de execução de cada teste, código-fonte entregue, exibição de relatório gerado automaticamente e de relatório entregue (*.txt*, *.pdf*, *.ps* ou *.html*). O sistema também realiza a criação automática de um arquivo compactado, para *download*, com todas as soluções dos alunos. A administração é destinada aos professores, portanto, o acesso só é possível fornecendo-se um nome de usuário e senha especiais;

(ii) ***Validação*** – Diz, para cada teste realizado, se a resposta é correta/incorreta (pela comparação a um "Módulo Gabarito") ou inexistente. A Figura 2 ilustra este processo. A caixa com o símbolo "=" se encarrega de comparar todas as n variáveis (n testes) em memória, V_1, V_2, ... V_n, definidas pelo "*Script* de Testes", entre os resultados da submissão $V_i^{(a)}$ e do gabarito $V_i^{(b)}$, $i \in [1, n]$. A validação pretende avaliar a exatidão do algoritmo implementado, pelo aluno, para cada tipo de entrada estipulado pelo professor;

(iii) ***Visualização*** – Nesta modalidade, não é necessária a implementação de um gabarito. Uma página *HTML*, com saídas numéricas, gráficos ou imagens, é gerada automaticamente e armazenada como resultados de cada aluno, conforme as especificações dadas pelo "*Script* de Testes". A partir desta página, o professor pode, por exemplo, verificar a qualidade ou forma de uma imagem produzida por um algoritmo, o perfil de um gráfico gerado por uma função, os valores de uma matriz, um índice de correlação, de erro, ou outro valor desejado.

Caso o professor opte em utilizar a modalidade *(ii)*, então ele deverá implementar também o programa correto cujas saídas servirão como gabarito ao *Ortoprog*, ou então evidenciar diretamente as estruturas de dados com os valores esperados. Na modalidade *(iii)* – uma abordagem nova em relação ao sistema *SuSy* – isto não se faz necessário pois a idéia, neste caso, é a de verificação dos resultados das soluções por comparação visual.

A desvantagem desta arquitetura reside no fato de a programação ser apenas em Python ou em C/C++. Nesta última opção, deve-se seguir algumas restrições de programação (bem documentadas) de modo a permitir a construção *on line* de *wrappers*, ou seja, a plataforma recebe o código, realiza a compilação, exibe os resultados da mesma e, se não houver erro, cria um módulo Python com as funções *buit-in* do aluno que, a partir deste momento, passam a ser tratadas normalmente em qualquer das modalidades do corretor.

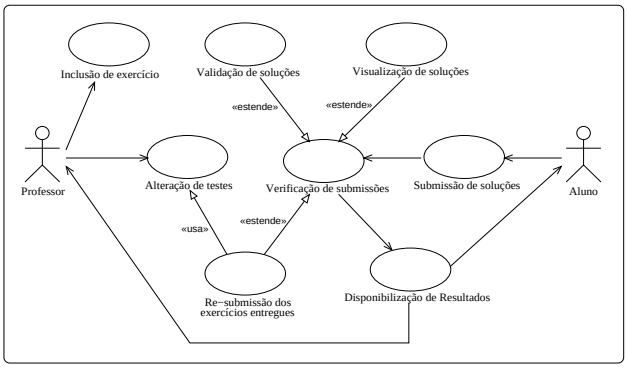


Figura 1. Casos de uso da plataforma *Ortoprog*

3 *Common Gateway Interface*

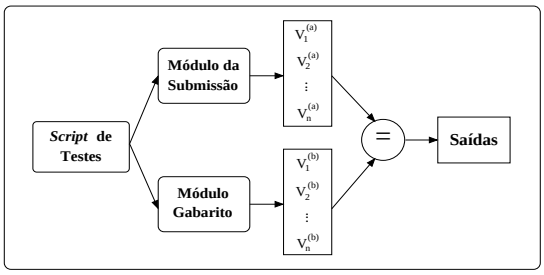


Figura 2. Modalidade de Validação

3. Testes e Resultados

A seguir será visto o procedimento de implementação de exercícios através de exemplos, e alguns resultados obtidos. A Figura 3a mostra um exemplo de "*Script* de Testes", da modalidade de validação, para as funções "mymeshgrid" e "mydisk" solicitadas pelo professor num dado exercício (os protótipos de tais funções devem ser descritos detalhadamente). Nota-se que são feitos quatro testes para cada uma, basicamente mudando-se os valores dos parâmetros de entrada. Os valores das variáveis iniciadas por "`saida__`" serão então comparadas, em memória, pelo *Ortoprog*, com os valores das variáveis de mesmo nome do "Módulo Gabarito". Já a Figura 3b mostra um "*Script* de Testes" para a modalidade de visualização. Observa-se que as variáveis iniciadas por "`saida__`", neste caso, são estruturas que definem o tipo da saída (imagem, gráfico, numérica), a descrição do teste e o valor da variável de teste. Para o primeiro *script* de testes e uso da modalidade de validação, obtém-se as "Saídas" conforme a interface exibida na Figura 4a. A Figura 4b, por sua vez, mostra uma interface de administração do professor. E, por fim, a Figura 4c ilustra o ambiente de interação inicial de submissão dos exercícios e alguns resultados gerados em relação a modalidade de visualização e *script* de testes dado pelo Figura 3b.

O *Ortoprog* foi aplicado em exercícios de processamento de imagens [Silva 2003], demonstrando ser um sistema estável. Dentre as atividades de programação pode-se citar: síntese e manipulação de imagens, histograma, transformações de brilho e contraste, transformações geométricas, convolução, filtros, transformadas (DFT, Wavelets, ...), codificação JPEG, operações em imagens binárias (rotulação, dilatação e erosão). Algumas aplicações dos alunos também foram testadas pela plataforma: contagem de laranja, avaliação de gabarito, detecção de bolacha quebrada, classificação de peças, reconhecimento de partitura musical, detecção de pombo, detecção de cérebro, calibração de microscópio. A página do projeto[4] mantém informações destas atividades e um ambiente de demonstração.

```
saida__1 = mymeshgrid((range(2),range(2),
                       range(3)))
saida__2 = mymeshgrid((range(5),range(4)) )
saida__3 = mymeshgrid((range(5),))
saida__4 = mymeshgrid((range(2),range(1),
                       range(2),range(5)))
saida__5 = mydisk((2,2,3))
saida__6 = mydisk((2,2,3),'chessboard')
saida__7 = mydisk((10,11),'chessboard')
saida__8 = mydisk((10,11),'city-block')
```

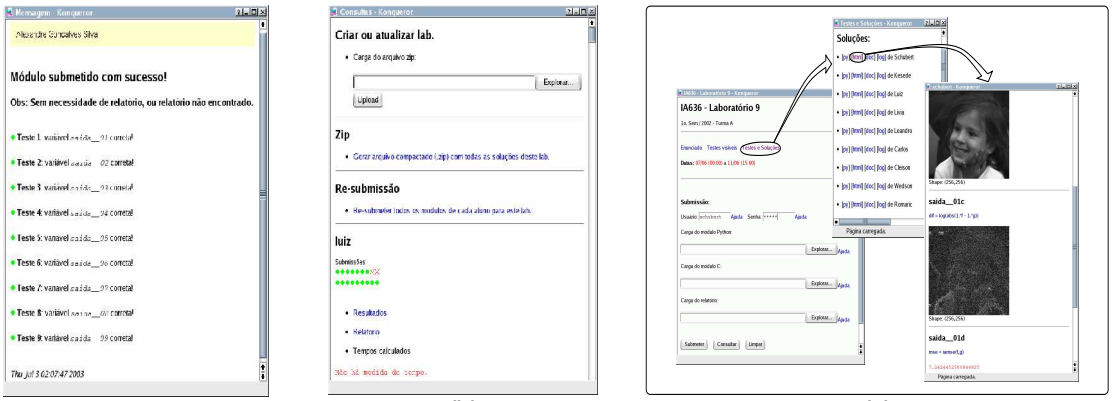
(a)

```
f = iaread('lenina.pgm')
g = iajpegtest(f)
saida__01a = {'tipo' : 'imagem',
              'desc' : 'f = iaread("lenina.pgm")',
              'valor': f}
saida__01b = {'tipo' : 'imagem',
              'desc' : 'g = iajpegtest(f)',
              'valor': g}
dif = Numeric.log(abs(1.*f - 1.*g)+1)
saida__01c = {'tipo' : 'imagem',
              'desc' : 'dif = log(abs(1.*f – 1.*g))',
              'valor': dif}
(mse,psnr,cpearson) = iastat(f,g)
saida__01d = {'tipo' : 'numerico',
              'desc' : 'mse = iamse(f,g)',
              'valor': mse}
saida__01e = {'tipo' : 'numerico',
              'desc' : 'psnr = iapsnr(f,g)',
              'valor': psnr}
saida__01f = {'tipo' : 'numerico',
              'desc' : 'cpearson = iacpearson(f,g)',
              'valor': cpearson}
```

(b)

Figura 3. Exemplos de *scripts* de testes

4 *http://marahu.dca.fee.unicamp.br/ortoprog*



(a) (b) (c)

Figura 4. Exemplos de interfaces do sistema

4. Considerações Finais

Este trabalho propõe uma ferramenta de auxílio na análise de exercícios computacionais. Há trabalhos similiares em outros contextos, como por exemplo, o de verificação automática de projeto de circuitos digitais [Hendrich 2002]. Esta abordagem permite uma interação bastante dinâmica entre alunos, professor e sistema. A grande vantagem da plataforma *Ortoprog*, em relação ao *Susy*, é a possibilidade de verificação das mais variadas e complexas estruturas de dados de forma simples. Pode-se, por exemplo, determinar se uma matriz *n*-dimensional do aluno, resultado de alguma operação matricial, confere com o esperado, e verificar a eficiência do algoritmo implementado, em termos de tempo de execução, para cada teste. Além disto, este trabalho torna possível a criação automática de relatórios ilustrados para cada aluno e permite que o professor, ao entender que os testes feitos foram insuficientes, altere ou acrescente novos testes, re-submetendo-os automaticamente. O *Ortoprog* vem sendo aplicado, a três semestres, nas disciplinas de Visão Computacional e Morfologia Matemática da FEEC/Unicamp com resultados satisfatórios. Como trabalho futuro, poder-se-ia varrer um dado *script* de testes elaborado, variável a variável, de forma a deixar processo ainda mais transparente ao professor. Com o Python é possível identificar o tipo de cada objeto e, desta forma, dar tratamento especializado a cada variável na modalidade de visualização. Neste caso, um *script* de teste comum feito a um módulo qualquer, sem qualquer padronização, poderia ser interpretado pelo *Ortoprog*.

5. Agradecimentos

Aos alunos que cursaram IA636 e IA870, desde o primeiro semestre de 2002, pela contribuição à implementação da plataforma e à FAPESP pelo suporte financeiro (processo N° 00/13671-0).

6. Referências

Hendrich, N. “Automatic Checking of Student's Designs Using Built-In Selftest Methods, in *Proc. of 4th European Workshop on Microelectronics Education – EWME 2002*, May 2002, Baiona, Spain.

Lutz, M., and Ascher, D. “Learning Python”, O'Reilly & Associates, April, 1998.

Pardo, A. “A Platform for Parameterized Exercises in Web-Based Education”, in *Proc. of the Int. Conf. of the Society for Information Technology and Teacher Education 2002*.

Kowaltowski, T. “SuSy – An Automatic Submission and Testing System for Student Programs”. http://colibri.ic.unicamp.br/~tomasz/Susy, February, 2003.

Silva, A., Lotufo, R., Machado, R., and Saúde, A. “Toolbox of Image Processing Using the Python Language”, in *Proc. of the IEEE Int. Conf. on Image Processing 2003*.