Qualifica: Uma Ferramenta para Apoio a Construção de Algoritmos Estruturados

Mauro Marcelo Mattos, Jean Fábio Fuchs

Departamento de Sistemas e Computação – Universidade Regional de Blumenau (FURB)
CEP –89035-160– Blumenau – SC – Brasil

mattos@inf.furb.br, jeanfabiofuchs@hotmail.com

Resumo. *O presente trabalho apresenta um método de desenvolvimento de algoritmos baseado na premissa de que um enunciado bem elaborado e uma análise detalhada deste enunciado por parte do aluno pode contribuir de forma importante no aprendizado de construção de algoritmos. O trabalho descreve também uma ferramenta computacional que foi construída para facilitar o processo de análise dos enunciados. O resultado final é um pseudocódigo desenvolvido pelo aluno..*

1 Introdução

De acordo com Casas (1999), a pedagogia em ciências de educação está baseada em dois princípios: (a) a instrução pode desenvolver as habilidades do aprendiz para que compreenda intuitivamente como funciona o mundo natural em vez de inculcar-lhe a representação formal e as habilidades de raciocínio que os cientistas usam; (b) a instrução que pode ajudar o aprendiz a desenvolver o seu modelo mental (existente) para uma concepção mais exata da realidade.

Segundo as diretrizes curriculares do MEC (MINISTÉRIO DA EDUCAÇÃO, 1999, p.6), "A programação, entendida como programação de computadores, é uma atividade voltada à solução de problemas. Nesse sentido ela está relacionada com uma variada gama de outras atividades como especificação, projeto, validação, modelagem e estruturação de programas e dados, utilizando-se das linguagens de programação propriamente ditas, como ferramentas. Ao contrário do que se apregoava há alguns anos atrás, a atividade de programação deixou de ser uma 'arte' para se tornar uma ciência, envolvendo um conjunto de princípios, técnicas e formalismos que visam à produção de software bem estruturado e confiável. Portanto o estudo de programação não se restringe ao estudo de linguagens de programação. As linguagens de programação constituem-se em uma ferramenta de concretização de software, que representa o resultado da aplicação de uma série de conhecimentos que transformam a especificação da solução de um problema em um programa de computador que efetivamente resolve aquele problema".

Neste contexto Mattos, Fernandes e Lopez (1999) afirmam que, "os estudantes que iniciam um curso de Graduação em Informática, normalmente encontram uma primeira dificuldade relacionada com a disciplina de Introdução a Programação (ou com nome similar), cujo principal objetivo é o de introduzir os conceitos básicos de lógica de programação. Esta dificuldade é, na maioria das vezes, decorrente da falta de experiência com os aspectos relacionados a ambientes industriais e/ou comerciais, pois é

a partir destes ambientes que são caracterizados os exercícios propostos".

Analisando-se o perfil dos alunos, verifica-se que em sua maioria, são oriundos do 2º Grau e, portanto, possuem conhecimentos abstratos sobre áreas científicas (matemática, física, biologia, e outros.). Porém, quando se deparam com a descrição textual dos enunciados dos problemas apresentados nesta disciplina introdutória, geralmente encontram dificuldades em identificar como extrair as informações necessárias para iniciar a solução destes problemas. (MATTOS, 2002; KOVACIC, 2003, p. 794).

Conforme Carvalho e Chiossi (2001, p. 19), quando os requisitos não são totalmente compreendidos, registrados e comunicados para a equipe de desenvolvimento, pode haver discrepância entre o que o sistema construído faz e o que deveria fazer. Estas discrepâncias no contexto de introdução à programação conduzem ao desenvolvimento de soluções erradas (na melhor das hipóteses) ou mesmo à dificuldade no desenvolvimento de uma solução (mesmo que incorreta).

Assim, neste artigo é apresentada uma metodologia de ensino/aprendizagem e uma ferramenta que procura superar as dificuldades apontadas anteriormente estando o texto organizado da seguinte forma: a seção 2 apresenta a contextualização do trabalho; a seção 3 descreve o método de desenvolvimento de algoritmos utilizado; a seção 4 descreve a ferramenta desenvolvida para automatizar o método. Por fim, são apresentados alguns trabalhos correlatos e os resultados obtidos e as limitações da ferramenta proposta.

2 Contextualização do trabalho

O objetivo desse trabalho é apresentar uma metodologia de ensino/aprendizagem para disciplina de introdução a programação que permita ao aluno desenvolver as habilidades necessárias para a interpretação dos enunciados de problemas característicos desta disciplina e também conhecer uma técnica que facilita o processo de construção de soluções algorítmicas. Esta metodologia foi concebida a partir de experiências práticas de sala de aula, e vem sendo desenvolvida na forma de projeto de pesquisa e projetos de conclusão de curso desde 1998 (MATTOS, FERNANDES e LÓPEZ, 1999; MATTOS, 2000; GUBLER, 2002; HEIZEN, 2002; MATTOS, 2002; FREITAS; 2003; FUCHS, 2006).

A intenção principal desse artigo é divulgar essa experiência sem querer afirmar que essa seja uma metodologia ideal para a disciplina, mas sim, uma proposta que foi colocada em experiência e que trouxe bons resultados. Foi possível constatar uma melhora no aproveitamento e um aumento no grau de interesse e satisfação dos alunos no decorrer do curso.

3 O método

Nesta seção é apresentado o método de ensino de introdução a programação desenvolvido pelo Prof. Mauro Mattos (Mattos, 2005). O método está baseado em duas premissas: a construção de enunciados detalhados contendo exemplos de entradas e saídas esperadas e, um processo de análise detalhada dos dados de entrada e de saída. Tendo em vista facilitar a explicação, será utilizado um dos enunciados de problemas de introdução a programação utilizados em aula (Quadro 1). A partir deste enunciado o

aluno deve construir uma tabela conforme apresentado na Figura 1. Nesta tabela o aluno registra, da esquerda para a direita e, de cima para baixo, a seqüência com que os dados de exemplo são entrados no sistema e qual a saída esperada para aquela entrada.

Observe-se que, cada entrada é registrada em uma linha específica (na coluna das entradas) e as saídas são registradas uma linha abaixo da última entrada (na coluna correspondente às saídas).

- Enunciado: Ler e imprimir um conjunto de dados contendo nome e uma nota referente a um grupo de n alunos. Parar a leitura quando o nome digitado for = "fim".
  - **Exemplo de dados:**

aluno:Daniel Ana Elvis Joao Maria fim
Nota : 7 6 5 10 6 *

Quadro 1 - Exemplo de enunciado de problema

A partir do momento em que o aluno informou todas as entradas e saídas, o próximo passo é qualificar as informações fornecidas identificando um nome para a variável que será utilizada para armazenar aquele valor entrado ou saído (quando for o caso) e um tipo (inteiro, caractere, lógico, string). A Figura 2 representa esta operação.

Figura 1 – Relacionando saídas após as entradas

Tendo sido qualificadas as variáveis de entrada e saída o próximo passo é identificar se os nomes das variáveis estão repetidos. Por exemplo, na Figura 2 os nomes de variáveis "nome" e "nota" repetem-se várias vezes. Este exemplo permite algumas reflexões, quais sejam:

a) Se o aluno está aprendendo a utilizar estruturas de repetição (*while*, *repeat*, *for*), geralmente no início do semestre, então, a constatação de que os nomes das variáveis repetem-se indicam uma situação objetiva da necessidade do emprego deste tipo de construção;

b) Se o aluno já conhece as estruturas de repetição, então a constatação da repetição pode facilitar a introdução do conceito de estruturas de armazenamento do tipo matrizes;

c) Uma vez que o aluno tenha assimilado o conceito de matrizes para armazenamento de dados homogêneos, o próximo passo é a possibilidade da introdução do conceito de registros, pelo agrupamento de nomes de variáveis repetidas.

Entradas	Processamento	Saídas
Nome: Daniel		
Nota: 7		
		Nome: Daniel **Nota**: 7
Nome: Ana		
Nota: 6		
		Nome: Ana **Nota**: 6
Nome: Elvis		
Nota: 5		
		Nome: Elvis **Nota**: 5
Nome: João		
Nota: 10		
		Nome: João **Nota**: 10
Nome: Maria		
Nota: 6		
		Nome: Maria **Nota**: 6
Nome: fim		

Figura 2 – Identificação dos nomes das variáveis

O próximo passo consiste em identificar como os dados de saída são produzidos a partir do fornecimento dos dados de entrada. Supondo-se que o enunciado propusesse a emissão de um relatório contendo o nome e a nota dos alunos cuja nota é maior que 6, a coluna processamento seria utilizada para registrar a necessidade de uma operação de teste para filtrar a saída. A Figura 3 indica que, quando foram informados os dados: "nome: Elvis" e "nota: 5", não houve saída porque a nota ficou abaixo do limiar estabelecido no enunciado. Portanto, não foi produzida uma saída para aquela entrada.

Entradas	Processamento	Saídas
Nome: String: Daniel		
Nota: inteiro: 7		
	> 6, produz saída	
		Aluno: Daniel Nota: 7
Nome: String: Ana		
Nota: inteiro: 6		
	> 6, produz saída	
		Aluno:Ana Nota: 6
Nome: String: Elvis		
Nota: inteiro: 5		
	< 6, não produz saída	
Nome: String: João		
Nota: inteiro: 10		
	> 6, produz saída	
		Aluno:João Nota: 10
Nome: String: Maria		
Nota: inteiro: 6		
	6 não é maior que 6, portanto, não produz saída	
Nome: String: fim		

Figura 3 – Anotação de necessidade de manipulação dos dados de entrada

Outro aspecto que pode ser identificado a partir da análise dos dados de entrada é a condição de que a nota de “Maria” é “6” e que, o primeiro esboço de solução pode fazer com que o nome “Maria” não seja impresso tendo em vista que não foi considerada a condição de “maior ou igual” (embora este possa ser um requisito da especificação do problema). Ou seja, a partir da análise dos dados, o aluno é conduzido a refinar as suas proposições lógicas no sentido de equacionar o problema proposto.

Numa análise mais detalhada, o aluno pode perceber que o processo de repetição das entradas produzindo as saídas implicaria na repetição do conjunto de linhas apresentado na Figura 4a. Isto não seria uma solução aceitável do ponto de vista computacional. Além disso, o destaque para a condição de parada (nome do aluno = “fim”) poderia levar o aluno a produzir uma solução conforme apresentado na Figura 4b.

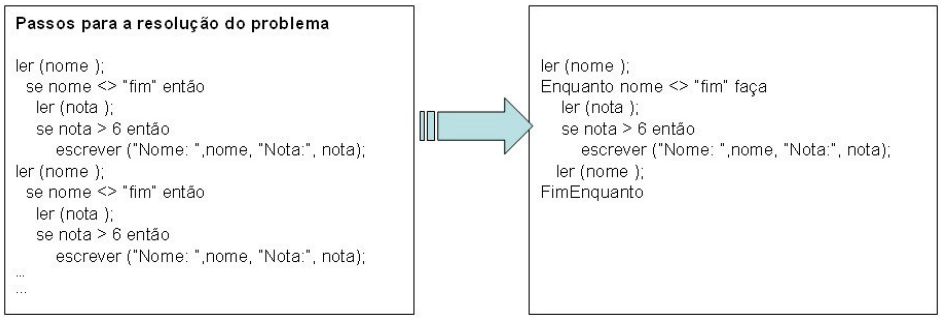

Figura 4– a) exemplo de um pseudocódigo extraído a partir do detalhamento da planilha; b) exemplo de um refinamento de um bloco de repetição.

Este processo de fazer com que o aluno detalhe as entradas e saídas, posicionando-as de forma deslocada nas linhas da tabela conduzem-no a pensar em termos temporais – requisito implícito na construção de soluções algorítmicas. Com isto o aluno tem facilidade em identificar:

a) O que ocorre antes do que (exemplo entra-se primeiro o nome depois a nota);

b) O que ocorre depois do que (exemplo: entra-se primeiro o nome e a nota e depois se exibe no nome e nota).

4 A ferramenta desenvolvida

A partir da aplicação prática do método acima descrito, propôs-se a construção de uma ferramenta que automatizasse o processo. O sistema foi concebido na forma de dois módulos: um módulo professor e um módulo aluno. O módulo professor permite que o professor configure um exercício enquanto o módulo aluno permite ao aluno solucionar o problema proposto (FUCHS, 2006).

A Figura 5 mostra um exemplo da tela principal do programa com o editor do enunciado do problema e a definição das estruturas da base de dados, com suas tabelas, atributos e registros. Nesta figura é possível identificar: (1) configuração do nível de complexidade do exercício, no caso aqui é intermediário; (2), exibe a lista de atributos da tabela atual (Curso); (3), exibe a lista de tabelas do exercício, ficando a tabela atual (Curso) em destaque, exibindo detalhadamente o seu *alias* e a quantidade de atributos e registros; (4) mostra a fase atual da criação do exercício, que no caso aqui o professor

está Definindo a Base de Dados e Registros; (5) um botão que permite o professor ir para a próxima fase de criação do exercício que é a Formatação de Saída; (6) editor de textos aonde o professor deverá descrever o enunciado do exercício; (7) mostra a lista de registros de entrada da tabela atual (Curso) do exercício.

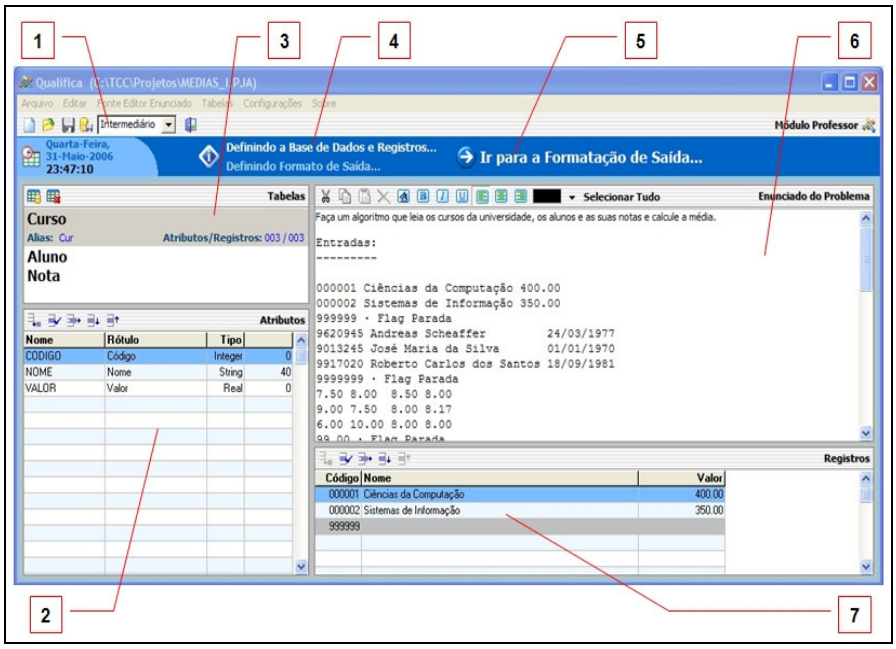

Figura 5– Tela principal do módulo professor.

A Figura 6 apresenta alguns dos recursos disponíveis ao professor na configuração das tabelas de exemplos a serem utilizados pelos alunos na solução do problema proposto.

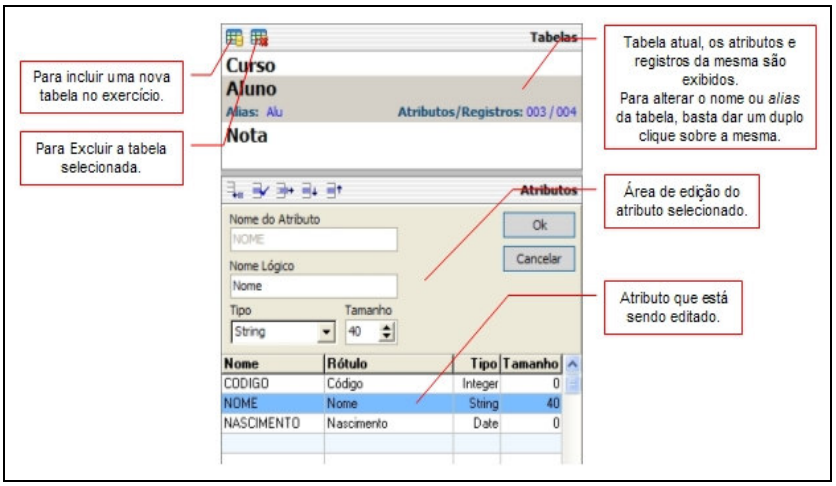

Figura 6 – Recursos para criação das tabelas de exemplos.

A Figura 7 apresenta um diagrama de atividades do módulo professor e a Figura 8 apresenta a tela do módulo aluno. Nesta figura é possível identificar: (1) botões para avançar e retornar as fases da solução do exercício, o aluno só poderá ir adiante na

solução depois de concluir a fase atual; (2) descreve a fase em que o exercício se encontra no momento: “Classificando as Entradas e Saídas...”, é fase em que o aluno terá que descobrir a seqüência de leitura das entradas e as saídas; (3) os registros de entradas que foram selecionados, tem a aparência de um botão pressionado; (4) um registro de entrada que está em destaque, quando o *mouse* é posicionado, a mesma se torna intermitente; (5) as entradas que ainda não foram selecionadas; (6) o botão para visualizar o formato de saída em formato texto; (7) os itens do formato de saída, nas cores iguais a grade significam que já foram selecionados; (8) os itens do formato de saída que ainda não foram selecionados; (9) o tempo total que o aluno utilizou para resolver o exercício, a partir da primeira alteração em qualquer parte da tela o tempo é iniciado ou reiniciado caso o aluno esteja dando continuidade a um exercício que foi parcialmente resolvido; (10) uma mensagem exibindo qual o próximo passo que o aluno deve tomar, no caso, qual o próximo registro de entrada que deve ser selecionado; (11) exibe o nível do exercício, no caso aqui é intermediário; (12) mensagens informando quais fases da solução do exercício que o aluno já completou; (13) exibe a grade de classificação das entradas, aqui especificamente uma entrada que possui saídas vinculadas a ela; (14) a grade de classificação das saídas, caso a entrada tenha alguma saída vinculada é então exibida.

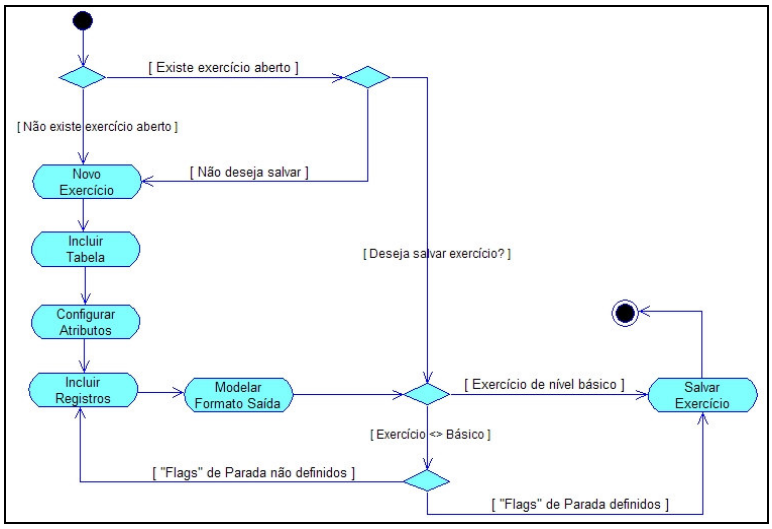

Figura 7 – Diagrama de atividades do módulo professor.

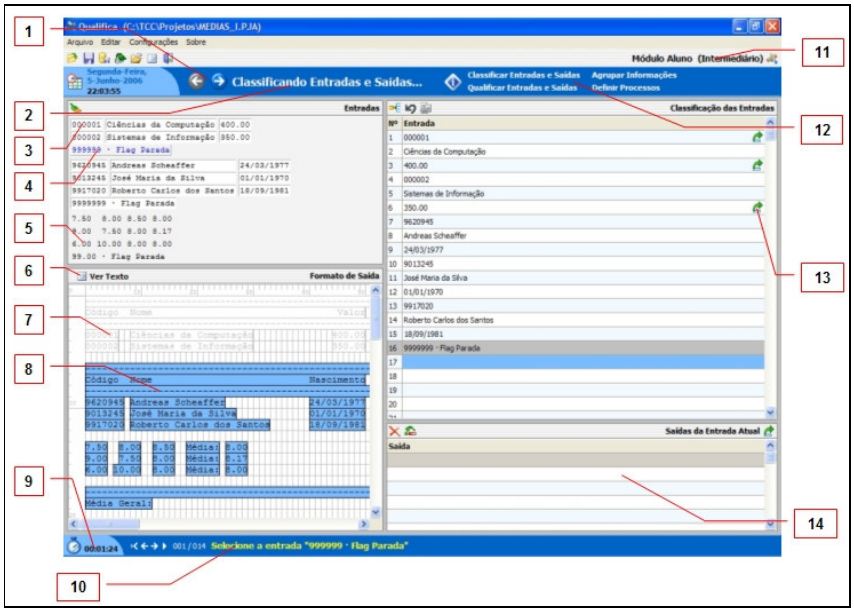

Figura 8 – Tela do modulo do aluno.

O conceito de agrupamento de entradas diz respeito ao processo repetitivo de entrada de dados em um programa que é associado a um campo (ou a vários campos de um registro ou de um *array*) até que uma condição de fim seja detectada.

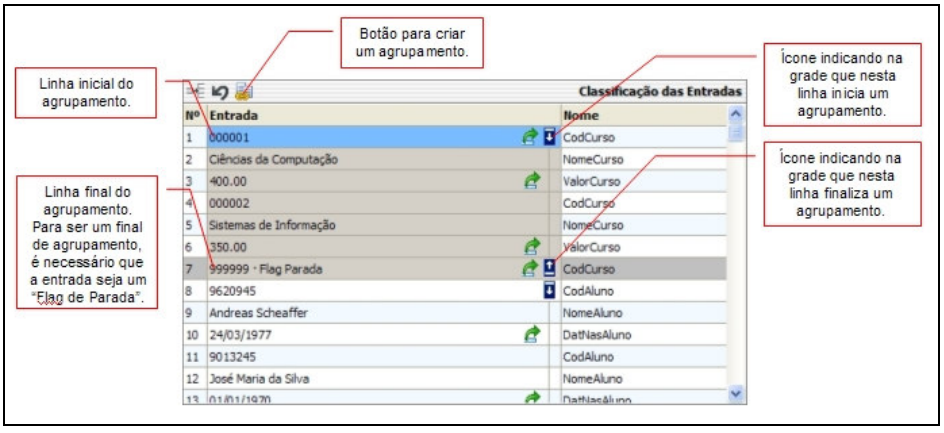

Figura 9 – Agrupamento informações

Quando o processo de associação entre entradas e saídas é encerrado, a janela que contém estas informações passa a ficar oculta e é desabilitada a possibilidade de qualquer alteração nas mesmas. Contudo, o aluno pode consultá-las clicando no botão localizado na parte superior da barra lateral. O processo de agrupamento (Figura 9) consiste em clicar na linha da entrada inicial e arrastar o mouse até a linha da entrada final (que deve ser uma entrada definida no módulo professor como "Flag de Parada").

Depois de marcado o bloco, basta clicar no botão "Agrupar" (Figura 9) e a tela de cadastramento do agrupamento (Figura 10) irá aparecer para que o aluno informe o nome do agrupamento, tipo do agrupamento – "procedimento" ou "laço de repetição", tipo do laço de repetição e uma dica ou descrição sobre o que este agrupamento trata,

esta descrição irá aparecer no código fonte no Portugol. Além disso, o sistema apresenta o intervalo das linhas inicial e final que foram selecionadas na grade de entradas (Figura 9) e que resultarão no novo agrupamento.

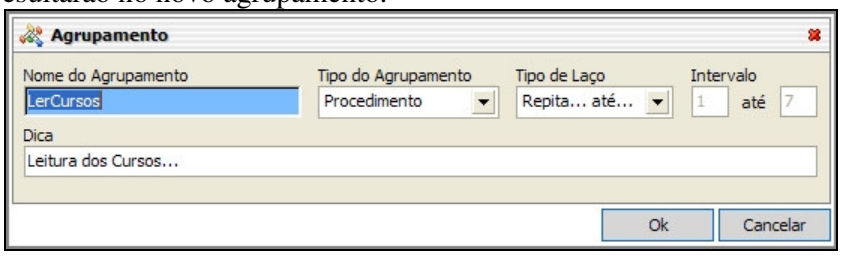

Figura 10 – Tela de cadastramento de agrupamentos

A Figura 11 apresenta o diagrama de atividades que estão envolvidas no processo de solução de um exercício no ambiente da ferramenta construída.

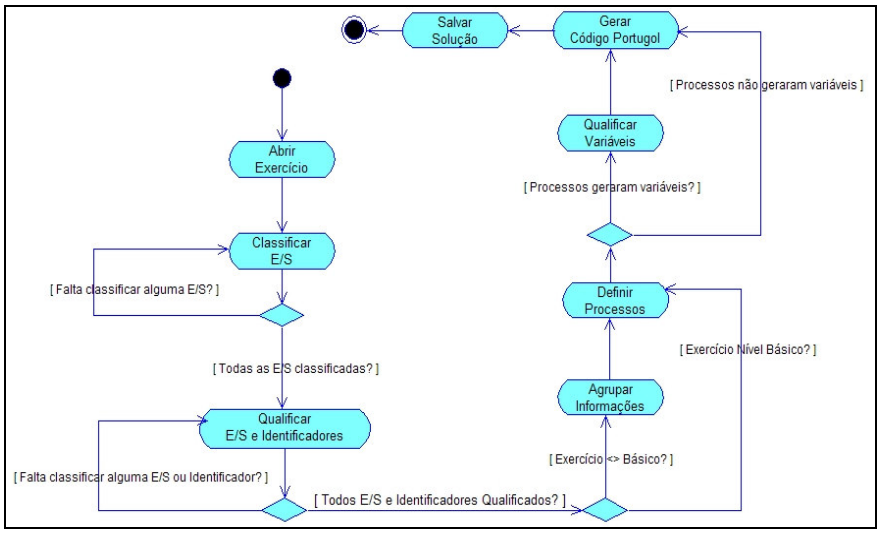

Figura 11 – Diagrama de atividades do processo de solução de um exercício pelo aluno

Tendo em vista conduzir o aluno no processo de aprendizagem, existe uma barra de informações que mantém o aluno posicionado em que fase ele está (Figura 12). Um aspecto importante a destacar é a possibilidade que o aluno tem de navegar entre as fases voltando atrás ou avançando até o ponto em que ele conseguiu chegar na solução do problema.

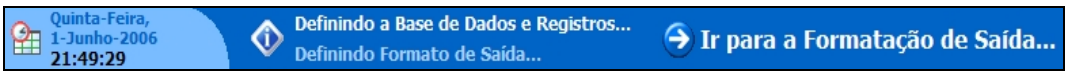

Figura 12 – Barra de informações para o aluno.

Uma funcionalidade importante é que na medida em que o aluno vai "consumindo" os dados de exemplo cadastrados pelo professor, os mesmos vão sendo marcados como já utilizados. A Figura 13 representa a situação em que o aluno utilizou todos os dados de exemplo do exercício.

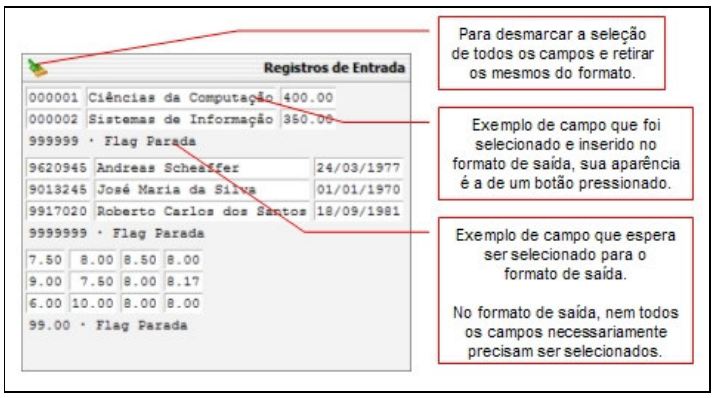

Figura 13 – Dados de entrada totalmente utilizados pelo aluno.

Da mesma forma, os dados do relatório de saída devem ser completamente "consumidos" pelo aluno à medida que as entradas vão sendo informadas (Figura 14)

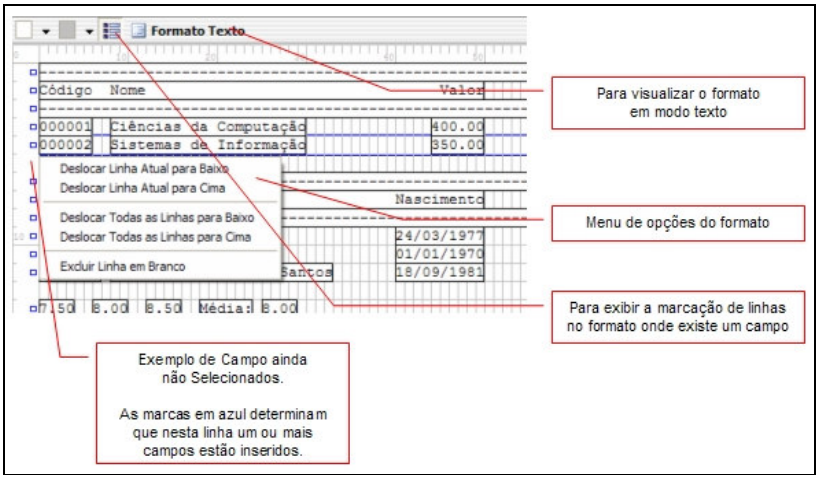

Figura 14 – Dados de saída totalmente utilizados pelo aluno.

A Figura 15 apresenta um apanhado de telas do sistema caracterizando o módulo professor sendo utilizado na construção de uma especificação detalhada do problema e telas do módulo aluno caracterizando a resolução do mesmo até a geração de um pseudocódigo da solução proposta. Maiores detalhes sobre a arquitetura do sistema podem ser obtidos em Fuchs (2006).

5 Trabalhos correlatos

A questão de ensino de algoritmos e lógica de programação é um tema recorrente. De acordo com Santiago e Dazzi (2003), relacionam, entre outros os trabalhos de Brown (1987,1988), Stubbs e Webre (1988), Stasko (1990), Brown (1991), Amorim e Rezende (1993), Szwarcfiter e Markenson (1994), Cares (2002), Medeiros (2001) e Mendes e Gomes (2000). Além disso, podemos citar: Gloor (1992), Esmin (1998), Colleen,Stasko e Ashley (1999), Heinzen (2002), Gubler (2002), Silveira e Esmin (2003), Santiago e Dazzi (2003) e Pereira Júnior e Rapkiewicz (2006), entre outros.

Segundo Pereira Júnior e Rapkiewicz (2004) apud Ferrandin e Stephani(2005), a análise de 105 artigos sobre o tema mostra que há três vertentes na busca de soluções

para os problemas do processo: Ferramentas, Estratégias e a união de ambas. Esta análise parece sugerir que a união de ferramentas computacionais e estratégias têm se demonstrado como melhor proposta. O presente trabalho associa uma estratégia (refinamentos sucessivos) a uma ferramenta computacional como forma de auxiliar no processo de ensino-aprendizagem.

6 Resultados e limitações

O presente trabalho apresentou os fundamentos para a utilização de técnicas de desenvolvimento estruturado de programas na construção de uma ferramenta de apoio ao ensino de desenvolvimento de algoritmos. Como referido no texto, o projeto atual está inserido no contexto de um projeto mais amplo de desenvolvimento de uma ferramenta para ensino de algoritmos Conforme apresentado, a filosofia sobre a qual o sistema foi construído está baseada em dois conceitos principais: (i) uma especificação detalhada do problema por parte do professor e (ii) uma análise detalhada da especificação por parte do aluno. Esta análise detalhada é conduzida pela ferramenta de tal forma que, após verificar todos os elementos da especificação o aluno obtém um pseudocódigo da solução algorítmica do problema a ser desenvolvido.

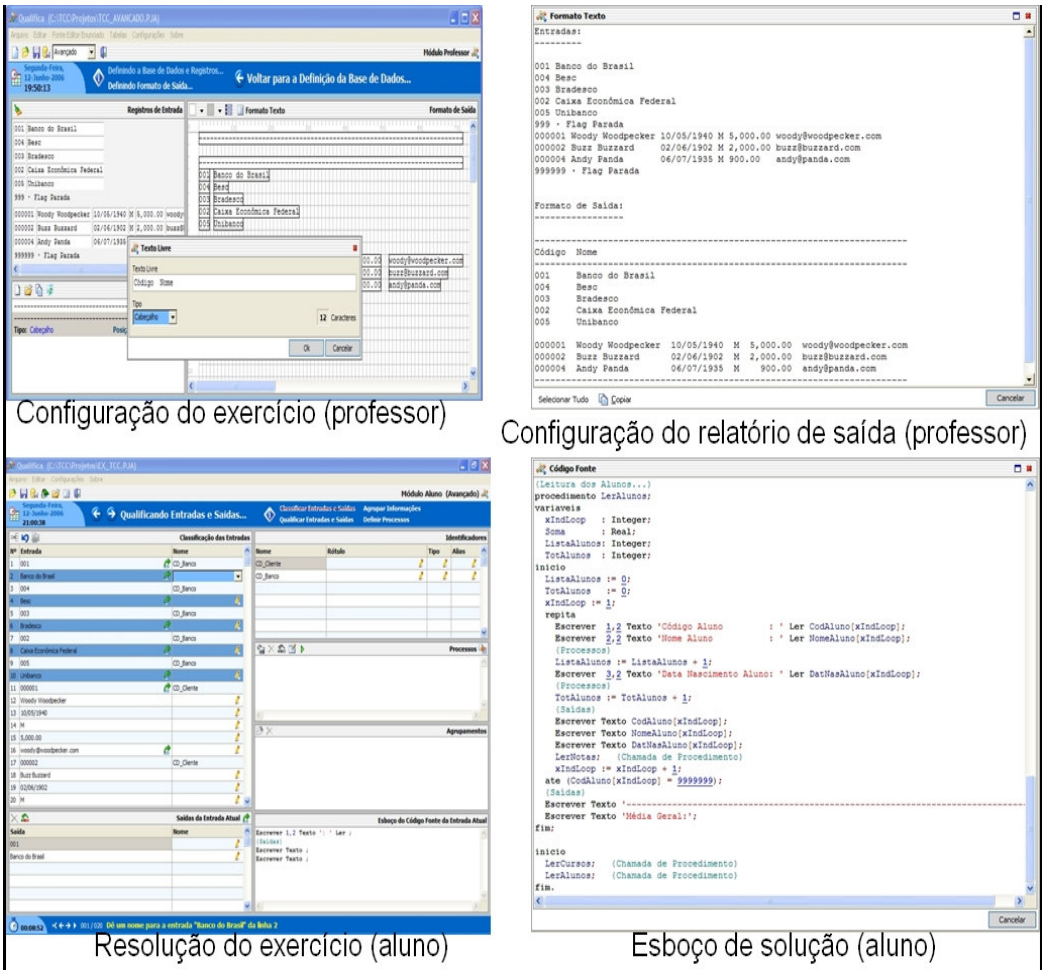

Figura 15 – Telas do modo professor e aluno

O método apresentado foi aplicado em turmas de primeiro e segundo semestre durante o período de 2004/1 e 2004/2. A avaliação dos acadêmicos foi que a estratégia de refinamento atuou como um facilitador no processo de compreensão do enunciado de um problema na medida em que auxiliou os acadêmicos na organização das idéias de como construir uma solução algorítmica para os problemas propostos. A ferramenta descrita foi finalizada em jul/2006 sendo esperada a sua utilização para validação no segundo semestre de 2007.

Referências

AMORIM, R. V.; REZENDE, P. J. Compreensão de Algoritmos através de Ambientes Dedicados a Animação. In: SEMISH, 10., 1993.

BROWN, M. H. Algorithm Animation. The MIT Press, 1987.

BROWN, M. H. Exploring Algorithms Using Balsa-II. Computer, maio 1988. p. 14-36.

BROWN, M. H. Zeus: A System for Algorithm Animation and Multi-View Editing. Procedings...IEEE Workshop on Visual Languages, 1991.

CARES, P. L. L. Ambiente para teste de mesa utilizando fluxograma. TCC(Graduação)– Faculdade de Ciência da Computação, Universidade do Vale do Itajaí, Itajaí, 2002.

CARVALHO, A.M.B.R.; CHIOSSI, T.C.S. Introdução à engenharia de software. São Paulo: Unicamp, 2001.

CASAS, A.A. Contribuições para modelagem de um ambiente inteligente de educação baseado em realidade virtual. Florianópolis, 1999. Paginação irregular. Disponível em: <http://www.eps.ufsc.br/testes99/casas/>. Acesso em: 01 jun. 2004.

COLLEEN, K, STASKO, J.;ASHLEY, T. Rethinking the evaluation of algorithm animations as learning aids: an observational study. Graphics, Visualization, and Usability Center, Georgia Institute of Technology, Atlanta, GA, Technical Report GIT -GVU-99-10, March 1999.

ESMIN, A. A. A. . Portugol/Plus : Uma Ferramenta de Apoio ao Ensino da Lógica de Programação baseado no Portugol. In: IV Congresso Ibero-americano de Informática Educativa. Brasília, 1998, Brasilia/DF. Anais do IV Congresso Ibero-americano de Informática Educativa. Brasília, 1998.

FERRANDIN, M.;STEPHANI, S.L. Ferramenta para o ensino de Programação via Internet1 . In: SULCOMP - I Congresso Sul Catarinense de Computação, 2005, Criciúma. Anais do Sulcomp.

FREITAS, G. Protótipo de um sistema de animação de algoritmos. 2003. 60 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

FUCHS, J.F. Qualifica: uma ferramenta de suporte ao desenvolvimento de algoritmos. 2006. 107 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

GLOOR, P.A. AACE – algorithm animation for computer science education. In Proceedings of the 1992 IEEE Workshop on Visual Languages, pages 25-31,Seattle, WA, September 1992.

GUBLER, A.I. Protótipo de um sistema especialista para auxiliar no ensino de algoritmos. 2002. 55 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

HEINZEN, L.A. Módulo de raciocínio baseado em casos em uma ferramenta de apoio ao ensino de lógica de programação. 2002. 62 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

KOVACIC, Z. J. A comparison of learning and teaching styles. In: InSITE 2004 – Informing Science and Information Technology Education 4., 2004, Rockhampton, Australia. Proceedings… Santa Rosa, California, 2004. Paginação irregular. Disponível em: <http://proceedings.informingscience.org/InSITE2004/105kovac.pdf>. Acesso em: 1 abr. 2006.

MATTOS, M.M.; FERNANDES, A.; LÓPEZ, O.C. Sistema especialista para apoio ao aprendizado de lógica de programação. In: Congresso Ibero-americano de Educação Superior em Computação, 7., 1999, Florianópolis. Anais... Assunção: Universidad Autônoma de Asunción, 1999. p. 1-12.

MATTOS, Mauro. M. Construção de abstrações em lógica de programação. In: SBC 2000, 10., 2000, Curitiba. Anais... Curitiba: Editora Universitária Champagnat, 2000a. p. 60-60.

MATTOS, M.M. Learning how to build abstractions in programming logics classes. In: IE 2002 – CONGRESSO IBEROAMERICANO DE INFORMATICA, 6., 2002, Vigo, Espanha. Proceedings... Vigo, Espanha, 2002. Paginação irregular.

MATTOS, M.M. Linguagens para programação de sistemas. 2005. Paginação irregular. Notas de aula (Disciplina de Linguagens para Programação de Sistemas, Curso de Ciências da Computação). Depto de Sistemas e Computação, Universidade Regional de Blumenau, Blumenau.

MEDEIROS, C.L.; DAZZI, R.L.S. Aprendendo Algoritmos com Auxílio da Web. II CONGRESSO BRASILEIRO DE COMPUTAÇÃO, 2., 2002, Itajaí. Anais... Itajaí: UNIVALI, CTTMar, 2002.

MENDES, A. J. N.; GOMES, A. J. Suporte a aprendizagem de programação com o ambiente SICAS. In: Congresso Ibero Americano de Informática Educativa - RIBIE,5., 2000, Viña del Mar-Chile. Anais... Viña del Mar-Chile: Universidad de Chile, 2000.

MINISTÉRIO DA EDUCAÇÃO. Diretrizes curriculares de cursos da área de computação e informática. Brasília, 1999. Disponível em: <http://www.mec.gov.br/sesu/ftp/curdiretriz /computacao/co_diretriz.rtf>. Acesso em: 1 jun. 2005.

PEREIRA JR, J.C.R., RAPKIEWICZ, C. E. “Um Ambiente Virtual para apoio a uma Metodologia para Ensino de Algoritmos e Programação”. RENOTE. Revista Novas Tecnologias na Educação, v. 3, 2005.

SILVEIRA, I.J.; ESMIN, A. A. A. . AVA - Um Ambinete Visual para a Construção de Algoritmos. In: ICECE'2003 International Conference on Engineering and Computer Education, 2003, , São Vicente / Santos. Anais ICECE, 2003.

SANTIAGO, R.; DAZZI, R. L. S. . Ferramentas que auxiliam o desenvolvimento da lógica de programação.. In: XII SEMINCO - Seminário de Computação, 2003, Blumenau. Anais do XII SEMINCO. Blumenau : Furb, 2003. p. 113-120.

STASKO, J. T. Tango: A Framework and System for Algorithm Animation. Computer, setembro 1990. p. 14-36.

STUBBS, D. F.; WEBRE, N. W. Data Structures with Abstract Data Types and Pascal, Pacific Grove, Brooks/Cole, 2 ed., 1988.

SZWARCFITER, J.; MARKENSON, L. Estruturas de Dados e seus Algoritmos, LTC, 1994.