AVALIAÇÃO DO ENSINO-APRENDIZAGEM DE PROGRAMAÇÃO USANDO UMA ABORDAGEM BASEADA EM PADRÕES ELEMENTARES DE PROGRAMAÇÃO

Marilza Antunes de Lemos,[a] Roseli de Deus Lopes[b]

RESUMO

As dificuldades que estudantes encontram no aprendizado de linguagens de programação têm sido relatadas em diferentes épocas e em diferentes áreas de pesquisa. Este artigo baseia-se na proposta de um modelo de processo de construção de programas para aprendizes, o qual foi concebido para apoiar a construção de sistemas que minimizem alguns dos reconhecidos problemas enfrentados por programadores novatos. A partir desse modelo foi implementado um sistema tutor para aprendizado de programação em linguagem C, o qual foi experimentalmente utilizado com um grupo de alunos de engenharia elétrica. Este artigo relata e discute os resultados desse experimento, que apontam para possibilidades promissoras de aplicação desta abordagem.

***Palavras-chave**:* Modelos mentais. Padrões de programação. Sistema tutor.

ABSTRACT

The difficulties faced by students to learn programming languages have been noticed and described by several researchers from different areas. This paper is based on a proposed model named Program Construction Process for Trainees. It was developed in order to enable the construction of systems that can minimize some of the known problems faced by naïve programming students. Following this model, it was developed a tutor system for the learning process of programming using the C language, named TutorC. This tutor system was experimentally used by a group of Electrical Engineering students. This paper presents the evaluation results about the use of this cognitive approach, which shows some positive improvements on the learning process of programming when this approach is applied.

***Key words**:* Introdução. Mental models. Programming patterns. Tutoring system.

INTRODUÇÃO

As dificuldades que estudantes encontram no aprendizado de linguagens de programação têm sido relatadas em diferentes épocas e áreas de pesquisa, tais como educação, psicologia da programação (uma subárea da ciência cognitiva), sistemas tutores inteligentes da inteligência artificial (SHUTE; PSOTKA, 1994) e padrões de programação da ciência da computação (HILLSIDE. NET, 2003). Estudos empíricos da psicologia da programação realizados na década de 1980 por Soloway e Ehrlich (1984), Bonar e Soloway (1983) provaram que programadores experientes agrupam fragmentos de código (*chunks*) criando estruturas denominadas "planos mentais de programação", as quais condensam ações primitivas de programação. Os planos mentais de programação são armazenados na base de conhecimento interna do programador ao longo de sua experiência (WALLINGFORD, 1998). Dessa forma, programadores experientes conseguem mapear metas de um problema diretamente nesses planos de programação. Gellen-

[a] Professora Doutora, Universidade Estadual Paulista, Depto. de Engenharia de Controle e Automação, Campus de Sorocaba, Av. Três de Março, 511; CEP 18087180; Fone (15) 3238-3411. E-mail: marilza@sorocaba.unesp.br.

[b] Professora Doutora, Universidade de São Paulo, Depto. de Engenharia de Sistemas Eletrônicos, Escola Politécnica, Av. Prof. Luciano Gualberto, travessa 3, nº 158; CEP- 05508-900– São Paulo. -E-mail: roseli.lopes@poli.usp.br

beck, Cook e Wiedenbeck (apud PANE; MYERS, 1996) afirmam que aprendizes não conseguem fazer uso dessa técnica porque não aprenderam, ainda, a relacionar tais estruturas de código às ações de alto nível. Assim, os problemas mais proeminentes no aprendizado de programação parecem estar situados ao longo de dimensões cognitivas.

Considerando idéias e iniciativas das diferentes áreas de pesquisa envolvidas, foi concebido um modelo de processo de construção de programas para aprendizes (LEMOS, 2004) para apoiar a construção de sistemas que minimizem alguns dos principais e reconhecidos problemas enfrentados por programadores novatos. O modelo é proposto como uma ferramenta cognitiva capaz de auxiliar aprendizes a executarem atividades mentais de programação que não podem ser observadas diretamente ou que podem ser pouco observáveis. Baseado nesse modelo cognitivo, foi implementado o sistema TutorC (LEMOS et al., 2003a) para aprendizado de programação em linguagem C, utilizado, experimentalmente, com um grupo de alunos da disciplina de Introdução à Computação no curso de Engenharia Elétrica de uma instituição de ensino superior.

Este artigo relata os resultados e avaliações desse experimento, que indicam caminhos promissores para aplicação desta abordagem. A seção 2 define a abordagem cognitiva e sua relação com padrões de programação; a seção 3 descreve o modelo do processo de construção de programas para aprendizes em programação e sua implementação no sistema TutorC; a seção 4 descreve a metodologia de avaliação da abordagem proposta; a seção 5 apresenta e discute os resultados obtidos e, finalmente, a seção 6 relata as conclusões.

ABORDAGEM TRADICIONAL X ABORDAGEM COGNITIVA

Tradicionalmente, enquanto um professor em sala de aula explica a construção ou compreensão de um programa, ele também: (i) identifica objetivos a serem alcançados, extraídos a partir do enunciado; (ii) relembra conceitos da linguagem de programação; (iii) mostra o mapeamento entre objetivos e estruturas da linguagem; (iv) fornece o significado das linhas de código utilizadas no programa; (v) explica instanciações do problema no código, entre outras atividades. Porém, todo esse conhecimento é transmitido ao aluno de maneira informal, ou seja, não é documentado como, nessa situação, a carga cognitiva exigida do estudante é muito grande, ele acaba por não assimilar todo o conhecimento com sucesso. A abordagem cognitiva avaliada neste trabalho formaliza alguns tipos de conhecimento manipulados na abordagem tradicional: identificação das metas do problema e sua decomposição em submetas, planos e ações primitivas de programação, mapeamentos entre metas e planos, mapeamentos entre dados do problema e ações primitivas de programação. Em resumo, a diferença entre a abordagem tradicional e a proposta neste trabalho reside no aspecto da formalização e documentação dos tipos de conhecimento necessários à atividade de programação. Neste artigo faz-se referência a duas abordagens, com os termos abordagem tradicional e abordagem cognitiva.

FUNDAMENTOS DA ABORDAGEM COGNITIVA

A ciência cognitiva, em particular a área de psicologia da programação, divulgada pelo Grupo de Interesse da Psicologia da Programação (PPIG) (PPIG, 1987), preocupa-se com o desenvolvimento de teorias sobre como programadores experientes e como aprendizes compreendem e constroem programas. Estudos empíricos têm resultado em propostas de modelos mentais de programadores e de como diferenciar programadores experientes de aprendizes (VON MAYRHAUSER; VANS, 1994). A modelagem cognitiva leva em conta a natureza do conhecimento. Self (1995) afirma que muitos domínios têm sido modelados com conhecimento objetivo, entretanto logo se torna visível que os problemas reais de aprendizado não residem no conhecimento objetivo, mas na sua relação com o conhecimento menos objetivo.

No domínio da programação o foco move-se da sintaxe da linguagem para aspectos de projeto de programação. A programação é considerada uma atividade cognitiva; portanto, a maioria dos estudos empíricos da literatura descreve a análise de tarefa cognitiva (CTA - *Cognitive Task Analysis*) em programação. Este tipo de análise é um processo sistemático pelo qual os elementos cognitivos são identificados durante o desempenho da tarefa. Assim, análise de tarefa cognitiva foca atividades mentais que não podem ser observadas diretamente ou que podem ser pouco observáveis. A abordagem proposta neste trabalho é denominada “cognitiva”

porque explora resultados provenientes da área da ciência cognitiva para criar ferramentas para aprendizado de programação. A teoria é aplicada ao problema da dificuldade de construção do modelo mental de programa pelo aprendiz em programação.

PADRÕES PEDAGÓGICOS PARA PROGRAMAÇÃO

Do ponto de vista cognitivo, a programação é freqüentemente definida como um processo de transformar um plano que está numa forma familiar (plano natural) num plano que seja compatível com o computador (plano de programação ou programa). Na execução desse processo muitas dificuldades se originam porque existe uma grande distância entre os dois planos. A abordagem deste trabalho tem a proposta de disponibilizar componentes cognitivos para aprendizes enquanto executam a atividade de construção de programas. Assim, *chunks* de programação são elementos de interesse para apoiar o aprendiz a transpor a distância entre planos naturais e o programa a ser construído. Embora esses termos sejam muito citados nas áreas da psicologia da programação e de sistemas tutores inteligentes para programação, não constam trabalhos de modelagem desses elementos nessas áreas de pesquisa.

O uso de planos de programação surgiu, inicialmente, no desenvolvimento dos primeiros sistemas tutores inteligentes (BONAR; CUNNINGHAM, 1988; JOHNSON; SOLOWAY, 1985), porém, por terem sido usados como conhecimento interno ao sistema, não para uso explícito pelo aprendiz, não constam especificações de tais planos na literatura. Uma comunidade nova, interessada em documentar conhecimento especialista em programação, é a comunidade de projeto de padrões pedagógicos para programação (BERGIN, 2001; ECKSTEIN et al., 2001; WALLINGFORD, 1998; ASTRACHAN; WALLINGFORD, 1998). Pesquisadores desta recente área de pesquisa propõem uma forma de documentar tal conhecimento sob a denominação de padrões de programação.

Quando um programador se torna um especialista, seu conhecimento de programação se torna independente da sintaxe de linguagens de programação. Um especialista usa um vocabulário de alto nível para descrever um problema e sua solução. Esse vocabulário é formado por sentenças, tais como laço de busca, ação alternativa, escolha seqüencial, as quais permitem que o especialista pense sobre o planejamento do programa e represente uma solução num alto nível de abstração. As sentenças estão associadas a construções de código, denominados "idiomas" (*idioms*) na comunidade de padrões para programação, que o programador especialista conhece. O aprendizado dessas novas construções e sua relação com termos de alto nível são passos importantes para novatos se tornarem especialistas (ANDERSON, 1995).

Padrões de programação definem conceitos de alto nível relacionados com construções de código, e descrevem a aplicabilidade da construção de código num contexto. Padrões, de um modo geral, tentam encapsular conhecimento especialista numa forma acessível e utilizável. No domínio da programação, padrões são como blocos de construção para projeto e construção de *software* (COAD, 1992). Wallingford (2002) define um padrão pedagógico para programação, em sua forma mais simples, como uma regra constituída de três partes:

(1) um contexto que descreve um conjunto de situações nas quais o padrão se aplica;
(2) um problema que é descrito por um conjunto de requisitos e metas a serem alcançadas dentro do contexto;
(3) uma solução para o problema, que se constitui de um texto explicativo de como obter uma configuração de *software* que resolve o problema. Um padrão explica ainda por que essa solução é suficiente e descreve como implementá-la, fornecendo exemplos.

Lendo o contexto de padrões, o usuário pode reconhecer se está numa situação relatada num certo contexto, porém nem sempre ele tem a clara noção das dificuldades a serem enfrentadas em busca da solução para o seu problema. O padrão torna explícitas as dificuldades e fornece uma solução que resolve o problema e as supera da melhor maneira.

Padrões estão classificados de acordo com a complexidade do *software* a ser desenvolvido. Padrões mais complexos fornecem um vocabulário de projeto poderoso e facilitam o desenvolvimento de *software* complexo, sendo destinados para uso por programadores experientes; são os mais facilmente encontrados na literatura. Padrões mais simples, chamados "padrões elementares" de programação, descrevem conhecimento básico em programação, tendo como propósito ajudar educadores e estudantes no aprendizado

da arte de programar. Porém, essa é uma preocupação recente dentro da área de padrões pedagógicos, sendo poucos os trabalhos existentes. A Quadro 1 mostra dois exemplos de padrões de programação.

Quadro 1 - Padrões elementares de programação para seleção (Bergin, 1999)

Nome	Contexto/Problema

se-ou-não	Alguma ação pode ou não ser apropriada, dependendo de uma condição a ser testada
	
	if (measuredHeat() > subBoilThreshold)
	{ shutDownGenerator();
	}

ação	Uma de duas ações é apropriada dependendo de uma condição. Quando a condição é verdadeira deseja-se executar uma condição, quando falsa deseja-se executar uma ação diferente
	
	if (numericGrade > 60)
	{ output ("passing");
	}
	else
	{ output("failing");
	}

O modelo do processo de construção de programas, descrito na próxima seção, baseia-se nas idéias de padrões elementares de programação e da psicologia da programação para se constituir como uma ferramenta didática especialmente dirigida a aprendizes.

O MODELO DO PROCESSO DE CONSTRUÇÃO DE PROGRAMAS

A representação mental de um programa em consideração, que o programador elabora durante o processo de entendimento ou de construção, é chamada "modelo mental do programa", e o processo global que conduz à criação deste modelo chamado "modelo cognitivo de programação" (VON MAYRHAUSER; VANS, 1994); (STOREY et al., 1999). Para maior clareza, o processo cognitivo de construção de programas será denominado aqui de "modelo do processo de construção de programas".

O modelo de processo de construção de programas para aprendizes (LEMOS, 2004) consiste em componentes e estratégias para planejamento em programação procedimental que, manipulados pelo aprendiz, visam promover a construção de conhecimento cognitivo em programação. Os elementos-chave no modelo são os planos de programação, que, quando combinados, podem representar soluções para problemas de programação, i.e, geram uma representação do programa num nível mais alto de abstração. Planos de programação são modelados com base nos padrões elementares de programação e de material didático de cursos e livros da área. A Figura 1 apresenta a estrutura genérica que define um plano de programação no modelo.

Todo plano tem um "nome" que o identifica e a descrição da "situação" em que o uso do plano é adequado. Essa informação serve como guia para o aprendiz selecionar um plano correto para resolver um problema de programação. Cada plano está associado a um "esquema", que representa um *chunk* de programação. O esquema deve ser instanciado sempre que o plano for utilizado numa solução em particular. Um conjunto de "ações primitivas" explica a lógica existente no esquema do plano; cada ação primitiva possui a linha de código que a implementa, denominada "esquema primitivo".

A enumeração, que pode ser vista na Figura 1, define a ordem das ações e dos esquemas primitivos no plano. O conjunto de esquemas primitivos e sua respectiva ordenação definem o esquema do plano.

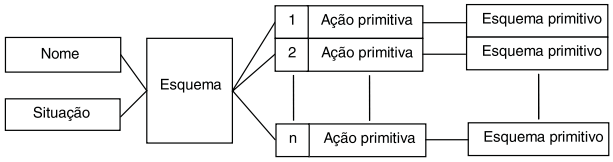

Figura 1 - Estrutura genérica de um plano de programação

O Quadro 2 apresenta um plano de programação para linguagem C. Esquemas correspondem à generalização de blocos de código específicos, permitindo, assim, seu uso na construção de diversos programas por meio da instanciação das chamadas "metavariáveis" (rótulos precedidos do caracter $).

Quadro 2 - Plano soma de seqüência automática

Situação
Você quer somar uma seqüência de números. O início e o tamanho da seqüência são conhecidos. Os números k são gerados automaticamente de forma crescente. A distância entre um número e outro da seqüência é de uma unidade. Pode ser desejável realizar algum processamento com o valor de k antes de realizar a soma.
Ações Primitivas
1: Inicializa soma com elemento neutro 2: Configura parâmetros da seqüência 3: Inicia corpo de ações a serem repetidas 4: Adiciona um no. da seqüência na soma anterior 5: Finaliza corpo de ações for
Esquemas Primitivos
1: soma = 0; 2: for(k = $inicio; k < = $tamanho; k++) 3: { 4: soma = soma + k; 5: }

Uma metavariável fornece a semântica para certo elemento a ser inserido no plano de programação. A Figura 2 apresenta as principais relações entre os componentes do modelo cognitivo. O modelo de problema, descrito por um conjunto de metas do problema, é resolvido implementando-se o modelo do programa, que, por sua vez, é composto pela hierarquia de metas do problema, planos e ações de programação. A hierarquia define o plano de programação que realiza determinada meta do problema, assim como as ações primitivas que o compõem. Cada ação primitiva é implementada por um esquema primitivo; o conjunto de esquemas primitivos ordenados forma o esquema do programa, que, quando instanciado com dados do domínio do problema, corresponde ao programa, propriamente dito. O modelo cognitivo utiliza duas bases de conhecimento: (a) a Biblioteca Cognitiva, que contém os planos de programação e descrições de problemas; (b) a Base de Conhecimento Interna, que corresponde ao conhecimento prévio do aprendiz. Dado um modelo de problema, o modelo do programa e o programa são obtidos pela manipulação dos componentes cognitivos e de implementação existentes nas bases de conhecimento.

A formalização dos componentes e estratégias do modelo pode ser vista em Lemos (2004).

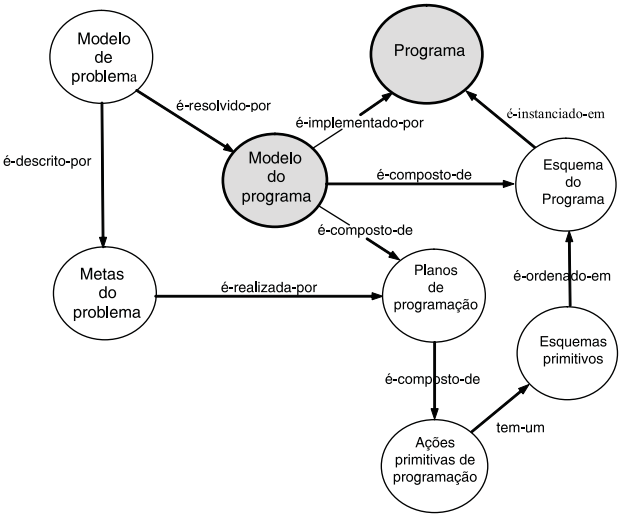

Figura 2 - Relações entre os componentes do modelo

(1) seleção de planos de programação para metas de problema, (2) ordenação de planos e tarefas primitivas e (3) mapeamento de tarefas primitivas sobre declarações em C.

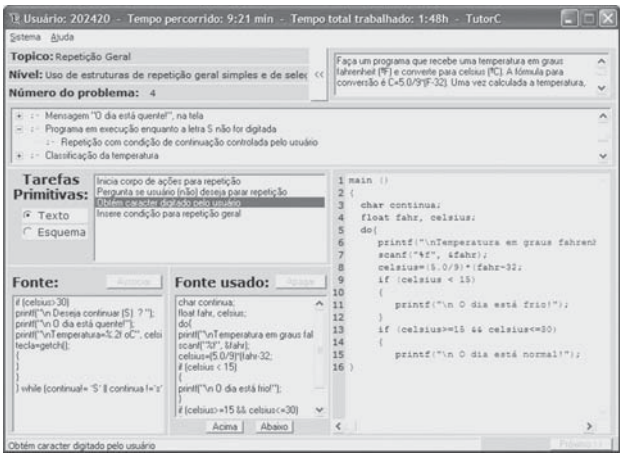

Figura 3 - Construção de programas no TutorC

A qualquer momento, tipicamente após ter seu programa finalizado e diagnosticado, o aprendiz pode disparar, a partir de TutorC, o Compilador Turbo C++ Versão 3.0 da Borland (2003). Nesse novo ambiente, o aluno pode compilar e executar seu programa, assim como fazer quaisquer alterações que vão além dos planos de programação propostos no TutorC. No TutorC, o aprendiz vai adquirindo familiarização e conhecimento sobre a sintaxe e a semântica da linguagem por associação. Para cada plano visualizado na interface, seu esquema (bloco de código generalizado) é também disponibilizado. Para cada ação primitiva selecionada pelo aprendiz na fase de ordenação, seu esquema e descrição em

texto podem ser visualizados. Finalmente, todas as declarações corretas da solução ficam disponíveis, em ordem alfabética, para que o aprendiz faça o mapeamento de cada esquema de ação primitiva com o código C correspondente. Assim, o aluno pode aprender sobre como instanciar dados do domínio do problema nas estruturas da linguagem de programação. As metavariáveis dos esquemas contribuem para esse aprendizado, uma vez que seus nomes refletem o papel que o dado do domínio desempenha na solução de problema. O aluno não precisa se preocupar com a sintaxe da linguagem, a qual é colocada na janela, automaticamente. Erros de sintaxe no início da aprendizagem são apontados como causadores de confusão em aprendizes.

METODOLOGIA DE AVALIAÇÃO

Para avaliar a aplicabilidade da abordagem cognitiva foi escolhida uma instituição de ensino superior, na qual a disciplina de Introdução à Computação é dividida anualmente em quatro módulos seqüenciais e para cada módulo os alunos são submetidos a duas avaliações presenciais: uma teórica (escrita em papel) e outra prática (no laboratório de informática). Alunos com desempenho inferior a 50% numa dada avaliação são necessariamente submetidos a uma reavaliação, sob pena de terem a nota da avaliação dividida por dois, o que compromete severamente a possibilidade de aprovação final na disciplina anual.

A Tabela 1 apresenta os dados obtidos na instituição referentes aos anos de 2002 e 2003 na disciplina de Introdução à Computação do curso de Engenharia Elétrica no período noturno.

Tabela 1 - Dados referentes ao desempenho dos alunos no módulo 1 da disciplina de Introdução à Computação submetidos à abordagem tradicional

Ano	Turma	Número de alunos	Alunos com desempenho inferior a 50% na avaliação do módulo 1	
			Número	%
2002	1	55	24	43
	2	47	19	40
2003	1	45	17	37
	2	46	22	47
Média		48	20	41

Os dados indicam que, em turmas de aproximadamente 48 alunos submetidos à abordagem tradicional, em torno de 41% apresentaram desempenho inferior a 50% na primeira avaliação teórica da disciplina, correspondente ao módulo 1.

Os 39 estudantes das turmas de 2003 que tiveram desempenho inferior a 50% na avaliação do módulo 1 foram convidados, antes de serem submetidos às reavaliações, a participar de um minicurso seguindo a abordagem cognitiva, ao invés de assistirem a aulas de reforço na abordagem tradicional, como realizado em anos anteriores. A duração do minicurso na abordagem cognitiva foi de cinco semanas, nos meses de agosto e setembro; as presenças dos alunos foram controladas, assim como o tempo de uso do sistema TutorC, o qual registra várias informações a respeito do aluno e seu percurso. Durante a reavaliação de teoria foi permitida a consulta ao conjunto de planos de programação, na forma de uma apostila distribuída aos alunos; as provas anteriores permitiam consultas apenas dos comandos da linguagem C.

Os critérios adotados para considerar um dado aluno nos testes e avaliações da abordagem cognitiva deste artigo foram:

(i) o aluno teve desempenho inferior a 50% em uma ou nas duas avaliações (teoria e laboratório) do módulo 1 da disciplina;

(ii) o aluno assistiu a um mínimo de dez horas de um total de 25 horas de aulas presenciais do minicurso;

(iii) O aluno realizou um mínimo de oito horas de trabalho individual no TutorC, em horários de sua livre escolha;

(iv) o aluno, ao realizar atividades no TutorC, fez seleção diversificada de problemas no tutor, assim como uma quantidade de, no mínimo, quatro problemas de cada tópico, totalizando no mínimo 16 problemas.

Dos 39 estudantes em reavaliação, 34 optaram por realizar o estudo cognitivo; desses, foram selecionados os 27 que respeitavam os critérios para este estudo, denominados aqui como “Grupo Selecionado”. Desse grupo, 15 realizaram reavaliação de teoria e laboratório; 11, apenas reavaliação de teoria e um realizou apenas reavaliação de laboratório. Para facilitar a apresentação dos dados referentes às reavaliações teóricas, na próxima seção, foram definidos

ainda outros dois grupos de alunos: "Grupo Participante": os 26 alunos que cumpriram o mínimo de horas e tarefas exigidas e que realizaram a reavaliação teórica, e "Grupo Não-participante", os sete alunos que não alcançaram o mínimo exigido e que realizaram a reavaliação teórica.

APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS

A Tabela 2 fornece a média e desvio-padrão das notas de avaliação e reavaliação de teoria e laboratório do módulo 1 das turmas de Introdução à Computação de dois anos consecutivos (2002 e 2003) utilizando-se a abordagem tradicional. A Tabela 2 mostra, ainda, a média e desvio-padrão das reavaliações realizadas no ano de 2003, utilizando-se a abordagem cognitiva.

Em 2002, o critério adotado para o laboratório não considerava a realização de provas de reavaliação; assim, as notas de laboratório de 2002 não foram utilizadas nessa comparação. Os anos anteriores a 2002 também não foram considerados, uma vez que a ementa e critérios de avaliação eram diferentes dos atuais. Essa diferença se deve principalmente à mudança de duração dos cursos noturnos de engenharia, os quais passaram de seis para cinco anos de duração a partir de 2002.

Tabela 2 - Média e desvio-padrão das notas do módulo 1 das turmas de 2002 e 2003

	Ano	Turma	Nº alunos	Alunos sob reavaliação			Avaliação na abordagem tradicional		Reavaliação		
				No.	%	G*	$\bar{\mu}$	σ	$\bar{\mu}$	σ	A**
Teoria Módulo 1	2002	1	55	24	43	-	1,18	1,59	2,40	2,59	A.T.
		2	47	19	40	-	1,71	1,60	2,88	2,07	A.T.
	2003	1	45	17	37	10	2,7	2,13	5,5	1,82	A.C.
		2	46	22	47	16	2,44	1,79	6,4	1,95	A.C.
Laboratório Módulo 1	2003	1	45	15	33	8	2,5	2,51	6,9	3,21	A.C.
		2	46	20	43	8	2,13	1,96	6,31	2,76	A.C.

* alunos participantes do Grupo Selecionado.
** A.T. - Abordagem Tradicional; A.C. - Abordagem Cognitiva.

Analisando os dados apresentados na Tabela 2 observa-se uma melhora significativa nas médias das reavaliações realizadas no ano de 2003, no qual se aplicou a abordagem cognitiva. Pode-se atribuir essa melhora à utilização da nova abordagem, uma vez que dados de 2002 mostram que poucos alunos se recuperam nas reavaliações. É importante lembrar que o grau de dificuldade das avaliações e reavaliações foi mantido e amostras podem ser vistas em Lemos (2004). No ano de 2002, as médias referentes à abordagem tradicional indicam que os alunos que apresentaram dificuldade no início do aprendizado (módulo 1) evoluem muito pouco quando se mantém a mesma abordagem de ensino-aprendizagem.

A Figura 4 apresenta o histograma das notas obtidas na avaliação de teoria do módulo 1 após o uso da abordagem tradicional e na reavaliação após o uso da abordagem cognitiva. Como exemplo, observa-se que no quarto intervalo de notas do histograma, na abordagem tradicional, seis alunos, e, na abordagem cognitiva, três obtiveram notas entre 3,00 e 3,99.

A Figura 5 apresenta o histograma das notas obtidas na avaliação de laboratório do módulo 1 após o uso da abordagem tradicional e na reavaliação após o uso da abordagem cognitiva. Para facilitar a comparação dos resultados obtidos com o uso das duas abordagens foram calculadas a média e o desvio-padrão das notas nas duas situações. Quantitativamente, a dispersão das notas pode ser caracterizada pelo desvio-padrão (σ) do conjunto de notas, calculados na Tabela 3.

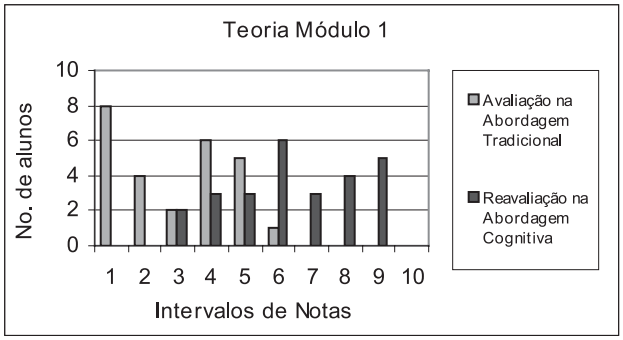

Figura 4 - Histograma das notas da prova de teoria M1 após o uso da abordagem tradicional e após o uso da abordagem cognitiva

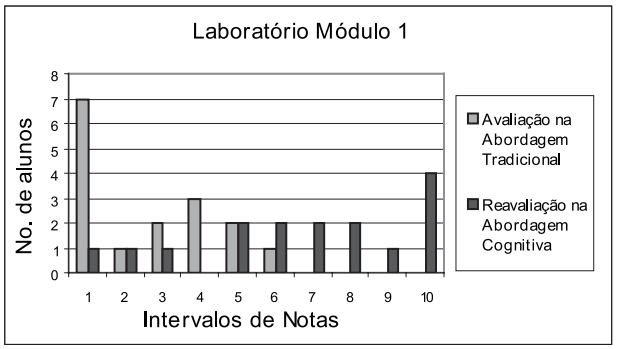

Figura 5 - Histograma das notas da prova de laboratório M1 após o uso da abordagem tradicional e após o uso da abordagem cognitiva

Tabela 3 - Média e desvio-padrão das notas de provas do grupo de alunos

	Teoria Módulo 1		Laboratório Módulo 1	
	$\bar{\mu}$	□	$\bar{\mu}$	□
Avaliação na abordagem tradicional	2,54	1,94	2,31	2,18
reavaliação na abordagem cognitiva	6,10	1,89	6,63	2,91

Analisando os dados, observa-se que houve significativo progresso no desempenho dos alunos no laboratório, porém a dispersão em relação à média é maior do que a apresentada nas provas de teoria em sala de aula. Pode-se atribuir essa diferença a alguns fatores. No laboratório, o aluno necessita de habilidades adicionais, tais como familiarização com o compilador e prática em tratar erros nas várias fases do desenvolvimento do programa. Além disso, na prova de laboratório, o aluno não pôde consultar os planos de programação, o que exigiu dele uma carga cognitiva maior. A dispersão mostra que esses fatores adicionais são diferenciados para cada aluno.

Outro levantamento de dados realizado diz respeito à porcentagem de acertos na prova de reavaliação de teoria pelo grupo de alunos que cumpriu o mínimo de horas exigido (26 alunos), denominado Grupo Participante, e os que não cumpriram (7 alunos), denominado Grupo Não-Participante.

A prova teve como objetivo verificar três tópicos de conhecimento no aluno: (1) compreensão de programas; (2) construção de programas; (3) identificação de metas e planos de programação. A Tabela 4 apresenta, em porcentagem, a média de acertos para cada um dos tópicos.

Tabela 4 - Porcentagem de acertos na reavaliação de teoria módulo 1 na abordagem cognitiva

	Grupo Participante	Grupo Não-Participante
Compreensão de programas	58,0%	13,5%
Construção de programas	63,2%	27,4%
Identificação de metas e planos	61,0%	27,0%

Os dados indicam uma tendência de que o progresso promovido envolve não somente construção de programas como também a compreensão de programas. Observando os dados da Tabela 3, o grupo participante obteve 58% de acertos em compreensão de programas contra 13,5% do grupo não participante. Em construção de programas, o grupo participante obteve 63,2% de acertos contra 27,4% do Grupo Não-Participante. Pode-se justificar esse progresso pelo seguinte fato: na abordagem tradicional enfatiza-se, tipicamente, a prática de simulações (inspeção de valores de variáveis) para compreensão do comportamento do programa (ROCHA, 1995), ao passo que a abordagem cognitiva enfatiza o conhecimento sobre ações primitivas de programação e seu mapeamento com o código.

Outra observação possível, com base nos dados da Tabela 4, diz respeito à identificação de metas do problema e sua associação com planos de programação. Os dados indicam que tal conhecimento está diretamente relacionado com a capacidade de construção de programas. Os dois grupos mostram essa relação. O Grupo Participante obteve 61% de acertos ao identificar metas e seus planos e 63,2% em construção de programas. Por sua vez, o Grupo Não-Participante obteve 27% de acertos ao identificar metas e seus planos e 27,4% em construção de programas. Esse é um resultado importante, que corrobora resultados de estudos empíricos da psicologia

da programação (BROOKS, 1983; JOHNSON, 1990), conforme já mencionado.

Obviamente, outras aplicações devem ser realizadas para refinar as conclusões sobre a eficácia dessa nova abordagem, uma vez que alguns pontos são questionáveis. Por exemplo, a Tabela 1 indica que as notas pela avaliação tradicional já eram inferiores nas turmas de 2002 em relação a 2003, o que mostra uma tendência de haver também um pior desempenho na reavaliação. Por outro lado, a diferença de desempenhos na reavaliação entre as duas abordagens parece ser grande para justificá-la como tendência: a diferença entre a nota de avaliação e a de reavaliação foi de 1,19, em média, na abordagem tradicional, ao passo que na abordagem cognitiva a diferença foi de 3,38. Uma justificativa para a diferença de desempenho entre as abordagens poderia vir do fato de os alunos que passaram pela reavaliação na abordagem cognitiva terem sido controlados na realização de tarefas para resgate da aprendizagem, o que não aconteceu no caso anterior. De qualquer forma, na abordagem tradicional sempre houve aulas de reforço com tarefas práticas, porém sem controle de presença sobre os estudantes. As consultas durante as provas sempre ocorreram, porém na abordagem tradicional é feita aos comandos da linguagem, ao passo que na abordagem cognitiva a consulta é sobre planos de programação.

CONCLUSÕES

Os resultados obtidos com a avaliação preliminar apresentada neste trabalho fornecem indicativos de que a adoção do modelo proposto em cursos de introdução à programação pode facilitar o aprendizado dos alunos, que, tipicamente, apresentam dificuldades nessa área quando expostos somente à abordagem tradicional de ensino de programação. O processo de construção de programas embutido no modelo avaliado não supõe que este seja uma representação fiel do comportamento de programadores experientes, mas, sim, um meio para promover o desenvolvimento cognitivo de aprendizes em programação. O programa que o aprendiz define nesta abordagem pode ser visto como uma descrição do seu processo de pensamento, ou seja, existem uma proposta de solução do problema no nível mental (o modelo abstrato do programa) e uma descrição da solução no nível da implementação (o programa). A proposta do sistema TutorC é permitir um mapeamento entre essas duas representações na forma de um caminho que deve ser percorrido explicitamente pelo estudante.

A abordagem fornece ao aprendiz uma representação explícita dos componentes cognitivos (por meio de uma apostila de planos de programação ou armazenados no sistema TutorC), os quais se deseja que o estudante aprenda a usar e modificar. No Tutor C o aprendiz é livre para realizar estratégias cognitivas de programação, tais como seleção de planos de programação, ordenação de planos e ações de programação, assim como a instanciação desses componentes. Por outro lado, TutorC controla o processo de construção do programa fornecendo metas do problema, planos prontos e outras facilidades, assim como realiza diagnósticos impedindo o aprendiz de prosseguir o processo com erro. Esse equilíbrio entre controle e liberdade é desejável quando se aplica o construtivismo no ensino de ciência da computação. O construtivismo defende a idéia da construção do conhecimento por meio do "errar" para "aprender", da liberdade em oposição ao controle.

No entanto, a experiência tem sugerido que o efetivo uso do paradigma do construtivismo no ensino de ciência da computação torna-se viável somente após o aluno ter conseguido construir uma estrutura de conhecimento básica de uma máquina computacional e alguma percepção da importância dos níveis de abstração (BEN-ARI, 2001). Na abordagem cognitiva, o aprendiz tem o apoio de componentes e estratégias em diferentes níveis de abstração, e é sobre essa base de conhecimento explícita que o aprendiz constrói seu novo conhecimento sobre programação. Assim, planos de programação são como andaimes numa construção, que, aos poucos, podem ser retirados para que o aprendiz continue sua evolução de forma autônoma.

REFERÊNCIAS

BEN-ARI, M. Constructivism in computer science education. *Journal of Computers in Mathematics & Science Teaching*. Association for Computing Machinery, Inc. 20(1), p. 45-73, 2001. Disponível em: <http://www. weizmann.ac.il/G-CS/BENARI>. Acesso em: jan. 2004.

BONAR, J.; SOLOWAY, E. Uncovering principles of novice programming. *ACM*, p. 10-13, 1983.

BROOKS, R. Towards a theory of the comprehension of computer programs. *International Journal of Man-Machine Studies*, v. 18, p. 543-554, 1983.

HILLSIDE.NET. Hillside Group Homepage. 1993-2005. Contém informações sobre padrões e linguagens de padrões para aplicação em desenvolvimento de software. Organiza conferências, workshops e publicações para documentar práticas de sucesso em *software*. Disponível em: <http://www.hillside.net>. Acesso em: jun. 2003.

JOHNSON, W. L. Understanding and Debugging Novice Programs. *Artificial Intelligence*, v. 42, p. 51-97, 1990.

LEMOS, M. A. *Uma abordagem baseada em padrões elementares para aprendizado de programação*. 2004. Tese (Doutorado em Engenharia Elétrica) - Universidade de São Paulo, São Paulo.

LEMOS, M. A.; BARROS, L. N.; LOPES, R. D. Modeling plans and goals of Programming in an intelligent tutoring system. In: WORKSHOP ON KNOWLEDGE REPRESENTATION AND AUTOMATED REASONING FOR E-LEARNING SYSTEMS (KRR-5) held on 18th International Joint Conference on Artificial Intelligence (IJCAI), 8, 2003a. Acapulco, México.

LEMOS, M. A.; BARROS, L. N.; LOPES, R. D. Uma biblioteca cognitiva para o aprendizado de programação. In: WORKSHOP DE EDUCAÇÃO EM COMPUTAÇÃO (WEI), XI. Simpósio Brasileiro de Computação (SBC), Campinas, 02-08 agosto, 2003b.

OLIVEIRA, R. T. D. Construção de um Modelo de domínio num tutor inteligente para Ensino da LinguagemC. In: SIMPÓSIO INTERNACIONAL DE INICIAÇÃO CIENTÍFICA DA USP, 10. São Carlos, 2002.

PANE, J. F.; MYERS, B. A. Usability issues in the design of novice programming systems. Pittsburgh: Human-Computer Interaction Institute, Carnegie Mellon University, 1996. (*Technical Report* Carnegie Mellon University, CMU-HCII-96-101). Disponível em: <http://www-2.cs.cmu.edu/~pane/publications.html>.

ROCHA, H. V. Representações computacionais auxiliares ao entendimento de conceitos de programação. In: *Computadores e conhecimento:* repensando a educação. Gráfica Cultural Unicamp. Separata 16, 1995. Disponível em: <http://www.nied.unicamp.br/publicacoes/>. Acesso em: jan. 2004.

ROLINO, A. Explicando erros para o estudante no TutorC. In: SIMPÓSIO INTERNACIONAL DE INICIAÇÃO CIENTÍFICA DA USP, 11. São Carlos, 2003.

SHUTE, V. J.; PSOTKA, J. Intelligent tutoring systems: past, present and future. *Handbook of Research on Educational Communications and Technology*, Scholastic Publications, D. Jonassen (Ed.), 1994.

SOLOWAY, E.; EHRLICH, K. Empirical Studies of Programming Knowledge. *IEEE transactions on software engineering*, IEEE Computer Society, v. SE-10, n. 5, p. 595-609, Sep. 1984.

UNI. Iowa, *University of Northern Iowa*, 1995-2005. Contém a homepage de padrões elementares, integra a comunidade de pesquisadores, trabalhos realizados e em andamento na área. Disponível em: <http://www.cs.uni.edu/~wallingf/patterns/elementary/>. Acesso em: jun. 2003.

WALLINGFORD, E. Elementary patterns and their role in instruction. In: OOPSLA'98 EDUCATORS SYMPOSIUM NOTES, 1998. Disponível em: <http://www.cs.uni.edu/~wallingf/ patterns/elementary/chiliplop98/summary.html>. Acesso em: 20 jul. 2003.

DADOS DOS AUTORES

Marilza Antunes de Lemos

É Doutora em Engenharia Elétrica pela Universidade São Paulo (2004), professora Assistente do departamento de Engenharia de Controle e Automação da Universidade Estadual Paulista Júlio de Mesquita Filho, Campus de Sorocaba e pesquisadora no Laboratório de Automação Industrial do campus. Orientou projetos educacionais envolvendo robótica, *fuzzy* e visão computacional. Outras áreas de atuação são sistemas microprocessados, programação de sistemas discretos de manufatura, sistemas nebulosos e sistemas tutores para aprendizado de programação.

Roseli de Deus Lopes

É Doutora em Engenharia Elétrica pela Universidade de São Paulo (1998), professora Assistente do Departamento de Engenharia de Sistemas Eletrônicos da USP. Especialista em Meios Eletrônicos Interativos; pesquisadora do Laboratório de Sistemas Integráveis (LSI) da EPUSP, onde coordena o Núcleo de Aprendizagem/ Trabalho/Entretenimento e o Núcleo de Tecnologias Livres, sendo responsável, atualmente, pelo projeto "Um Laptop por criança" no Brasil. Coordena, ainda, a Feira Brasileira de Ciências e Engenharia (Febrace) desde 2002 e é a atual diretora da Estação Ciência da USP.