COBENGE 2005
XXXIII - Congresso Brasileiro de Ensino de Engenharia
"Promovendo e valorizando a engenharia em um cenário de constantes mudanças"
12 a 15 de setembro - Campina Grande Pb
Promoção/Organização: ABENGE/UFCG-UFPE

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

Marilza Antunes de Lemos – marilza@sorocaba.unesp.br
Universidade Estadual Paulista, Departamento de Engenharia de Controle e Automação
Av. Três de Março, 511 – Alto da Boa Vista
18087-180 – Sorocaba – SP – Brasil

Roseli de Deus Lopes – roseli.lopes@poli.usp.br
Universidade de São Paulo, Departamento de Engenharia de Sistemas Eletrônicos - EPUSP
Avenida Prof. Luciano Gualberto, travessa 3, nº 158
05508-900 – São Paulo – SP – Brasil

Leliane Nunes de Barros – leliane@ime.usp.br
Universidade de São Paulo, Departamento de Ciência da Computação - IME
Rua do Matão, 1010 - Cidade Universitária
05508-090 – São Paulo – SP – Brasil

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 promissoras possibilidades de aplicação desta abordagem cognitiva.

Palavras-chaves: Sistema tutor, Padrões de programação

1. 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 sub-área 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 nos anos 80 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, os quais condensam ações primitivas. 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. GELLENBECK, COOK e WIEDENBECK apud PANE e 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 definida na literatura como aquela que auxilia pessoas a executarem atividades mentais que não podem ser observadas diretamente ou que podem ser pouco observáveis, tais como ajudar a pensar, conhecer ou aprender.

Baseado nesse modelo cognitivo foi implementado o sistema TutorC (LEMOS et al., 2003a) para aprendizado de programação em linguagem C. O sistema foi 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.

2. ABORDAGEM TRADICIONAL X ABORDAGEM COGNITIVA

Tradicionalmente, em sala de aula, o professor enquanto explica a construção ou compreensão de um programa identifica objetivos a serem alcançados, extraídos a partir do enunciado, relembra conceitos da linguagem de programação e mostra o mapeamento entre objetivos e estruturas da linguagem. Fornece ainda o significado das linhas de código utilizadas no programa, explica instanciações de dados do problema no código, entre outras atividades. Porém, todo esse conhecimento é transmitido ao aluno de maneira informal, ou seja, não é documentado. Uma vez que, nessa situação, a carga cognitiva exigida do estudante é muito grande, este acaba por não assimilar todo o conhecimento, com sucesso.

A abordagem cognitiva avaliada neste trabalho formaliza alguns tipos de conhecimento manipulados na abordagem tradicional: decomposição de problemas em metas a serem realizadas, 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 as abordagens reside no aspecto da formalização e documentação dos tipos de conhecimento necessários à atividade de programação. Neste artigo será feita referência às duas abordagens com os termos: abordagem tradicional e abordagem cognitiva.

2.1 Fundamentos da Abordagem Cognitiva

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, combinados, podem representar soluções para problemas de programação. Planos de programação são modelados a partir de padrões elementares de programação da área de Padrões Pedagógicos (UNI, 2003) 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** explicam a lógica existente no esquema do plano. Cada ação primitiva possui a linha de código que a implementa, denominada **esquema primitivo**.

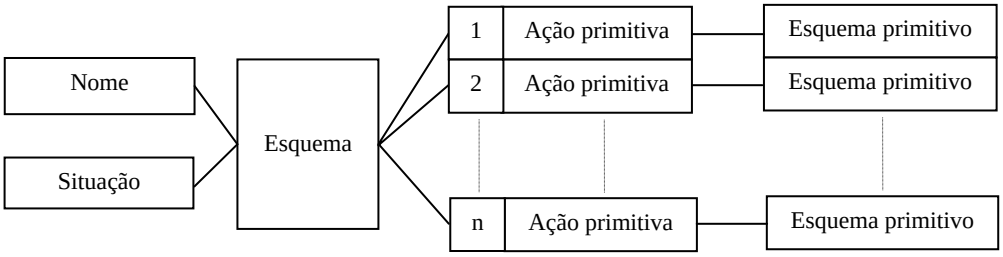

Figura 1 – Estrutura genérica de um plano de programação

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. Esquemas correspondem à generalização de blocos de código específicos permitindo, assim, seu uso na construção de diversos programas por meio da instanciação das chamadas meta-variáveis (rótulos precedidos do caracter $). Uma meta-variável fornece a semântica para um certo elemento a ser inserido no plano de programação. A Tabela 1 apresenta um plano de programação simples para linguagem C.

Tabela 1 – Plano de Programação Entrada-por-Teclado

Situação	**Ações Primitivas**	**Esquemas Primitivos**
Você está numa situação onde o usuário precisa interagir com o programa, informando dados. Esses dados podem ser numéricos ou caracteres.	1: Emite mensagem para o usuário	printf ($formato-de-saída);
	2: Obtém dados do usuário	scanf ($formato-de-entrada, $endereço);

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**. Este, 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 compõe. Cada ação primitiva é implementada por um **esquema primitivo**. O conjunto de esquemas primitivos ordenados forma o **esquema do programa**. Este, 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, a qual contém os planos de programação e descrições de problemas e (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). Baseando-se no modelo cognitivo, resumidamente descrito neste artigo, foi implementado o sistema TutorC, um ambiente para aprendizagem de programação em linguagem C (OLIVEIRA, 2002; ROLINO, 2003). A Figura 3 apresenta um exemplo de janela de TutorC.

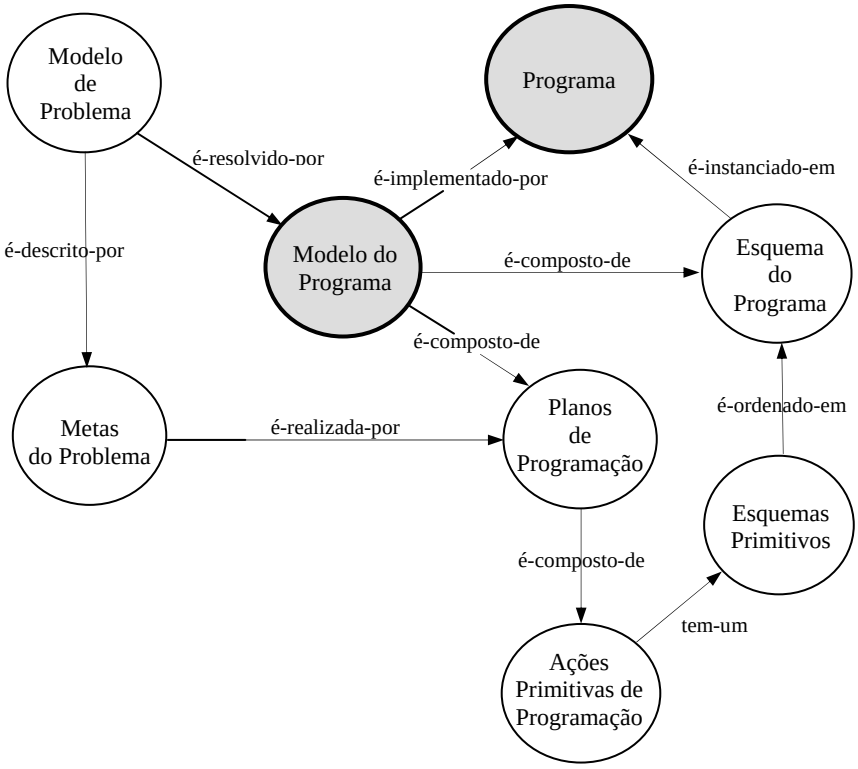

Figura 2 – Relações entre os componentes do modelo

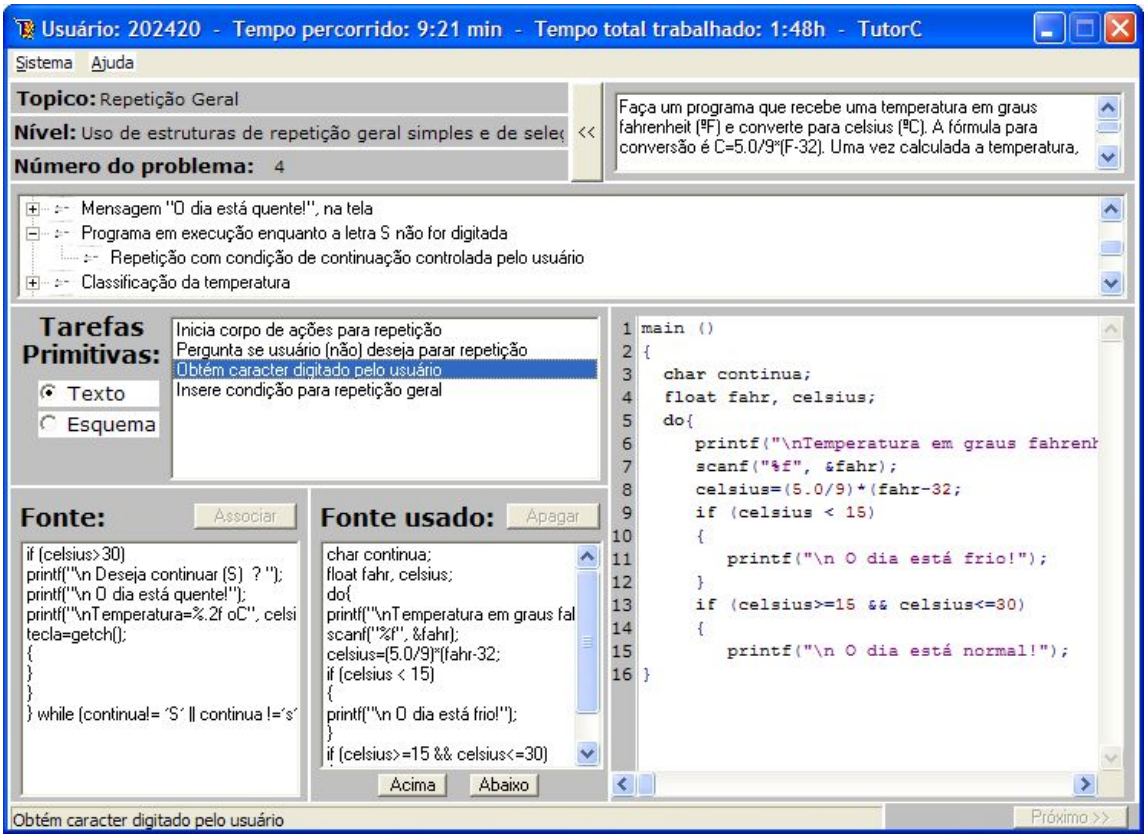

Figura 3 – Construção de Programas no TutorC

O conhecimento especialista de TutorC é modelado e armazenado por meio da BibPC (Biblioteca para Programação Cognitiva) (LEMOS et al., 2003b), uma ferramenta de autoria capaz de capturar conhecimento de planejamento em programação C, para uso e expansão por

educadores. Assim, TutorC é capaz de disponibilizar ao aprendiz, componentes de planejamento e de implementação de programas obtidos da BibPC. Com o TutorC, o aprendiz pode desenvolver programas em C estimulado pela presença de componentes como: planos e ações de programação, metas de problemas e declarações em linguagem C.

A janela da Figura 3 é apresentada ao aprendiz após este ter selecionado um conjunto de planos de programação válidos para a construção de uma solução correta para o problema sob estudo. Nela, o aprendiz pode visualizar a decomposição desses planos: suas ações e esquemas primitivos. Uma das tarefas nessa janela consiste em ordenar, de forma lógica, tais componentes. Além disso, TutorC oferece ao aprendiz os esquemas primitivos instanciados, ou seja, as linhas de código desordenadas, as quais devem ser mapeadas às ações primitivas para finalmente compor o programa.

3. METODOLOGIA DA APLICAÇÃO

Para avaliar a aplicabilidade da abordagem cognitiva foi escolhida a instituição de ensino superior particular, a Faculdade de Engenharia de Sorocaba, em que a primeira autora do artigo era a professora responsável pela disciplina de Introdução à Computação para o Curso de Engenharia Elétrica. Nesta instituição, a disciplina de Introdução à Computação é anual, dividida em quatro módulos sequenciais e para cada módulo os alunos são submetidos a duas avaliações presenciais: uma teórica (escrita) e outra de laboratório (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 e comprometendo severamente a possibilidade de aprovação final na disciplina anual.

Dados obtidos na instituição referentes aos anos de 2002 e 2003 da disciplina de Introdução à Computação do curso de Engenharia Elétrica no período noturno são apresentados na Tabela 2. Tais dados indicam que em turmas de aproximadamente 48 alunos submetidos à abordagem tradicional, em torno de 41% destes apresentaram desempenho inferior a 50% na primeira avaliação teórica da disciplina, correspondente ao módulo 1.

Tabela 2 – 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 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 participarem de um mini-curso seguindo a abordagem cognitiva ao invés de aulas de reforço na abordagem tradicional, como realizado em anos anteriores. A duração do mini-curso 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 apostila, a qual foi distribuída aos alunos.

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 mini-curso;
(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 dezesseis 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 realizaram apenas reavaliação de teoria e 1 realizou apenas reavaliação de laboratório.

Para facilitar a apresentação dos dados referente 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 7 alunos que não alcançaram o mínimo exigido e que realizaram a reavaliação teórica.

4. APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS

A Tabela 3 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 3 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 realização de provas de reavaliação. Assim, as notas de laboratório de 2002 não foram utilizadas nesta 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 pela mudança de duração dos cursos noturnos de engenharia, os quais passaram de seis para cinco anos de duração a partir de 2002.

Tabela 3 – 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*	$\overline{\mu}$	σ	$\overline{\mu}$	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 3 observa-se uma melhora significativa nas médias das reavaliações realizadas no ano de 2003, no qual aplicou-se a abordagem cognitiva. Pode-se atribuir essa melhora à utilização da nova abordagem uma vez que dados do ano 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 alunos, tiraram notas entre 3,00 e 3,99.

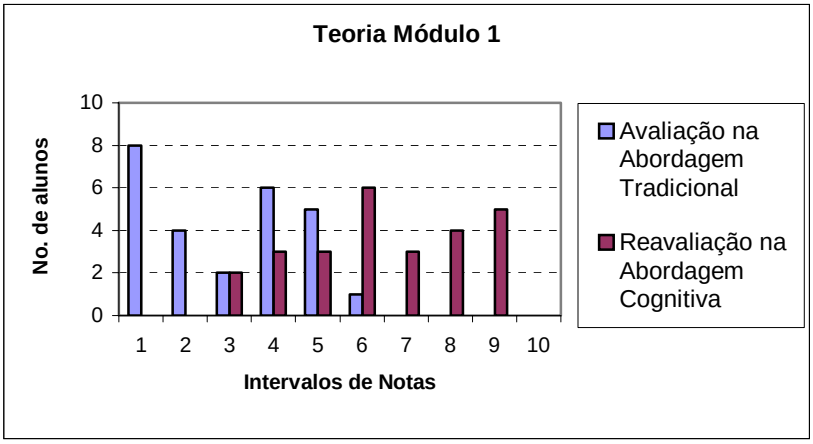

Figura 4 – Histograma das notas da prova de teoria M1 após o uso da abordagem tradicional e após o uso da abordagem cognitiva

A Figura 5 apresenta o histograma das notas obtidas na Avaliação de Laboratório do Módulo1 após o uso da Abordagem Tradicional e na Reavaliação após o uso da Abordagem Cognitiva.

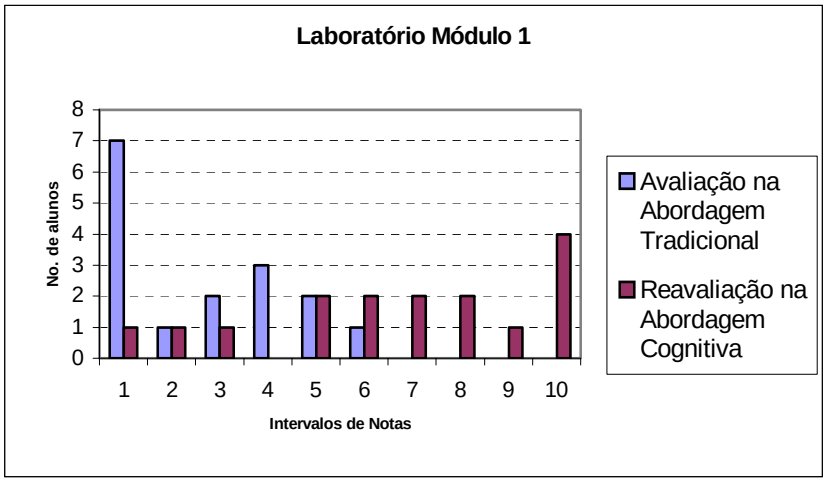

Figura 5 – Histograma das notas da prova de laboratório M1 após o uso da abordagem tradicional e após o uso da abordagem cognitiva

Para facilitar a comparação dos resultados obtidos com o uso das duas abordagens foram calculados a média, o desvio-padrão e a distribuição normal das notas nas duas situações. Quantitativamente, a dispersão das notas pode ser caracterizada pelo desvio-padrão (σ) do conjunto de notas definido na equação (1), como:

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{\mu})^2} \qquad (1)$$

onde x_i é o resultado da *i*-ésima nota, *N* é o número total de notas e $\bar{\mu}$ é a média aritmética das notas dada pela equação (2), como:

$$\bar{\mu} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad (2)$$

Assim, o desvio-padrão fornece uma medida do grau de dispersão das notas em relação à média. A Tabela 4 apresenta a média e o desvio-padrão das notas do grupo selecionado.

Tabela 4 – Média e desvio-padrão das notas de provas do grupo de alunos

	Teoria Módulo 1		Laboratório Módulo 1	
	$\bar{\mu}$	σ	$\bar{\mu}$	σ
Avaliação na Abordagem Tradicional	2,54	1,94	2,31	2,18
Reavaliação na Abordagem Cognitiva	6,10	1,89	6,63	2,91

A partir da média e desvio-padrão obtidos em cada uma das abordagens foram gerados os gráficos da função de distribuição normal f(x), das notas (x), dada pela equação (3):

$$f(x) = \frac{1}{\sqrt{2\pi}\,\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \qquad (3)$$

A Figura 6 mostra os gráficos das funções de distribuição normal das notas de provas de teoria realizadas com as duas abordagens.

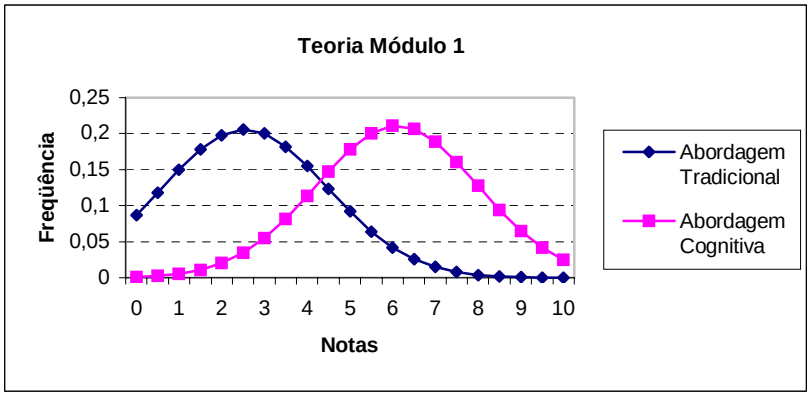

Figura 6 – Distribuição normal das notas de teoria com o uso da abordagem tradicional e com o uso da abordagem cognitiva

A Figura 7 mostra os gráficos das funções de distribuição normal das notas de provas de laboratório realizadas com as duas abordagens.

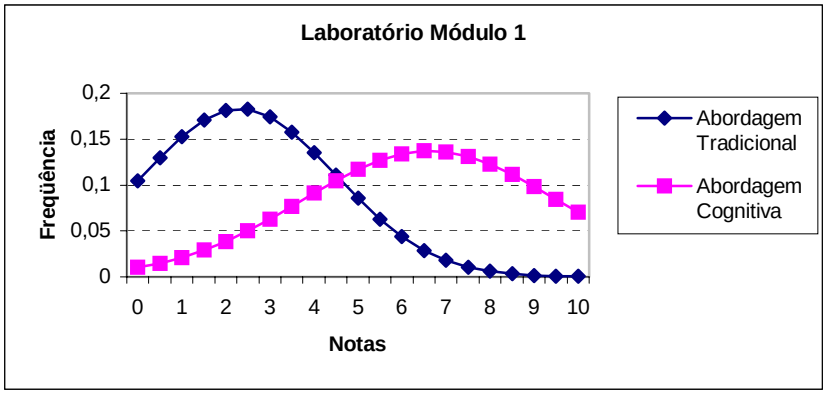

Figura 7 – Distribuição normal das notas de laboratório com o uso da abordagem tradicional e com o uso da abordagem cognitiva

Analisando o gráfico de distribuição normal da Figura 7 observa-se que houve progresso no desempenho dos alunos no laboratório, porém a dispersão em relação à média é maior do que a apresentada na Figura 6 que corresponde às provas de teoria, em sala de aula. Alguns fatores explicam essa diferença. 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, exigindo 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 e (3) identificação de metas e planos de programação. A Tabela 5 apresenta, em porcentagem, a média de acertos para cada um dos tópicos.

Tabela 5 – 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 compreensão de programas. Observando os dados da Tabela 5, 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. Na abordagem tradicional enfatiza-se, tipicamente, a prática de simulações (inspeção de variáveis) para compreensão de programas (ROCHA, 1995), enquanto que a abordagem cognitiva enfatiza o conhecimento sobre ações primitivas de programação e seu mapeamento com o código.

Outra observação possível, a partir dos dados da tabela 5, 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. 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 com resultados de estudos empíricos da Psicologia da Programação (BROOKS, 1983; JOHNSON, 1990), conforme mencionado na seção 1.

5. CONCLUSÕES

Os resultados obtidos com a avaliação preliminar realizada fornecem indicativos de que a adoção do modelo proposto em conjunto com a abordagem tradicional em cursos de introdução à programação pode facilitar o aprendizado dos alunos que, tipicamente, apresentam dificuldades nessa área.

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 nessa abordagem pode ser visto como uma descrição do seu processo de pensamento. Ou seja, existe uma proposta de solução do problema no nível mental (o modelo mental do programa) e uma descrição da solução no nível da implementação (o programa). O mapeamento entre as duas representações é um caminho 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 no sistema TutorC), os quais deseja-se que o estudante aprenda a criar. Por outro lado, 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. Esse equilíbrio entre controle e liberdade é desejável quando se aplica o construtivismo em Ciência da Computação.

A experiência tem sugerido que o efetivo uso do paradigma do construtivismo no ensino de Ciência da Computação tem se viabilizado 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.

6. REFERÊNCIAS BIBLIOGRÁFICAS

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://stwww.weizmann.ac.il/g-cs/benari/articles/ cons.pdf>. 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: XI WORKSHOP DE EDUCAÇÃO EM COMPUTAÇÃO (WEI). 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 Linguagem C. 10º SIMPÓSIO INTERNACIONAL DE INICIAÇÃO CIENTÍFICA DA USP, 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. Capítulo do livro **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. 11º SIMPÓSIO INTERNACIONAL DE INICIAÇÃO CIENTÍFICA DA USP, 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, September, 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 julho 2003.

EVALUATION OF THE PROGRAMMING LEARNING-TEACHING USING AN APPROACH BASED ON PROGRAMMING ELEMENTARY PATTERNS

Abstract: *The difficulties faced by students to learn programming languages have been observed and described by several researchers from different areas. This article is based on a model of Program Construction Process for novices developed in order to enable the construction of systems that can minimize some of the recognized problems faced by students in programming learning. Following this model, we developed a tutor system for programming learning using C language. This tutor system was experimentally used with a group of Electrical Engineering students. This article presents and evaluates the results of the use of this cognitive approach, which shows clear positive possibilities.*

Key-words*:* Tutoring system, Programming patterns