Ambiente Interativo e Adaptável para ensino de Programação

Homero L. Píccolo[1], Vinícius de F. Sena[1], Kamila B. Nogueira[1], Marcus O. da Silva[1], Yuri A. N. Maia[1]

[1]Departamento de Ciência da Computação – Universidade de Brasília
Caixa Postal 4466 – 70910-970 - Brasília - DF - Brasil

homero@cic.unb.br, vinicius_sena@ig.com.br,
kamila.b.nogueira@gmail.com, marcus_oliveira15@hotmail.com,
maiayuri@hotmail.com

Abstract. *This work describes an educational environment, interactive and adaptable, named TutorICC. It was developed for teaching of Pascal programming at the Universidade de Brasília. TutorICC is suitable for b-learning , the students having not to attend to classes, although their presence is required for exams. TutorICC can adapt the hardness level and sequence of the lessons based on the progress of each student. There are great interactiveness between TutorICC and the student, who is invited to build his own programs at each tutorial step. These programs are immediately checked by TutorICC, that, based on students' accuracy, guides them to the best path to achieve the required knowledge.*

Resumo. *Este artigo descreve um ambiente interativo e adaptável, denominado TutorICC, desenvolvido e utilizado para o ensino de programação em Pascal na Universidade de Brasília. O TutorICC é utilizado na forma de ensino semi-presencial, em que os alunos não têm aulas, mas fazem as provas no laboratório da Universidade. O conteúdo está dividido em níveis de dificuldade, permitindo ao aluno estabelecer o caminho mais conveniente para ele, dentro do conteúdo da disciplina. Há uma grande interatividade com o aluno, que é convidado a construir seus próprios programas a cada passo do tutorial. Esses programas são corrigidos imediatamente pelo TutorICC, que recomenda ao aluno o melhor caminho a seguir dentro do conteúdo da disciplina.*

1. Introdução

Este artigo descreve um ambiente para o ensino de programação que está sendo desenvolvido e testado no Departamento de Ciência da Computação da Universidade de Brasília (UnB), na disciplina Introdução à Ciência da Computação (ICC), denominado TutorICC. Esta disciplina consiste basicamente no ensino da linguagem Pascal e está sendo ministrada em 12 turmas, para 360 alunos, em sistema semi-presencial. O aluno deve comparecer apenas para fazer as provas. Todo o seu estudo, treinamento e avaliação são feitos utilizando o TutorICC.

As provas consistem em programas que o aluno deve desenvolver presencialmente no laboratório da Universidade. Esses programas são corrigidos por um programa Corretor de provas, que faz parte do TutorICC.

O conteúdo da disciplina pode ser percorrido pelo aluno de forma variada. Ao longo do aprendizado ele vai sendo testado pelo TutorICC. A cada passo é apresentado ao aluno um problema, para que construa um programa. Esse programa é corrigido *on line* pelo TutorICC, que recomenda qual o próximo passo a ser dado. Desta forma o ritmo de aprendizagem se adapta ao perfil do aluno, podendo ser mais rápido ou mais lento, dependendo de sua maior ou menor facilidade em aprender programação.

Este artigo está organizado em sete seções: as seção 2, 3, e 4 apresentam conceitos básicos e trabalhos efetuados no contexto do ensino de computação; as seções 5 e 6 apresentam o sistema TutorICC: na seção 5 é apresentada a dinâmica do processo de ensino/aprendizagem pelo sistema e na seção 6 descrevem-se alguns de seus recursos mais importantes. Na seção 7 fazem-se alguns comentários finais sobre o sistema e suas possibilidades futuras.

2. Ensino-aprendizagem de fundamentos de programação

A disciplina Introdução à Ciência da Computação (ICC) é de serviço, sendo oferecida pelo Departamento de Ciência da Computação para alunos de outros cursos. É comum que os professores não se sintam encorajados a lecionar disciplinas de serviço, por vários motivos, entre eles, o fato de serem desvinculadas de suas linhas de pesquisa, além de que os alunos são de outros departamentos. A disciplina quase sempre foi ministrada por professores substitutos, que em geral têm pouca experiência didática e entre os quais a rotatividade é alta.

Segundo Pereira Júnior e Rapkiewicz (2004), há uma dificuldade por parte dos alunos em assimilar os conceitos envolvidos em programação. Os estudantes de disciplinas introdutórias na área da Computação muitas vezes têm dificuldade de empregar o raciocínio lógico envolvido, o que gera um ambiente desmotivante, causando uma grande evasão e também grande reprovação na disciplina. Há muitas dificuldades envolvidas na tarefa de desenvolver as habilidades relacionadas ao aprendizado de uma nova linguagem de programação e de como resolver problemas a serem resolvidos por um computador (Barros *et al,* 2004).

Os professores que ministram as disciplinas relacionadas aos fundamentos da programação também encontram problemas, como a heterogeneidade dos alunos, que provêm dos mais variados cursos, grande quantidade de alunos por professor, etc. (Castro *et al*, 2004).

Diante desse quadro é premente a necessidade de pesquisar estratégias e ferramentas de ensino de disciplinas iniciais de programação. Desenvolver formas de ensino apoiadas por ambientes colaborativos e inteligentes que cuidem de auxiliar no entendimento, execução e visualização de resultados mostra-se uma boa proposta de ensino-aprendizagem (Pereira Júnior e Rapkiewicz, 2004).

3. Sistemas Tutores Inteligentes

Os Sistemas Tutores Inteligentes, ou STI, podem ser definidos como sistemas de ensino-aprendizagem baseados em computador que procuram adaptar-se às necessidades do

aluno (Self, 1999). Esses programas de computador utilizam técnicas de Inteligência Artificial (IA) para representar o conhecimento e com isso possibilitar uma interação com o aluno (Sleeman, 1982).

Entre outros autores, Self (1999) afirma que nos STI há três elementos básicos: o modelo do domínio, o modelo do estudante e o modelo do tutor. O modelo do aluno representa o estudante em seu conhecimento, comportamento e cognição. O modelo do domínio procura modelar o conhecimento detido por especialistas de um determinado assunto e o modelo do tutor possui estratégias de ensino que procuram aproximar o conhecimento do aluno ao conhecimento do modelo de domínio.

Os STI se constituíram como uma evolução dos antigos sistemas de instrução auxiliada por computador que utilizavam técnicas de IA em seus projetos (ICAI – *intelligent computer assisted instruction*). Segundo Vicari e Giraffa (2003) a diferença entre os ICAI e os STI é que estes últimos tomam decisões pedagógicas de acordo com um modelo cognitivo do aluno e sua interação com um programa. Como conseqüência, a estrutura de um STI seria dividida em módulos e a seqüência de apresentação do conteúdo dar-se-ia de acordo com a interação com o aluno.

4. Trabalhos Anteriores

Na literatura encontram-se diversas tentativas de criar ferramentas ou desenvolver estratégias de ensino que permitam minimizar a dificuldade na aprendizagem dos conceitos fundamentais de computação. Já em Anderson e Reiser (1985) desenvolveu-se o LISPTutor, o primeiro STI criado para auxiliar no ensino de programação. Este tutor objetivou prover um ambiente amigável para a resolução de um problema e um editor inteligente. Quando o aluno cometia um erro de programação ou de planejamento, ou ao pedir ajuda, o programa fornecia informações que guiavam o aluno pelo caminho da solução.

No Simpósio Brasileiro de Informática na Educação (SBIE) e no Workshop sobre Educação em Computação (WEI), entre 1999 e 2003, por exemplo, houve pelo menos um artigo por ano sobre ensino de programação em um dos eventos (Pereira Júnior e Rapkiewic, 2004). As ferramentas apresentadas em eventos como o SBIE e o WEI têm apresentado, entre outras, as seguintes funcionalidades: exposição do conteúdo de programação; simulação de execução de programas ou algoritmos; e mecanismos automáticos de correção e contribuição de soluções computacionais para problemas. A viabilização de ambiente colaborativo de correção de código (Tobar, 2001) e simulação de diálogos de alunos (Santos Júnior e colegas, 2009) também têm estado entre outras funcionalidades apresentadas.

4.1 Exposição e Avaliação Relacionadas ao Conteúdo

Um elemento presente na maioria das ferramentas encontradas é o fato de possuírem um módulo de exposição ou avaliação acerca do conteúdo, ou ao menos estarem inseridas em um contexto maior com essa finalidade. Mota e colegas (2009), por exemplo, trouxeram em sua ferramenta exemplos de programação e estratégias didáticas. O PROGTUTOR apresentado em Neto e Schuvartz (2007) possui tanto definições e exemplos quanto exercícios.

4.2 Simulação de Execução de Programas

Mota e colegas (2009) apresentaram em sua estrutura o JavaTool uma ferramenta que simula graficamente um programa submetido a ele. Mostra-se o código fonte, uma animação criada automaticamente e um console. Em Souza (2009) é exposto o VisuAlg, que interpreta um programa em uma linguagem pseudo-algorítmica, corrige-o e simula sua execução passo a passo, explicitando variáveis e escopo, entre outras informações.

4.3 Mecanismos de Correção de Código

Barros e colegas (2004) mostraram um mecanismo em que a partir da submissão de um código, instruções relacionadas a erros e mau uso de padrões eram dadas. Mota e colegas (2009) apresentaram um corretor que fazia duas coisas: analisava a complexidade do código baseando-se em um método estatístico e testava a correção da solução usando testes de entrada e saída.

5. Dinâmica de Ensino/Aprendizagem no TutorICC

O sistema adotado para a disciplina ICC é o semi-presencial, em que o aluno deve comparecer apenas para fazer as provas. Todo o seu estudo e treinamento em programação é feito via internet, utilizando o TutorICC.

As provas consistem em programas que o aluno deve desenvolver presencialmente no laboratório da Universidade. Esses programas são corrigidos por um programa corretor de provas automático, que foi desenvolvido juntamente com o TutorICC. Ao longo do semestre são aplicadas provas a cada quinze dias, para evitar o acúmulo de matéria.

O aluno estuda o conteúdo da disciplina por meio de exemplos de programas, que são apresentados num visualizador gráfico. O visualizador apresenta na tela, de forma gráfica, as variáveis que estão sendo utilizadas, os arquivos, as entradas e saídas do programa, etc., para cada passo do programa. O aluno percorre cada programa passo a passo, observando a representação gráfica. Este visualizador também foi desenvolvido como parte do TutorICC.

Os exemplos de programas que fazem parte do conteúdo da disciplina estão divididos em vários níveis de dificuldade. Depois que o aluno estuda um determinado programa, o TutorICC apresenta a ele um problema análogo ao que ele estudou. Ele elabora o programa numa interface disponibilizada pelo TutorICC, e em seguida o envia ao TutorICC. O TutorICC corrige o programa do aluno, utilizando o mesmo corretor automático utilizado para a correção de provas, e dá uma nota ao aluno. Juntamente com a nota, o TutorICC pode recomendar ao aluno que vá para o próximo item da disciplina, ou que reveja os mesmos conceitos em exemplos disponíveis com um menor grau de dificuldade. O TutorICC apenas recomenda, mas o aluno fica livre para escolher sua própria trajetória no conteúdo da disciplina.

Desta forma, o ritmo de aprendizagem pode se adaptar ao perfil do aluno. Um aluno com mais facilidade para aprender programação pode caminhar mais rapidamente pelo conteúdo da disciplina, enquanto outro pode traçar uma trajetória mais lenta e gradual, através de uma maior quantidade de exemplos. O sistema é altamente interativo, pois a cada novo conceito o aluno é convidado a desenvolver um programa e a submetê-lo à correção, recebendo imediatamente a nota obtida e uma recomendação sobre como prosseguir no seu curso.

6. Recursos do TutorICC

Neste tópico são apresentados os principais recursos do TutorICC: o visualizador gráfico de programas, a distribuição do conteúdo em níveis de dificuldade e o sistema de correção automatizada de programas e de provas.

6.1. Visualizador gráfico de programas

O visualizador gráfico de programas tem a aparência mostrada na Figura 1.

Na parte esquerda do aparece o código do programa que está sendo estudado. O aluno pode percorrer o código linha a linha. As linhas são numeradas para facilitar o aluno a se localizar, e uma faixa em *high ligth* mostra qual o comando que está sendo simulado no momento.

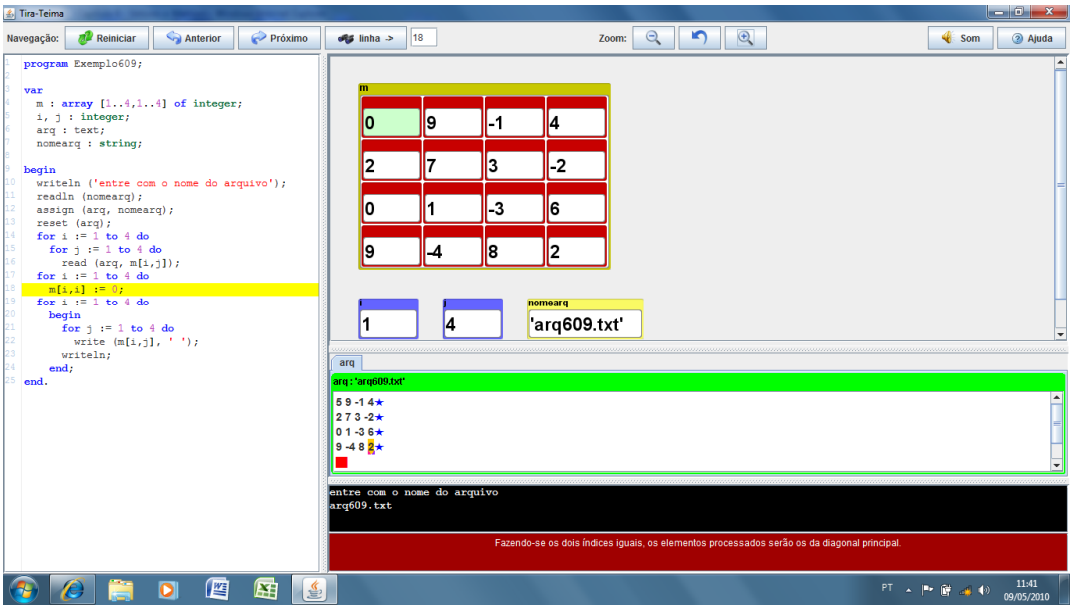

Figura 1. Visualizador Gráfico de Programas

O lado direito da tela é dividido em quatro partes: a primeira mostra cada uma das variáveis que estão sendo utilizadas pelo programa. Na Figura 1 há uma matriz de 4 x 4 inteiros, além de outras variáveis simples. A segunda parte da tela, do lado direito, mostra os arquivos que estão abertos no momento. No exemplo apresentado há apenas um arquivo aberto, contendo 16 números inteiros. A terceira parte da tela, do lado direito, faz uma simulação do console enquanto o programa é executado, mostrando os valores de entrada e de saída do programa. A quarta parte da tela é uma área reservada para explicações que o professor queira dar ao aluno em determinadas linhas de código. Essa janela, em geral, fica fechada. Ela só é aberta nos pontos em que o professor julga necessário fazer um comentário.

Pode-se representar qualquer estrutura de dados estática: variáveis simples como *integer*, *real*, *boolean*, *string* e *char*, ou compostas, como *arrays* e *records*, com todas as suas combinações possíveis: um *array* dentro do *record*, um *array* de *records*, etc.

Para cada exemplo o professor pode adaptar o desenho das variáveis, escolhendo o tamanho, cor e posição de cada uma, de modo a tornar a apresentação mais didática.

O visualizador foi desenvolvido conjuntamente com o restante do TutorICC, utilizando-se a linguagem Java. Sua principal contribuição em relação aos demais trabalhos encontrados na literatura, é que permite que o professor customize a apresentação visual em cada exemplo. Pode escolher posição, tamanho e cor de cada uma das variáveis, dispondo-as da forma que achar mais didática para captar a atenção dos alunos. Além disso, pode inserir comentários em determinados pontos do programa.

6.2 Correção automatizada de programas e de provas

Foi desenvolvido em conjunto com o TutorICC um sistema de correção de programas. Consiste em um programa escrito em Java, que recebe o programa do aluno, escrito em Pascal, o compila e o executa. O professor prepara de antemão, para cada programa a ser corrigido, uma bateria de testes, com valores de entrada e os correspondentes valores de saída esperados. Outras soluções semelhantes, como em Mota e colegas (2009), foram apresentadas em trabalhos anteriores.

O programa corretor executa o programa do aluno, colocando como valores de entrada as previstas pelo professor, e recolhendo as saídas do programa do aluno. Compara essas saídas com as esperadas pelo professor e atribui nota 0 ou 10 ao teste. Para cada programa são executados vários testes, de modo que a nota possa se espalhar entre 0 e 10.

Este programa corretor está sendo utilizado em duas situações. A primeira é dentro do próprio TutorICC, para interagir com o aluno, verificando se ele está aprendendo, para fazer as recomendações convenientes quanto ao próximo passo a ser dado no conteúdo da disciplina.

A segunda situação é a correção das provas. Os alunos fazem provas quinzenalmente, utilizando o laboratório da Universidade. O sistema de correção, além de dar a nota, devolve ao aluno um relatório mostrando qual a solução esperada e quais os testes aplicados ao seu programa, bem como seus acertos e seus erros. Desta forma garante-se também uma qualidade de avaliação do aluno, pois o que se verifica é sua capacidade de construir programas.

6.3 Distribuição do conteúdo em níveis de dificuldade

O conteúdo da disciplina foi distribuído em diversos níveis de dificuldade. Apresentam-se os conceitos de forma expositiva, utilizando textos e animações interativas. Somado a isso, foi construída uma classificação de programas, por meio da elaboração de uma listagem de conceitos que se desejam transmitir. Esses conceitos foram agrupados de dois em dois. Cada grupo deu origem a uma árvore de exemplos: na raiz da árvore há um exemplo contendo os dois conceitos, e no nível mais baixo da árvore há exemplos contendo um, ou o outro, ou eventualmente nenhum dos conceitos.

Se o aluno compreende bem o exemplo da raiz então ele pode passar adiante sem ver os exemplos do nível de baixo da árvore. Se ele tiver dificuldade com o exemplo da raiz, então convém que ele examine os exemplos do nível inferior, nos quais ele encontrará os mesmo conceitos expostos de forma mais lenta e gradual. Todos os exemplos são analisados pelo aluno por meio do visualizador gráfico de programas.

A cada exemplo que o aluno examina, o TutorICC apresenta a ele o enunciado de um problema que exige os mesmos conceitos do exemplo. O aluno escreve o programa numa interface disponibilizada pelo TutorICC, e o submete para correção. Recebe em seguida sua nota acompanhada de uma recomendação, que pode ser a de seguir para novos tópicos ou para a revisão dos mesmos tópicos através de exemplos mais detalhados e mais fáceis.

Com este recurso o aluno pode adaptar a maneira de percorrer o conteúdo da disciplina à sua própria capacidade de aprendizagem.

7. Considerações Finais

O TutorICC agrega recursos que estão sendo estudados e têm sido objeto de atenção por parte da comunidade acadêmica interessada no ensino de programação. Seu grande mérito consiste em ser um programa já implementado e em pleno uso, atendendo aproximadamente 360 alunos da disciplina Introdução à Ciência da Computação, na Universidade de Brasília. Foi usado ao longo de um semestre em uma única turma piloto. Em seguida, foi testado em três turmas. Atualmente está sendo usado em doze turmas, e um único professor coordena todas as turmas, ajudado por monitores que fazem plantão para tirar dúvidas e cuidam da aplicação das provas. Isso tem permitido um melhor aproveitamento dos recursos humanos do Departamento, pois os professores que antes davam aulas de ICC agora podem lecionar disciplinas optativas de sua área especifica.

Embora o TutorICC ainda não tenha passado por uma avaliação rigorosa com turmas de controle, como seria desejável, pode-se dizer que está funcionando bem. No final de cada semestre os alunos avaliam o curso e o TutorICC tem sido bem acolhido pelos alunos. O índice de reprovação na disciplina continua mais ou menos com o mesmo valor apresentado nas turmas presenciais, isto é, aproximadamente 50%. As provas aplicadas nas turmas semi-presencias têm mesmo nível de dificuldade que havia nas turmas presenciais, garantindo assim o mesmo nível de exigência.

Foram e continuam sendo feitas estatísticas de acesso a cada exemplo de aplicação, cujo resultado ajuda a aferir a adequação da distribuição do conteúdo nos vários níveis de dificuldade. Dessa forma pode-se aprimorar a distribuição dos exemplos, em função da experiência de uso.

O TutorICC caminha para se transformar em um STI: o modelo do domínio consiste na distribuição dos conceitos que se quer transmitir pelos diversos exemplos, divididos em vários níveis de dificuldade. O modelo do aluno é dado pelo percurso que o aluno vai traçando dentro do conteúdo geral da disciplina. O modelo do tutor consiste na estratégia de interação e avaliação do aluno a cada exemplo, que permite ao TutorICC aconselhar ao aluno a direção a ser tomada. Pretende-se futuramente enriquecer a base de exemplos (atualmente há 170 exemplos funcionando com visualizador gráfico), e reconstruir a relação entre esses exemplos (grafo de percursos possíveis) a partir da experiência adquirida nos primeiros semestres de teste.

Todo os módulos do TutorICC foram desenvolvidos no Departamento de Ciência da Computação da UnB, utilizando a linguagem Java, para construir o

visualizador gráfico de programas e o corretor automático de programas. Para o corpo do TutorICC utilizou-se PHP e JavaScript.

Referências

Anderson, J.R., Reiser, B. J. (1985) The Lisp Tutor. Byte, v. 10, n. 4, pp. 159-175.

Barros, L. N.; Delgado, K. V.; Machion, A. C. (2004) ITS for Programming to Explore Practical Reasoning. In: XV Simpósio Brasileiro de Informática na Educação. SBC.

Castro, T. H. C., Castro Júnior, A. N., Menezes, C. S. (2004) Aprende – um Ambiente Cooperativo de Apoio à Aprendizagem de Programação. UFAM. UFES. In: XV Simpósio Brasileiro de Informática na Educação. SBC.

Mota, M. P., Brito, S. R., Moreira, M. P., Favero, E. L. (2009) Ambiente Integrado à Plataforma Moodle para Apoio ao Desenvolvimento das Habilidades Iniciais de Programação. In: XX Simpósio Brasileiro de Informática na Educação. SBC.

Neto, W. C. B., Schuvartz, A. A. (2007) Ferramenta Computacional de Apoio ao Processo de Ensino-Aprendizagem dos Fundamentos de Programação de Computadores. In: XVIII Simpósio Brasileiro de Informática na Educação. SBC.

Pereira Júnior, J. C., Rapkiewicz, C. E. (2004) O Processo de Ensino-Aprendizagem de Fundamentos de Programação: Uma Visão Crítica da Pesquisa no Brasil. WEI, I : 2004 nov. 19-21 : Vitória - ES, Rio das Ostras – RJ.

Santos Júnior, G. P. S., Fechine, J. M., Costa, E. B. (2009) Analogus: Um Ambiente para Auxílio ao Ensino de Programação Orientado pelo Raciocínio por Analogia. WEI, XVII: 2009 jul. 20-22 : Bento Gonçalves – RS.

Self J. (1999) The defning characteristics of intelligent tutoring systems research: ITSs care, precisely. International Journal of Arti_cial Intelligence in Education, 10(3-4):350 364.

Sleeman, D. (1982) Assessing aspects of competence in basic algebra. In: Intelligent tutoring systems. Academic Press, 1982.

Souza, C. M. (2009) VisuAlg - Ferramenta de Apoio ao Ensino de Programação. Revista TECCEN, Vassouras-RJ, v. 2, n. 2, setembro. Universidade Severino Sombra.

Tobar, C. M., Rosa, J. L. G., Coelho, J. M. A., Pannain, R. (2001). Uma Arquitetura de Ambiente Colaborativo para o Aprendizado de Programação. In: XII Simpósio Brasileiro de Informática na Educação

Viccari R. M.; Giraffa, L.M.M. (2003) Fundamentos dos Sistemas Tutores Inteligentes. Capítulo do Livro: Sociedades Artifciais. Porto Alegre – RS. Bookman.