AUXILIANDO A APRENDIZAGEM DE ALGORITMOS COM A FERRAMENTA WEBPORTUGOL

Higor Hostins[1], André Raabe[2,3]

[1]Bacharelado em Ciência da Computação
[2]Mestrado em Computação Aplicada
[3]Mestrado Acadêmico em Educação
UNIVALI - Universidade do Vale do Itajaí
Rua Uruguai, 458 – 88302-202 – Itajaí – SC – Brasil
{higorhostins, raabe}@univali.br

Abstract. *This paper presents the development and use of WebPortugol tool which aims to aid students to develop the programming logic at initial computer science courses. The tool was developed inside an applet Java and can be used directly through the browser. An algorithm verification mechanism was included in the tool for testing pre-defined input and output values. This allowed the identification of the developed solutions correctness and fostered students to initiate a debug process*

Resumo. *Este artigo apresenta o desenvolvimento e a utilização da ferramenta WebPortugol que busca auxiliar na construção da lógica de programação nas disciplinas iniciais de cursos da área de Computação. A ferramenta foi desenvolvida em um applet Java e pode ser utilizada através de um navegador Internet. Foi incluído um recurso de verificação dos algoritmos baseado em valores de entrada e saída pré-definidos que possibilitou a identificação da conformidade das soluções desenvolvidas e com isso incentivou os alunos a iniciarem a depuração das suas soluções.*

1. Introdução

A aprendizagem de algoritmos é considerada fundamental em cursos da área computacional. Seu principal objetivo é iniciar o desenvolvimento da lógica de programação, que será amplamente utilizada durante o restante do curso. Porém, a disciplina de algoritmos é considerada desafiadora para a maioria dos alunos em virtude de possuir um alto índice de problemas de aprendizagem, desistências e reprovações (Rodrigues, 2004).

Uma das abordagens mais comuns para enfrentar este problema é o desenvolvimento de ferramentas computacionais que possam fornecer auxílio ao aluno na construção da lógica algorítmica e ao professor na organização e ampliação da capacidade de atendimento aos problemas de aprendizagem.

Neste sentido, este artigo apresenta o desenvolvimento e aplicação da ferramenta WebPortugol, cujo objetivo é auxiliar no desenvolvimento de algoritmos em pseudo-linguagem (português estruturado). Buscamos desenvolver uma solução própria, ao invés de adotar uma existente, para possibilitar a integração com o Sistema Tutor Inteligente que já vem sendo utilizado em nossa universidade e também para colocar a

prova uma idéia simples explorada por Miranda (2004) e que possibilita ao aluno realizar a verificação de conformidade do algoritmo desenvolvido com base em valores pré-estabelecidos de entrada e saída.

Este artigo está organizado da seguinte forma: a seção 2 detalha o contexto e a motivação para construção da ferramenta, a seção 3 apresenta a ferramenta WebPortugol e detalha aspectos computacionais de sua construção, a seção 4 analisa os resultados já obtidos e a seção 5 apresenta conclusões e perspectivas futuras.

2. Contexto e Motivação

A construção de ferramentas para apoio a aprendizagem de programação é talvez uma das áreas de pesquisa com maior número de contribuições dentro da Ciência da Computação. É possível identificar duas explicações para esta popularidade: (i) o reconhecimento do alto índice de problemas de aprendizagem entre os aprendizes de lógica de programação motivando a construção de ferramentas e metodologias que possam auxiliar a reduzir estes problemas; (ii) o fato de que muitos dos trabalhos de tecnologia educacional e em especial de sistemas tutores inteligentes (STI) serem desenvolvidos por Cientistas da Computação que tem alguma ligação direta ou indireta com a disciplina e buscam domínios bem formalizados para a aplicação de teorias e técnicas inovadoras Kinshuk (2002).

A pesquisa na área não é recente, e em geral os trabalhos focalizam na construção de ferramentas para apoio a prática com programação em pseudocódigo (Johnson et al, 1987; Esmin, 1998; Evaristo e Crespo, 2000; Santiago e Dazzi, 2004) ou por meio de animações e representações gráficas como fluxogramas (Brusilovsky, 1994; Bently e Kerninghan 1991; Dazzi et al., 2004). A construção de sistemas tutores inteligentes busca personalizar o atendimento aos aprendizes de programação como em (Du Boulay e Sothcott, 1987; Vicari, 1989; Song, 1997; El-Khouly et al, 2000; Almeida et al., 2002; Castro et. Al., 2002;). Também existem abordagens que focalizam a construção de ferramentas com enfoque nos aspectos de interação e usabilidade dos editores de programação (Pane e Myers, 1996; Guilbert e Girard, 2003) e ainda ambientes que promovem a organização da aprendizagem por meio de abordagens metodológicas diferenciadas (Menezes e Nobre, 2002; Giraffa et al., 2003, Pimentel et. al., 2003; Raabe et al. 2005).

É possível observar nos trabalhos desenvolvidos pela comunidade acadêmica brasileira que existe uma tendência para o uso de pseudo-linguagens com palavras reservadas em idioma português para a introdução das noções fundamentais de programação. Esta tendência também é observada nos livros didáticos utilizados nas disciplinas introdutórias e tem sido adotada para permitir que a aprendizagem esteja focalizada nos aspectos da lógica de programação sem se deter em detalhes específicos de sintaxe das linguagens de programação e reduzindo também a barreira do idioma estrangeiro utilizado nas linguagens de programação.

Grande parte do esforço dos desenvolvedores destas ferramentas está em possibilitar ao aluno um ambiente para experimentação das soluções desenvolvidas de forma que este possa expressar uma hipótese de solução, testá-la e depurá-la. Intrinsecamente esta abordagem busca auxiliar o aluno a desenvolver autonomia para construir algoritmos sem a dependência da correção do professor, habilidade esta que será fundamental nas demais disciplinas da área de programação.

Neste contexto, o grupo dos autores deste trabalho definiu duas linhas de atuação para apoiar a aprendizagem dos alunos da disciplina introdutória de algoritmos: (i) a construção de ferramentas para apoio a realização de testes de algoritmos em português estruturado e fluxogramas; (ii) a construção de sistemas tutores inteligentes (STI) para identificar o perfil de aprendizagem dos alunos e fornecer atividades adequadas a este perfil;

Na primeira linha de atuação inicialmente o grupo definiu a gramática do português estruturado (chamado de portugol) e uma notação correspondente em fluxograma. A partir dos primeiros experimentos de utilização destas notações, decidiu-se estender o fluxograma tradicional para reduzir alguns conceitos equivocados dos alunos. Desde então foram desenvolvidas três ferramentas: (i) CIFluxProg: Ferramenta para construção e interpretação de algoritmos utilizando fluxogramas e portugol (Dazzi, et al. 2004); (ii) Happy Portugol: ferramenta para teste e conversão de programas portugol para linguagem C (Fischer, 2005); (iii) Compilador BIP: Ferramenta para teste e conversão do portugol no *assembly* do processador hipotético BIP II (Morandi et al, 2006). Estas ferramentas desenvolvidas buscaram também promover um relacionamento interdisciplinar com outras disciplinas do curso de Ciência da Computação tais como Arquitetura de Computadores e Programação.

Na segunda linha de atuação, foi desenvolvido um STI acessível via Internet fundamentado em uma abordagem híbrida onde o professor e o sistema cooperam para atender melhor aos alunos. Esta abordagem chamada de ITA (*Intelligent Teaching Assistant*) foi utilizada seguindo princípios da teoria da Mediação de Feuerstein (1998) e vem apresentando resultados positivos na melhoria da atenção aos alunos e na ampliação do empenho destes em situações extra-classe (Raabe e Giraffa, 2006).

A motivação para desenvolvimento do trabalho apresentado neste artigo é a de coletar informações sobre o processo de desenvolvimento dos algoritmos em portugol pelos alunos a fim de utilizá-las no processo decisório do sistema tutor inteligente. Desta forma, identificou-se a necessidade de desenvolver uma nova ferramenta que pudesse ser utilizada de forma integrada com o STI e que fornecesse a este informações sobre a interação com o aluno.

Além disso, buscou-se criar uma ferramenta que permitisse um maior grau de autonomia ao aluno fornecendo recursos de interface que reduzem os enganos iniciais e que possibilitassem acompanhar detalhadamente a execução do programa, além de incluir um mecanismo simples de verificação da corretude da solução, inspirado no trabalho de Miranda (2004).

3. A Ferramenta WebPortugol

Conforme a motivação para o desenvolvimento da ferramenta os requisitos foram definidos sendo os principais os seguintes: (i) a ferramenta deve ser totalmente acessível via navegador Internet para viabilizar a integração como o STI; (ii) deve possibilitar a edição e testes de programas escritos na pseudo-linguagem portugol (iii) deve possibilitar a execução do algoritmo, passo a passo, ilustrando as variáveis utilizadas; (iv) deve apresentar mensagens de erro sintático em português com exemplos associados ilustrando a correta utilização das construções da linguagem (v) deve salientar as construções sintáticas válidas durante a edição (*syntax highlight*) (vi) deve permitir a disponibilização de questões compostas de um enunciado e um conjunto de

grupos de testes pré-definidos e (vii) deve realizar os testes pré-definidos e informar o aluno do sucesso ou falha, e neste caso quais valores geraram as falhas.

Além destes requisitos, desejava-se construir uma ferramenta com interface simples contendo apenas as operações necessárias para desenvolvimento da lógica de programação utilizando o portugol. A Figura 1 ilustra a interface do WebPortugol após a verificação de uma solução para desenvolvimento de um algoritmos de fatorial.

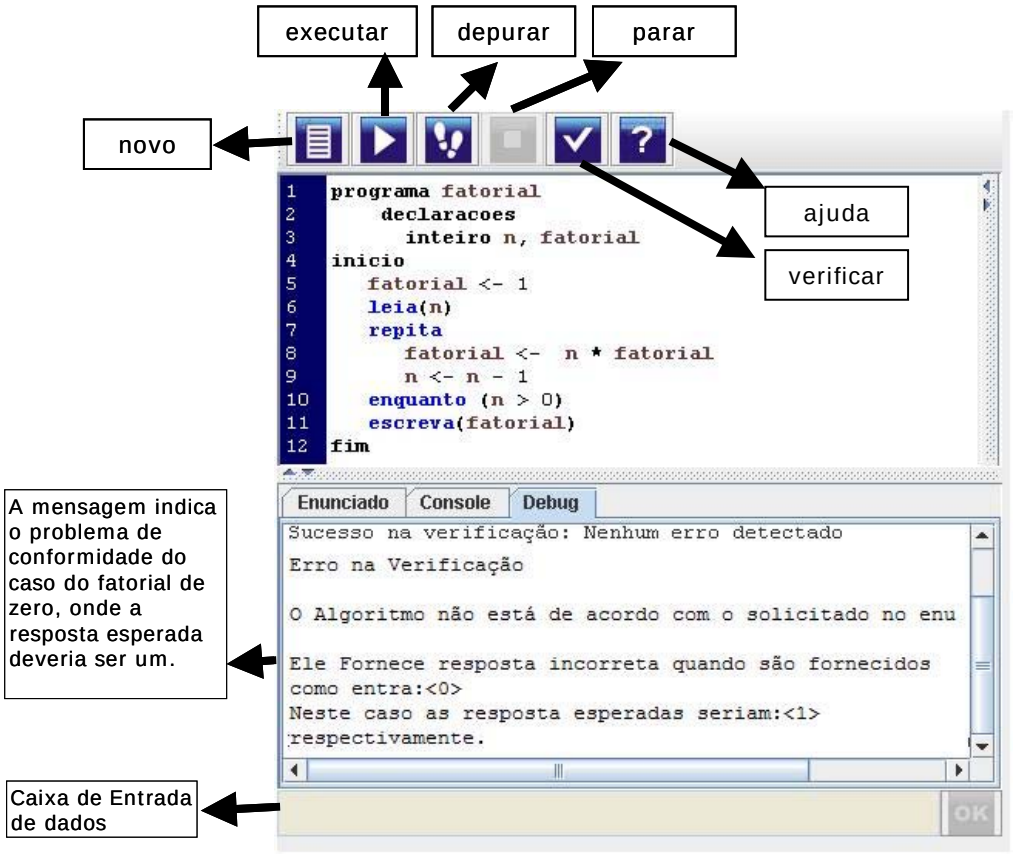

Figura 1 – Salienta as funcionalidades do WebPortugol.

O sistema pode ser usado livremente para construção de algoritmos ou então com enunciados de problemas a serem solucionados, onde torna-se possível realizar a verificação de grupos de teste previamente definidos.

A integração com o STI se dá por meio do envio de informações em duas situações distintas: (i) o aluno concluiu a questão e os testes de verificação com sucesso; (ii) o aluno desistiu de realizar a questão. As variáveis que são enviadas ao STI são: número de erros sintáticos, número de execuções, número de depurações (execuções passo a passo), tempo de desenvolvimento e grupos de teste verificados com sucesso. A utilização destas informações passam a compor o modelo do aluno no STI, no entanto o processo decisório do STI relativo à inclusão destas novas variáveis é um trabalho em andamento e deverá ser detalhado em trabalhos futuros.

3.1 Aspectos computacionais do desenvolvimento

A construção do WebPortugol seguiu o roteiro tradicional do desenvolvimento de compiladores. Foi desenvolvido um analisador léxico utilizando expressões regulares

para reconhecimento dos tokens[1] do portugol, um analisador sintático com o algoritmo LALR[2] e um analisador semântico cujas principais tarefas foram a verificação de tipos, construção da tabela de símbolos e a criação de uma árvore sintática abstrata (ASA). A ASA foi escolhida como representação intermediária do código gerado pois possibilita a interpretação deste utilizando um algoritmo simples para percorrê-la.

Para permitir a integração com o navegador, a ferramenta foi construída dentro de um *Applet*[3] da linguagem Java. Isto direcionou a construção da ASA através de objetos, por ser a abordagem mais natural nesta linguagem. Na bibliografia tradicional de compiladores (Aho et al. 1995; Louden 2004; Price e Toscani 2005) os exemplos de ASA são normalmente construídos utilizando estruturas e ponteiros. A solução adotada para a construção da árvore foi implementar uma classe para cada comando (nó) da árvore existente na linguagem construída, uma classe para armazenamento de variáveis e constantes e uma classe para armazenamento de operadores utilizados em expressões. Cada classe gerada é uma linha de execução (*thread*) que permite a execução da mesma em paralelo com outras linhas de execução (*threads*).

A interpretação do código é feita percorrendo a ASA gerada com os objetos, onde cada objeto que compõe a árvore é responsável pela sua própria execução interagindo com a interface. Por exemplo, ao executar uma operação de saída de dados (escreva) o texto correspondente é exibido na região do console da interface. Ao executar uma operação de entrada de dados (leia) o foco da interface é dirigido para a linha de digitação de dados e o valor digitado é capturado e armazenado na tabela de símbolos.

Um dos problemas encontrados nessa abordagem de execução da árvore foi como interromper a execução de uma classe enquanto outra classe está executando. A solução encontrada foi fazer com que assim que uma classe requisita que outra comece a executar, a solicitante entra em estado de espera até que a classe solicitada informe que sua execução terminou. O estado de espera foi implementado por um *loop* onde a *thread* entra em estado de *sleep* por alguns milisegundos e ao acordar verifica se o objeto que estava executando já concluiu a operação.

Uma característica importante da ferramenta é a possibilidade do aluno realizar a verificação da solução baseado em um grupo de testes pré-definidos. Para disponibilizar tal funcionalidade cada questão é definida em um arquivo XML que contém o enunciado e grupos de testes, que são conjuntos de valores de entrada e de saída que permitem testar a conformidade do algoritmo desenvolvido com o enunciado. A figura 2 apresenta um exemplo de questão definida via XML.

---

[1] Token: Unidades de informação léxica tais como variáveis, palavras reservadas, operadores, etc.

[2] LALR – **L**ook**A**head **L**eft to right with **R**ightmost derivation on reverse: algoritmo tradicional de análise sintática ascendente.

[3] *Applet*- Recurso da linguagem Java que possibilita que um navegador Internet exiba uma aplicação e onde o processamento ocorre na máquina cliente.

```
<?xml version="1.0" encoding="ISO-8859-15" ?>
- <query xmlns="questao">
    <enunciado>Solicite ao usuário um número natural e exiba o fatorial deste número.</enunciado>
  - <grupo>
      <entrada>5</entrada>
      <saida>120</saida>
    </grupo>
  - <grupo>
      <entrada>4</entrada>
      <saida>24</saida>
    </grupo>
  - <grupo>
      <entrada>1</entrada>
      <saida>1</saida>
    </grupo>
  - <grupo>
      <entrada>0</entrada>
      <saida>1</saida>
    </grupo>
</query>
```

Figura 2 – Exemplo de questão com grupos de testes em XML

Durante a verificação do algoritmo construído pelo aluno o sistema substitui cada operação de leitura via teclado por uma entrada do arquivo de teste e cada operação de escrita, que contenha variável, será comparada a um elemento de saída do arquivo de teste. Uma vez que todas as entradas tenham sido requisitadas e todas as saídas tenham apresentado os mesmos valores armazenados no XML o programa emite uma mensagem de sucesso na verificação, ou em caso contrário uma mensagem indicando a situação onde o algoritmo falhou.

Houve também a preocupação em seguir algumas das recomendações de Pane e Myers (1996) com relação a construção da interface do editor. Neste sentido, os seguintes recursos foram priorizados:

- Salientador de sintaxe (*Syntax Highlight*): Cada palavra reservada da gramática e toda variável declarada é colorida de uma forma diferente para facilitar a distinção por parte do usuário.
- Entrada de dados: A área de entrada de dados via teclado permanece desabilitada durante grande parte do tempo, tornando-se habilitada apenas durante uma operação de leitura. Quando uma operação do gênero ocorre o foco do sistema é alterado automaticamente para o campo de entrada facilitando assim a digitação do usuário. Ao terminar de digitar o valor o usuário pode simplesmente pressionar a tecla Enter do teclado ou clicar com o mouse sobre o botão OK.
- Divisões da interface: A interface do sistema é dividida em várias áreas, na parte inferior do sistema situa-se a área utilizada pelo sistema para informar o enunciado de questões, console de entrada de dados e saída de informações do código implementado pelo usuário e mensagens de erros (debug).
- Mensagens de erros significativas: Uma das maiores dificuldades para construção da interface de um compilador que busca facilitar a aprendizagem dos usuários menos experientes é emitir mensagens de erros que sejam simples de compreender. Exibir mensagens informando apenas os tokens que poderiam ser usados para substituir o erro é simples, porém, exibir mensagens informando e sugerindo ao usuário o que fazer é um processo difícil e amplamente reconhecido pela comunidade da área, uma vez que o mesmo estado interno de erro do compilador ocorre para diversos erros diferentes. Nossa abordagem foi

simular manualmente os erros mais comuns e inserir mensagem explicativas associadas a exemplos ilustrativos, porém em muitas erros sintáticos a mensagem não ajuda como deveria.

4. Resultados

Foram realizados testes com 40 alunos do primeiro período do curso de Ciência da Computação a fim de diagnosticar possíveis erros, validar a integração com o STI, levantar hipóteses quanto ao uso do sistema desenvolvido e coletar opiniões dos alunos. Durante os testes foi requisitado aos alunos desenvolverem, utilizando o WebPortugol, quatro diferentes algoritmos sendo os dois primeiros de assuntos já conhecidos e exigidos em prova, e os dois últimos de assuntos recém ministrados pelo professor.

Com base nos testes foi possível verificar que a ferramenta atende aos requisitos especificados e as suas funcionalidades estão operando de forma correta. O recurso de interface que mais facilitou o trabalho dos alunos com pouca experiência em programação foi o salientador de sintaxe, pois permitiu aos alunos facilmente identificarem erros de digitação comuns tais como acentuação de palavras chave (ex: então) e também a distinguirem entre os tipos de informações léxicas como palavras reservadas, *strings*, comentários, operadores e assim por diante. As mensagens de erro sintático com exemplos em pouquíssimos casos ajudaram os alunos que dificilmente liam-nas e os divisores da interface (*spliters*) foram constantemente redimensionados em função do espaço reduzido para a janela de código.

Com relação ao desempenho do interpretador, que poderia ser um aspecto crítico em virtude de a solução adotada utilizar linhas de execução (*threads*) e um mecanismo de sincronia entre estas, observou-se que o tempo de resposta não comprometeu os objetivos didáticos da ferramenta e deixou de ser foco de atenção dos desenvolvedores.

A integração com o STI foi feita por meio da inserção das variáveis analisadas (indicadas na seção 3) na base de dados do ambiente. Algumas destas variáveis coletadas durante o experimento também permitiram caracterizar outros aspectos da forma na qual os estudantes utilizaram a ferramenta. A Tabela 1 apresenta estas informações de forma resumida.

Tabela 1. Tabulação de dados do experimento

Questão	Alunos que Realizaram (%)	Alunos que Verificaram (%)	Tempo Médio Despendido	Média de Execuções	Média de Depurações
1	84,85	67,86	8,47	5,68	1,27
2	81,82	44,44	14,43	7,02	1,39
3	51,52	58,82	14,73	8,11	2,98
4	24,24	37,50	15,68	5,87	7,57

Nota-se que os assuntos consolidados foram realizados pela maioria dos estudantes, e que o tempo de solução variou proporcionalmente conforme a complexidade do algoritmo solicitado.

A partir das informações coletadas vislumbrou-se a possibilidade de realizar testes de hipóteses que pudessem explicar melhor alguns dos aspectos ligados ao benefício do uso da ferramenta. Quatro hipóteses foram levantadas:

1. As verificações que obtiveram resultados mal sucedidos levaram os alunos a uma depuração passo a passo da solução.

2. As soluções bem sucedidas reduziram o tempo de construção da solução pelos alunos.
3. Os alunos que verificaram as soluções depuraram passo a passo mais dos que os que não verificaram.
4. Os alunos que verificaram as soluções dedicaram mais tempo as questões dos que os que não verificaram.

Para verificação das hipóteses foram utilizados dois procedimentos estatísticos diferentes. Para as duas primeiras hipóteses foram realizados testes de correlação com distribuição de Student com grau de confiança de 95% e para as duas ultimas hipóteses foi utilizado o Teste Z. O objetivo do Teste Z é a comparação entre duas médias ou proporções.

Dentre as quatro hipóteses apenas foi possível aceitar a primeira, mostrando que as verificações com resultados mal sucedidos levaram os alunos a utilizar a funcionalidade de depuração passo a passo. As demais hipóteses não puderam ser comprovadas pelos procedimentos estatísticos adotados.

5. Conclusões

Este trabalho apresentou o desenvolvimento e resultados iniciais do uso da ferramenta WebPortugol, para auxílio a aprendizagem de lógica de programação. A ferramenta é parte integrante de um esforço de nosso grupo de pesquisa para melhorar o atendimento aos problemas de aprendizagem dos alunos que iniciam os cursos de Ciência da Computação. A ferramenta pode ser utilizada livre de custos na URL http://www.univali.br/webportugol, bastando para isso preencher um cadastro.

A característica que apresentou maiores benefícios na ferramenta é o verificador dos algoritmos. Ela possibilitou aos alunos ampliar o grau de autonomia durante o desenvolvimento das soluções, pois permitia que algum teste com valores pré-definidos fosse realizado, a exemplo do que muitas vezes faz o professor em sala de aula. Entende-se que esta funcionalidade não é suficiente para detectar e orientar todos os tipos de erros que os alunos cometem, mas certamente fornece alguma referência para identificação da conformidade da solução. No experimento realizado foi possível identificar a tendência dos alunos entrarem em um processo de depuração da solução após identificarem algum caso de erro na verificação. Isto reforça a nossa crença de este tipo de recurso possa apoiar o desenvolvimento da autonomia e da construção do conhecimento.

As perspectivas futuras deste trabalho incluem a reformulação do mecanismo de decisão do STI a fim de comportar as novas informações sobre o processo de desenvolvimento de soluções pelo aluno e também o desenvolvimento de novos experimentos em maior escala para ampliar a confiança sobre o indício de que o recurso de verificação proporciona maior autonomia.

6. Referências

AHO, Alfred V; SETHI, Ravi; ULLMAN, Jeffrey D.. Compiladores, princípios, técnicas e ferramentas. Rio de Janeiro: Guanabara Koogan, c1995.

ALMEIDA, E.; COSTA, E.; SILVA, K.; PAES, R.; ALMEIDA, A.; BRAGA, J. AMBAP: um ambiente de apoio ao aprendizado de programação. Workshop de Educação em Computação, Congresso anual da SBC, Florianópolis, 2002.

BRUSILOVSKY, P.. Program visualization as a debugging tool for novices. In: Proc. of INTERCHI'93 .Pp. 29-30. Amsterdam, 1994.

BENTLY, J. L.; KERNINGHAN, B. W. A system for algorithm animation. Computing Systems. Vol 4. No. 1, 1991.

BUTZ, C.; HUA, S.; MAGUIRE, B. A web-based intelligent tutoring system for computer programming. IEEE/WIC/ACM Conference on Web Intelligence (WI04), 2004.

CASTRO, T.; CASTRO JÚNIOR, A.; MENEZES, C.; CURY, D.. Arquitetura SAAP: Sistema de Apoio à Aprendizagem de Programação. Workshop de Educação em Computação, Congresso da Sociedade Brasileira de Computação, Florianópolis, 2002.

DAZZI, R. L. S.; SANTIAGO, Rafael de ; JESUS, Elieser Ademir de. Construtor e Interpretador de Fluxogramas - Uma Ferramenta de Ensino. In: Construtor e Interpretador de Fluxogramas - Uma Ferramenta de Ensino, 2004, Caceres-Espanha. VI Simposio Internacional de Informática Educativa (SIIE 2004), 2004.

DU BOULAY, B.; SOTHCOTT, C. Computers teaching programming: an introductory survey of the field. Artificial Intelligence in Education: learning environment and tutoring systems, v. 6, 1987

EL-KHOULY, M. M.; FAR, B. H.; KOONO, F. Z. Expert tutoring system for teaching computer programming languages. Expert System with Applications, New York, N. 18, 2000.

ESMIN, A. A. A. Portugol/Plus: Uma Ferramenta de Apoio ao Ensino de Lógica de Programação Baseado no Portugol. In: IV Congresso RIBIE, Brasília, 1998.

EVARISTO, J.; CRESPO, S. Aprendendo a programar: programando numa linguagem algorítmica executável (ILA). Rio de Janeiro: Book Express, 2000.

FEUERSTEIN, R. The Theory of Mediated Learning Experience: About The Human as a Modifiable Being. Ministry of Defense Publications, Jerusalem, 1998.

FISCHER, Marcos Roberto. Gerador de Código para o Happy Portugol. Trabalho de Conclusão de Curso. (Bacharelado em Ciência da Computação) - Universidade do Vale do Itajaí. 2006.

GIRAFFA, L.; MARCZAK, S.; ALMEIDA, G. O Ensino de algoritmos e programação mediado por um ambiente Web. Workshop de Educação em Computação, Congresso Anual da Sociedade Brasileira de Computação, Campinas, 2003.

GUILBERT, N. GIRARD, P. Teaching and learning programming with a programming by example system. International Symposium of End User development, Germany, 2003.

JOHNSON, W; SOLOWAY, E. PROUST: An automatic debugger for Pascal programs. Artificial Intelligence in Education: applications and methods. Addison Wesley. 1987.

LOUDEN, Kenneth C. Compiladores: princípios e práticas. São Paulo: Pioneira Thomson, 2004.

MENEZES, C. S.; NOBRE, I. A. M. Um ambiente cooperativo para apoio a cursos de introdução a programação. In: WORKSHOP DE EDUCAÇÃO EM COMPUTAÇÃO, 5, 2002, Florianópolis. Anais do Congresso da Sociedade Brasileira de Computação. Porto Alegre: SBC, 2002.

MIRANDA, E. M. Uma ferramenta de apoio ao processo de aprendizagem de algoritmos. Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Santa Catarina, Florianópolis, 2004.

MORANDI, Diana; RAABE, André Luís Alice ; ZEFERINO, Cesar Albenes. Processadores para Ensino de Conceitos Básicos de Arquitetura de Computadores. In: Workshop sobre Educação em Arquitetura de Computadores - WEAC 2006, 2006, Ouro Preto - MG. Anais do Workshop sobre Educação em Arquitetura de Computadores, 2006. v. 1. p. 17-24.

PANE, J. F.; Myers B. A. Usability Issues in the Design of Novice Programming Systems. Human-Computer Interaction Institute Technical Report CMU-HCII-96-101, 1996.

PIMENTEL, E.; FRANÇA, V.; NORONHA, R.; OMAR, N. Avaliação Contínua da Aprendizagem, das Competências e Habilidades em programação de Computadores. Workshop de Informática na Escola, Congresso Anual da Sociedade Brasileira de Computação, 2003.

PRICE, Ana Maria de Alencar; TOSCANI, Simao Sirineo. Implementação de linguagens de programação: compiladores. Porto Alegre: Sagra Luzzatto, 2005.

RAABE, A. L. A.; SILVA, J.M; GIRAFFA, L.M.M Um Ambiente EaD para promover Experiências de Aprendizagem Mediadas em uma Disciplina Presencial, Revista Informática na Educação Teoria e Prática, V.8, N.1, Porto Alegre, 2005.

RAABE, A. L.; GIRAFFA, L. M. M. Uma Arquitetura de Tutor para Promover Experiências de Aprendizagem Mediadas. Simpósio Brasileiro de Informática na Educação, SBIE 2006, Brasília, 2006.

RODRIGUES Jr., M. C. Experiências positivas para o ensino de algoritmos. Workshop de Educação em Computação e Informática, Salvador, 2004. Disponível em: <http://www.uefs.br/erbase2004/documentos/weibase/Weibase2004Artigo001.pdf>. Acesso em: jan. 2005.

SANTIAGO, R.; DAZZI, R. Ferramenta de apoio ao ensino de algoritmos. In: SEMINÁRIO DE COMPUTAÇÃO - SEMINCO, 13, 2004, Blumenau. Anais..., Blumenau, 2004.

SONG, J. et al. An Intelligent Tutoring System for Introductory C Language Course, Computers & Education Magazine, V. 28, N. 2, 1997.

VICARI, Rosa Maria. Um tutor inteligente para a programação em lógica: idealização, projeto e desenvolvimento. Coimbra: Tese de doutorado, Universidade de Coimbra, Coimbra, 1989.