Avaliação e evolução de um ambiente de suporte à aprendizagem de programação

Anabela Gomes, António José Mendes, Maria José Marcelino

AVALIAÇÃO E EVOLUÇÃO DE UM AMBIENTE DE SUPORTE À APRENDIZAGEM DA PROGRAMAÇÃO

Gomes A. J.[1], Mendes A. J.[2], Marcelino M.J.[2]

[1] Instituto Superior de Engenharia de Coimbra (Portugal)
anabela@isec.pt
[2] Departamento de Engenharia Informática. Universidade de Coimbra (Portugal)
{toze, zemar}@dei.uc.pt

Resumo

O ensino da programação através dos métodos tradicionais, onde a leccionação de conceitos dinâmicos é realizada utilizando principalmente materiais de índole estática, não se tem revelado muito eficaz. Neste documento é apresentado um ambiente, o SICAS, que de forma dinâmica, interactiva e apelativa permite a construção e simulação de algoritmos. De forma a medir a eficácia do ambiente desenvolvido e a procurar formas de o melhorar foram realizadas experiências. As sugestões chegadas por avaliadores e a constatação, por parte dos autores, de certos aspectos menos bem conseguidos resultaram numa proposta de alteração, cujo desenvolvimento está em curso.

1. Introdução

O insucesso generalizado verificado na aprendizagem de programação, por parte de alunos de qualquer grau de ensino é uma realidade, constituindo este aspecto um problema mais grave quando os alunos frequentam um curso superior de informática, cuja essência reside na capacidade de programar.

Existem opiniões variadas e por vezes divergentes no que respeita ao motivo para tal insucesso. Muitos dos professores que leccionam este tipo de matérias consideram que os fracos resultados se devem à falta de empenho e dedicação dos alunos. No entanto, muitos deles, também constatam que há alunos que batalham arduamente por aprender programação mas que nunca conseguem atingir este objectivo. Jenkins (2002) apresenta um conjunto de factores cognitivos apontados como causa. A dificuldade em programar reside, segundo Jenkins (2001), no facto de a programação não consistir numa habilidade única, nem apenas num conjunto de qualificações, mas antes numa hierarquia de aptidões. Bereiter e Ng. (1991) explicam estas hierarquias desde um nível mais baixo até à sua progressão no sentido de atingir o topo (verdadeira capacidade de programação). Para além disso os autores indicam que a programação reside num conjunto de múltiplos processos, cujo mais difícil consiste na tradução da especificação do problema para o algoritmo. Referem ainda que, a partir da concepção de um algoritmo correcto, o processo de codificação é essencialmente mecânico.

Gomes e Mendes (1998) apresentam também um conjunto de razões que, nas suas opiniões, constituem um entrave para que estes problemas não tenham sido ultrapassados através dos

métodos de ensino tradicionais. Os autores salientam que o problema principal reside na incapacidade de os alunos resolverem problemas, apresentarem soluções (algoritmos), ou seja, a dificuldade reside na concepção e formalização de uma solução para um determinado problema e não na sua codificação. Gomes e Mendes (1999) fazem também uma breve exposição sobre eventuais alternativas e sistemas complementares aos métodos de ensino tradicionais, que possam apoiar o aluno no seu processo de aprendizagem de programação. Apresentam uma resenha de várias ferramentas computacionais concebidas para o efeito ao longo dos anos. No entanto, apesar de se encontrarem relatos da obtenção de resultados positivos na sequência da utilização de algumas ferramentas, a verdade é que nenhuma delas tem uma utilização generalizada. Além disso, pelos estudos efectuados, nenhuma das ferramentas disponíveis se adequava ás exigências que a aprendizagem de programação apresenta, nomeadamente a possibilidade de criar e simular algoritmos. Surgiu então a ideia de conceber um sistema educativo – SICAS, que fornece um ambiente que pode ajudar os alunos não apenas a compreender o funcionamento de um algoritmo, mas que sobretudo lhes permita realizar, testar, experimentar, alterar e corrigir os seus próprios algoritmos.

Pensamos também que o sucesso de um ambiente de aprendizagem depende não apenas das suas funcionalidades e potencialidades, mas essencialmente da forma como interage e comunica com o utilizador. Assim, o SICAS apresenta não uma abordagem expositiva dos assuntos a ensinar/aprender, em que o aluno recebe a informação passivamente, mas antes exibe um ambiente que permite que os alunos desenvolvam os seus algoritmos com base na experimentação e na prática. Esse ambiente permite ao aluno a construção, observação, análise e simulação visual da forma como um dado algoritmo funciona ou se comporta mediante certas circunstâncias, possibilitando-lhe detectar eventuais erros, corrigi-los e através disso aprender ao seu próprio ritmo.

Ao longo dos anos têm sido efectuados diversos estudos com o intuito de verificar a eficácia da utilização de animações, na compreensão e resolução de problemas computacionais. A maioria deles constatou a valiosa contribuição do apelo visual dinâmico para a formação de conceitos, em detrimento da apresentação de imagens estáticas, e uma vantagem significativa da animação com elevados níveis de interactividade. Também pensamos que a mais valia do SICAS reside na possibilidade de simular, de forma animada e interactiva as resoluções algorítmicas construídas. Em Stasko et al. (1993), Ramani y Rao (1994), Hih y Alessi (1994) e Kann y al. (1997) são

apontados alguns estudos que indicam que as animações, para serem eficazes, têm de permitir interacção e vários níveis de controlo pelo aluno, salientando que, para o tipo de matérias referidas é vantajoso utilizar meios animados em detrimento de materiais não animados.

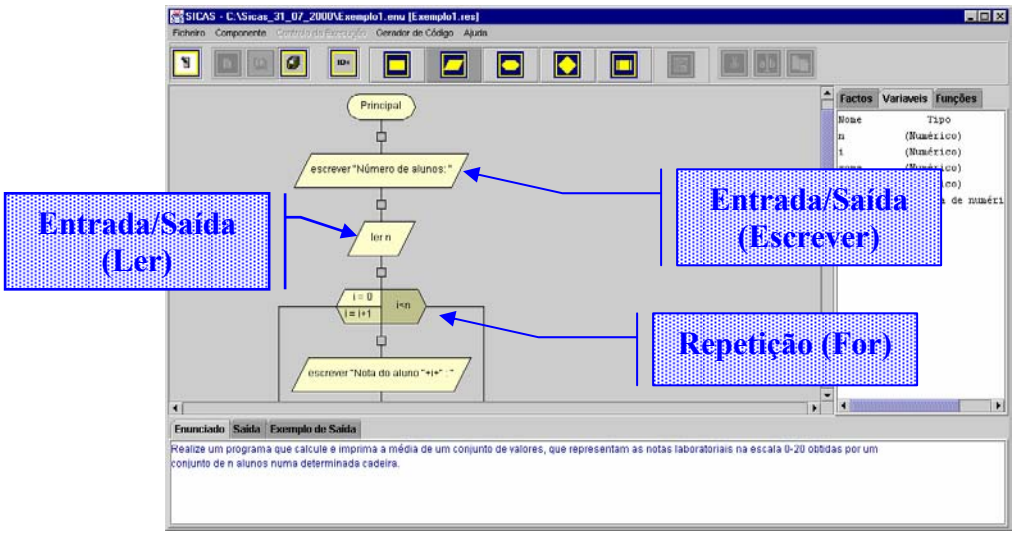
2. SICAS 1.0

O presente artigo apresenta um conjunto de testes feito ao SICAS 1.0, tendo levado a algumas propostas de alteração a contemplar numa nova versão, o SICAS 2.0. Para um melhor enquadramento é feito um resumo das principais características do SICAS 1.0.

O SICAS é um ambiente que possibilita, essencialmente, dois tipos de cenários: edição/resolução de problemas e execução/simulação de resoluções previamente construídas. No primeiro cenário, o utilizador pode construir algoritmos através de representações visuais – fluxogramas – recorrendo a simbologia gráfica que representa as principais estruturas necessárias à construção de um algoritmo. No segundo cenário, o utilizador pode simular a execução das resoluções construídas, analisando-as com o detalhe e ritmo desejado.

O SICAS apresenta dois modos de funcionamento referentes aos dois principais tipos de utilizadores (alunos e professores). As possibilidades de ambos os modos são semelhantes com a grande diferença de apenas o professor poder definir problemas, bem como fornecer dados de teste (soluções) para que os alunos possam testar as suas resoluções. Um problema é composto por um enunciado, uma ou mais resoluções (algoritmos) que lhe ficam associadas e, eventualmente, uma solução. A solução destina-se à indicação, pelo professor, de um conjunto de caracteres, representativos da saída esperada para a resolução, assumindo valores de entrada indicados também pelo professor. Estes pares, dados de entrada e resultados esperados, permitirão aos alunos verificar o funcionamento das suas resoluções.

Para a construção da resolução do problema existe um ambiente icónico. Neste, o utilizador pode construir a resolução do seu problema de programação sob a forma de fluxogramas, recorrendo a ícones ou a elementos de menus que representam as principais estruturas necessárias à sua resolução. No SICAS foram consideradas as seguintes estruturas para a construção de fluxogramas: Atribuição (tem como finalidade atribuir o valor de uma determinada expressão a uma variável), Entrada/Saída (possibilita ler (Entrada) valores para variáveis e escrever (Saída) expressões), Repetição (permite realizar a execução de uma acção um determinado número de vezes) e Selecção (tem como propósito escolher em alternativa o conjunto de acções a ser executado). A figura seguinte ilustra a construção de uma resolução.



A forma de introdução destes elementos é muito simples, bastando para tal seleccionar o tipo de elemento a introduzir, preencher, num diálogo que surje automaticamente, os campos necessários para a completa definição de cada estrutura e seleccionar o ponto de inserção pretendido. Também as linhas que interligam os vários componentes são automaticamente geradas, possuindo no seu centro pontos de inserção bem definidos, a fim de evitar diagramas inconsistentes.

Relativamente aos tipos de dados, o SICAS possibilita apenas a utilização de quatro tipos: "Numéricos", "Cadeia de Caracteres", "Tabela de numéricos" ou "Tabela de cadeia de caracteres". A ideia é a de sensibilizar os alunos para os tipos essenciais de dados pois pensamos que a diversidade de tipos de dados, apesar de ser um aspecto importante das linguagens de programação, constitui muitas vezes, numa fase inicial de aprendizagem da programação, um conjunto de dificuldades acrescidas desviando a concentração dos alunos do importante, a concepção do algoritmo.

O SICAS possibilita a definição de funções, o que permite introduzir o conceito de modularização desde o início da aprendizagem de programação. A construção de funções é realizada utilizando as estruturas de controlo anteriormente referidas. A existência de um conjunto de funções pré-definidas que podem ser utilizadas na construção de expressões é outra possibilidade incluída no sistema.

O SICAS permite também a execução de resoluções previamente construídas, de modo a que se possa verificar de forma gráfica e animada a lógica do algoritmo, os seus passos individuais e o

fluxo de execução. Em qualquer momento da simulação o utilizador poderá pará-la pelo período de tempo que desejar, a fim de melhor analisar o funcionamento do algoritmo.

O aluno pode também verificar a correcção da resolução construída. Pensamos que esta opção é desejável na medida em que possibilita ao aluno, depois de estudar a matéria, utilizar o programa para, liberto das preocupações classificativas, auto-avaliar a sua aprendizagem antes de se submeter a testes formais. As funcionalidades apresentadas são descritas com um grau de detalhe mais aprofundado em Gomes e Mendes (2000). De realçar, no entanto, que as possibilidades referidas são não apenas extremamente importantes para os alunos mas poderão constituir uma mais valia para apoio das actividades lectivas do professor, num contexto de sala de aula auxiliando-o nas suas tarefas de explicação e exposição da matéria. Neste contexto, permite uma variedade de actividades de que se destacam a possibilidade de fornecer exercícios para os alunos completarem, alterarem, corrigirem ou conceberem de raiz, mostrar diversas versões de determinado problema de forma rápida ou modificar programas por forma a responder a dúvidas dos alunos. O SICAS pode ainda ser utilizado numa perspectiva de avaliação na medida em que o professor pode indicar enunciados que pretende que os alunos resolvam e posteriormente consultar/avaliar as resoluções dos seus alunos.

3. Experiências

O SICAS já foi submetido a alguns testes. Foram realizadas avaliações informais que consistiram na distribuição da aplicação junto de professores que leccionam este tipo de disciplinas e avaliações com alunos.

Relativamente ao primeiro grupo de avaliadores, as opiniões gerais foram no sentido de que o SICAS é muito interessante e que pode ser muito útil para as cadeiras introdutórias de programação. A generalidade dos avaliadores foi da opinião que o SICAS constituía não apenas uma mais valia para a aprendizagem de programação pelos alunos, mas que seria muito útil para utilização em ambiente de sala de aula, como apoio às actividades do professor. Em especial gostaram bastante de ver incorporado a chamada de funções no fluxograma. Grande parte dos avaliadores achou também importante sensibilizar os alunos para os tipos essenciais de dados o que também não constitui uma prática muito abordada tradicionalmente no capítulo de algoritmia e que consideram importante. Acima de tudo, os professores gostaram bastante da parte de simulação do algoritmo, julgando ser o ponto mais forte do trabalho. Também apreciaram a possibilidade de validação dos resultados esperados, essencialmente por contribuir para a

detecção de erros lógicos, que às vezes a simulação normal deixa escapar. As maiores discrepâncias de opiniões residiram na interface gráfica, apesar da maioria dos avaliadores a achar agradável e bastante intuitiva.

Nas experiências com alunos o principal objectivo consistiu, numa primeira fase, em medir a intuitividade da interface e facilidade de navegação. A avaliação processou-se de duas formas. Uma consistiu na distribuição do SICAS a todos os alunos que frequentaram as disciplinas de "Programação e Algoritmos I" e "Algoritmos e Programação I" no DEI-FCTUC e no DEIS-ISEC, respectivamente. Assim, foram apresentadas aos alunos as principais potencialidades do ambiente, sendo o sistema disponibilizado num servidor acompanhado do manual de utilização que incluía também exemplos de construção de algoritmos e das suas simulações.

A outra forma de avaliação foi realizada nas aulas práticas de "Programação e Algoritmos I". Esta experiência utilizou a observação directa, a análise de tarefas e a recolha de comentários e opiniões e processou-se da forma a seguir descrita. Nas turmas da disciplina cada Docente apresentou o SICAS em traços gerais, salientando as principais funcionalidades, características, actividades que proporciona e significado dos vários ícones. Posteriormente, os alunos foram divididos em grupos de dois elementos cada. Os grupos eram geralmente heterogéneos, no que respeita à experiência anterior de programação. A cada grupo foi proposto que realizasse uma actividade no SICAS. Esta consistia na tradução do algoritmo apresentado no quadro, pelo professor, sob a forma de pseudo-código. A ideia consistia em avaliar essencialmente a facilidade de navegação do SICAS. Cada Docente acompanhou o mais possível os grupos, de forma a registar as suas actividades, reacções e dificuldades, nomeadamente o tempo médio de realização da actividade, o número de vezes que o grupo se enganou no tipo de elemento que pretendia colocar na resolução, coisas que devia ter feito e não fez porque se esqueceu ou porque não sabia que era necessário fazer, o número e tipo de ajudas que foram pedidas, o tipo de erros efectuados na introdução de objectos, entre outros.

Na aula seguinte, a cada aluno individualmente, foram apresentadas folhas com impressões dos ecrãs de Edição/Resolução de um problema e de Execução/Simulação de uma resolução. A ideia consistiu em apurar o que cada ícone sugeria, medindo a intuitividade da interface. Embora a maioria dos alunos conseguisse identificar os principais ícones referentes ao ambiente de resolução do problema, houve contudo alunos que não conseguiram identificar qualquer símbolo e houve também uma grande variedade de associações erradas. No que respeita ao cenário de

simulação os resultados foram na generalidade muito maus, o que se atribui ao facto de este aspecto ter sido apenas apresentado e não explorado na aula de demonstração, dando-se particular destaque apenas ao ambiente de resolução de problemas.

Posteriormente o SICAS passou a ser utilizado em contexto de sala de aula substituindo os tradicionais meios, caneta e papel, para a construção de algoritmos. Durante a primeira utilização, os alunos recorriam essencialmente aos menus para selecção das operações a realizar, em detrimento dos ícones, não os achando muito sugestivos. Pensamos que isto se deveu ao facto de, a grande maioria dos alunos terem pouca ou nenhuma experiência de utilização de fluxogramas, desconhecendo em geral a simbologia utilizada na sua construção. Esta ideia acentuou-se, pois após um período de várias aulas de utilização do SICAS, os alunos deixaram de recorrer aos menus para utilizarem os ícones.

A generalidade dos alunos achou que, após compreensão do seu funcionamento, o SICAS constitui uma mais valia, ajudando-os a entender muitos aspectos que só poderiam alcançar através da ajuda personalizada de um explicador.

4. SICAS 2.0

Da análise das reacções dos alunos na execução das suas tarefas e de comentários chegados por parte de uma miscelânea de avaliadores do SICAS, os autores decidiram fazer uma nova versão. Nesta os melhoramentos foram considerados essencialmente a dois níveis, aperfeiçoamento da interface e inclusão de novas funcionalidades consideradas relevantes.

Relativamente ao primeiro grupo de alterações pode-se destacar a reorganização dos ícones na barra de tarefas, bem como o redimensionamento e reposicionamento de alguns deles, com o objectivo de maximizar a área de resolução/simulação do problema. Outras alterações consistiram na melhoria do sistema de ajuda ao utilizador e na reformulação da maioria das mensagens de erro, tendo também sido introduzidas novas mensagens, de forma a melhorar a comunicação entre o SICAS e o utilizador. Para minimizar os erros do aluno e orientá-lo na realização das suas actividades, está a ser desenvolvido um "Sistema de aconselhamento" no âmbito de um trabalho de mestrado.

No sentido de facilitar a navegabilidade e velocidade de interacção foram adicionadas teclas de atalho para algumas operações comuns. Também a possibilidade de desfazer vários níveis de operações, vulgo *undo*, foi considerada, pois pensámos que muitas vezes seria útil para o utilizador poder voltar atrás nas suas acções, sem que para isso tivesse que despender muito

esforço e tempo. Dentro deste nível de alterações incluiu-se ainda a possibilidade de alteração de componentes através do duplo toque no botão do rato, surgindo a janela de alteração adequada a cada caso, bem como a exibição do comentário associado a cada bloco, sempre que o rato estiver sobre ele (*tooltips*). Houve também alterações de certos nomes de componentes ou secções tornando-os mais sugestivos. Foram ainda efectuadas algumas correcções, refinamentos e optimizações no que respeita a um conjunto de bugs e/ou incoerências existentes em variadas situações.

Do conjunto de alterações que o SICAS 2.0 contemplará, a que se tornou mais pertinente, sendo mencionada pela grande maioria dos avaliadores, foi a possibilidade de reutilizar uma função criada durante a resolução de um problema na resolução de outros problemas, de uma forma sugestiva e confortável. Para além de ser permitida a coexistência de várias resoluções para o mesmo problema, o que já acontecia na versão anterior, é também possível na nova versão aproveitar a resolução de um determinado problema para outro.

No que respeita aos testes e simulações das resoluções construídas, também foram implementadas várias sugestões. Uma delas consistiu na existência de múltiplos conjuntos de teste, fornecidos pelo professor, para a mesma resolução e não apenas um. No que se refere à simulação propriamente dita, foi reduzido o número de velocidades de execução existentes, de 3 (passo-a-passo, lento e rápido), para 2. Pelas experiências realizadas constatou-se que dois tipos de velocidade de execução do algoritmo seriam suficientes, o passo-a-passo e a apresentação do resultado final da execução, sem ter que esperar e ver a execução de todos os elementos. Além disso, na simulação passo-a-passo acrescentou-se a possibilidade de executar apenas partes do algoritmo, definidas pelo utilizador, através da introdução do conceito de *breakpoints*, bem como a possibilidade de alterar os valores das variáveis durante a execução, a fim de possibilitar a realização de determinados testes. Está também actualmente em estudo a implementação da possibilidade de apresentar o resultado das funções chamadas sem as ter que executar passo-a-passo (*skip*), entre outras funcionalidades típicas de operações de *debug*.

Uma outra funcionalidade importante é a possibilidade de converter automaticamente uma resolução expressa pelo fluxograma no código correspondente numa linguagem de programação, C ou JAVA. Esta opção estava implementada mas não completamente funcional no SICAS 1.0. Para além de simplificar o trabalho dos alunos, esta funcionalidade pode servir para realçar a ideia que, a um nível básico, a linguagem de programação utilizada não é o mais importante, pois

uma vez construido o algoritmo, a sua transcrição para uma qualquer linguagem deste tipo é um processo simples e mecânico. Por outro lado, esta funcionalidade pode servir como estímulo à utilização do ambiente, evitando que os alunos o vejam como uma perda de tempo quando o seu objectivo for a criação de um programa.

Ainda prevista, está a possibilidade de gerar relatórios sobre as actividades dos alunos. Neles constaria informação pertinente sobre o seu desempenho e evolução, para que o professor melhor pudesse orientar as actividades do aluno de forma pedagogicamente correcta. Referente a este aspecto está ainda prevista a possibilidade do professor aceder remotamente e de forma centralizada às resoluções/actividades de todos os alunos.

Por último, outra das alterações em implementação é a introdução de suporte para utilização em várias línguas. O SICAS 1.0 está implementado em Português, no entanto, tem havido numerosos pedidos para experimentar o SICAS por parte de pessoas de variadas nacionalidades. Assim, entendeu-se dotar o SICAS 2.0 com a capacidade de ser configurado para utilização com outras línguas.

Espera-se que no momento de apresentação deste artigo se esteja a dar início à realização de novas experiências com o SICAS 2.0 com todas as alterações mencionadas completamente funcionais.

5. Conclusões

O SICAS, descrito nesta apresentação, é um ambiente educativo que, de uma forma apelativa, dinâmica e atraente, promete ajudar os alunos a aprender aspectos básicos de programação estruturada. Esta aprendizagem centra-se na construção algorítmica, sob a forma de fluxogramas, de resoluções para problemas de programação.

Para que os alunos possam edificar o seu conhecimento de uma forma construtiva e esclarecer as suas dúvidas e modelos mentais é possível a simulação das resoluções construídas de forma dinâmica e interactiva.

A primeira versão do SICAS já foi objecto de avaliação por parte de um conjunto de professores com experiência no ensino da programação, bem como de alunos. Em resultado da análise das sugestões e críticas obtidas, bem como das experiências de avaliação está em desenvolvimento uma nova versão que será alvo de testes mais alargados no início do próximo ano lectivo, essencialmente no âmbito de disciplinas de introdução à programação.

6. Referências

Bereiter, C. and Ng., E. (1991). Three Levels of Goal Orientation in Learning. *Journal of the Learning Sciences*. Vol. 1, pp 243-271.

Gomes, Anabela (2000). *Ambiente de Suporte à Aprendizagem de conceitos básicos de programação*. Tese de Mestrado. Universidade de Coimbra, Portugal.

Gomes, A. e Mendes, A. J. (1998). Ambiente de suporte à aprendizagem de conceitos básicos de programação. *Actas do 3º Simpósio de Investigação e Desenvolvimento de Software Educativo*. Évora, Portugal.

Gomes, A. e Mendes, A. J. (1999). A animação na aprendizagem de conceitos básicos de programação. *Revista de Enseñanza y Tecnología*, No. 13, pp. 22-32.

Gomes, A. e Mendes, A. J. (2000). Suporte à aprendizagem da programação com o ambiente SICAS. *Actas do V Congresso Ibero-Americano de Informática Educativa*, Viña del Mar, Chile.

Jenkins, Tony. (2001). *The Motivation of Students of Programming*. MSc Thesis, University of Kent, Inglaterra.

Jenkins, Tony. (2002). On the Difficulty of Learning to Program. *Proceedings of 3rd Annual LTSN-ICS Conference*, pp 53-58, Loughborough University, Inglaterra.

Jenkins, Tony. (2001). *The Motivation of Students of Programming*. MSc Thesis, University of Kent, Inglaterra.

Kann, C., Lindeman, R. y Heller, R. (1997). Integrating Algorithm Animation into a Learning Environment. *Computers &Education*. Vol. 28, No. 4, pp. 223-228.

Ramani, K. V. y Rao, T. P. (1994). A graphics based computer-aided learning package for integer programming: the branch and bound algorithm. *Computers & Education*. No. 23, pp. 1-10.

Shih, Y. y Alessi, S. (1994). Mental Models and transfer of learning in computer programming. *Journal of Research on Computing in Education*. No. 26, pp. 155-175.

Stasko, J., Badre, A. y Lewis, C. (1993). *Do algorithm animations assist learning? An empirical study and analysis Proceedings of the INTERCHI'93 Conference on Human Factors in Computing Systems*, pp. 61-66. Amsterdam, Holanda.