Caderno de Algoritmos: um Ambiente de Resolução de Problemas de Programação

Carine Webber – CCTI/UCS – cgwebber@ucs.br

Matheus Zenato – CCTI/UCS – mzenatto@ucs.br

Tiago Arrosi – CCTI/UCS – tarrosi@ucs.br

Resumo. *Este artigo descreve um ambiente para a aprendizagem de programação através da resolução de problemas. O ambiente proposto é o resultado de um amplo estudo sobre plataformas de programação e softwares educativos. O resultado obtido consiste de uma aplicação web onde estudantes resolvem problemas de programação, depuram, testam, e se valem de diagnósticos automáticos. Os professores podem acompanhar o processo de aprendizagem de seus estudantes e interagir individualmente auxiliando-os e revisando suas construções. Experimentos iniciais já foram realizados e as principais considerações são apontadas neste artigo.*

Palavras-chave: ambientes de aprendizagem, aprendizagem através de resolução de problemas, diagnóstico do aluno

A Problem Solving Learning Environment for Computer Programming Beginners

Abstract. *Programming learning is the main focus of this work. This paper describes a problem solving learning environment for programming courses. The proposed environment is the result of an wide study on programming platforms and educational software. The result consists of a web application where students solve programming problems, debug, test and use automatic diagnosis. Professors can follow students´ learning processes and interact individually helping and reviewing their constructions. This paper concludes by presenting initial experiments and main contributions.*

Key-words: learning environments, problem solving learning, student diagnosis.

1. Introdução

Aprender a programar é uma tarefa complexa para a maioria dos alunos e a sua principal causa é a falta de habilidade para resolver problemas. A maioria dos alunos apresenta alguma dificuldade em problemas envolvendo cálculos matemáticos e conhecimentos de lógica (Gomes et al., 2006). A aprendizagem através da resolução de problemas é uma prática comum em áreas como a matemática, a física e a programação de computadores. Nestas áreas, estudos mostram que uma retroação imediata auxilia o aprendizado (Schulze, 1989), informando ao aluno seus os erros e indicando as melhorias que ele poderia fazer na resolução do problema.

A maioria dos ambientes de aprendizagem para o ensino da programação foi desenvolvida para ensinar os alunos a escrever programas. Kumar (2005), entretanto, se

preocupou em desenvolver um ambiente que pretende ajudar os alunos a compreender o programa que foi escrito e fazer com que os erros sejam encontrados e compreendidos de forma rápida e clara, sem a ajuda do professor. Segundo este autor, a maneira mais produtiva de fazer com que o aluno entenda um problema é quando o professor orienta o aluno através de uma estrutura que seja base da solução do problema. A execução de um algoritmo passo a passo é a forma de um ambiente virtual de aprendizagem ajudar o aluno a compreender uma solução desenvolvida para o problema.

Ambientes de aprendizagem para resolução de problemas são sistemas que fornecem aos alunos diversos problemas para serem resolvidos. Esses sistemas devem ser capazes de acompanhar e ajudar os alunos durante a resolução dos problemas. Eles devem fornecer ferramentas, dar dicas para o estudante solucionar os problemas e indicar se a solução do estudante está correta ou não. Ambientes baseados na resolução de problemas podem ser divididos em três grupos, segundo a forma com que lidam com os problemas:

1. No primeiro grupo temos os sistemas que resolvem automaticamente os exercícios, tais como o sistema JFLAP (Cavalcante et al., 2004) utilizado no ensino de conceitos sobre autômatos e máquinas de *Turing*.

2. O segundo grupo é composto pelos sistemas que administram os problemas fornecidos por um instrutor ou professor. A restrição desses sistemas é que eles possuem um repertório limitado de problemas a serem resolvidos.

3. O último grupo é formado pelos ambientes de aprendizagem que são capazes de criar novos problemas para serem resolvidos, usando *templates* pré-definidos para gerar problemas com respostas diferentes. Neste grupo encontram-se os sistemas PILOT (Bridgeman et al., 2000) para algoritmos gráficos e Gateway Labs (Baldwin, 1996) para fundamentos matemáticos na Ciência da Computação.

Neste trabalho propõe-se um ambiente de aprendizagem de programação que faz uso das técnicas descritas no primeiro e segundo pontos, sendo assim capaz de avaliar soluções de alunos para problemas previamente cadastrados por professores. O ambiente se baseia em um cenário de resolução de problemas, onde o aluno pode solicitar auxílio do sistema, dos seus pares e ainda do professor. Em nossa concepção, o professor assume um papel muito importante na aprendizagem, acompanhando os estudantes nos processos de resolução de problemas. O presente artigo está organizando em seis seções. A seção 2 apresenta uma breve revisão de quatro ferramentas relacionadas que se destinam a aprendizagem de programação. A seção 3 apresenta os principais recursos do ambiente de aprendizagem desenvolvido. A seção 4 detalha o componente de diagnóstico da solução do aluno. A seção 5 descreve um experimento realizado e os resultados obtidos na análise. A seção 6 finaliza o artigo apresentado algumas lições aprendidas e contribuições deste trabalho.

2. Softwares para Aprendizagem de Programação

Diversos softwares foram desenvolvidos para auxiliar os alunos na aprendizagem da programação de computadores. Cada um foi desenvolvido com um objetivo especifico, por isso possuem características que devem ser analisadas. Alguns buscam manter o aluno sempre motivado durante o aprendizado, outros buscam manter a interface simples para que o usuário não perca tempo tentando entender como ela funciona e onde estão as ferramentas que deseja utilizar. Existem também aqueles que mantêm um

ambiente focado no trabalho colaborativo para que o aluno sempre aprenda com a ajuda de pares online e dispostos a ajudar.

Analisamos alguns sistemas e realizamos uma comparação entre as funcionalidades que cada um deles possui. Os sistemas avaliados foram Visualg (Souza, 2009), Eclipse (Eclipse Foundation, 2009), Progranimate (Scott et al., 2008) e WEBportugol (WEBportugol, 2009). A tabela 1 apresenta uma comparação entre estes sistemas. Analisou-se cada ferramenta com relação aos seguintes critérios: execução direta; execução passo a passo; inspeção de variáveis; mensagens de erros explicativas; indicação de erros durante a programação; resolução automática de erros (o software fornece alguma forma automatizada para corrigir os erros encontrados); apresenta fluxograma do código para visualização; interface configurável; editor possui realce da sintaxe; permite alteração nas variáveis durante a execução.

Tabela 1 - Quadro Comparativo entre softwares analisados

	Visualg	**Eclipse**	**Progranimate**	**WEBportugol**
Execução direta	*	*	*	*
Execução passo a passo	*	*	*	*
Inspeção de variáveis	*	*	*	*
Mensagens de erro explicativas	*			
Indicação de erros durante programação		*	*	
Correção automática de erros		*		
Fluxograma do código			*	
Ambiente Configurável		*		
Realce da sintaxe	*	*	*	*
Alteração nas variáveis durante execução		*		

Após a análise de alguns ambientes utilizados na programação foi possível constatar que cada um deles possui características e ferramentas que os diferenciam. A ferramenta mais completa sem dúvida é o Eclipse, mas essa é uma ferramenta utilizada para o desenvolvimento de software profissional. Ela possui inúmeras funcionalidades que alunos de programação não necessitam, ela é bastante complexa e por isso essa ferramenta não é indicada no ensino da programação para alunos iniciantes. O ambiente Progranimate possui uma funcionalidade que ajuda muito o aluno no entendimento do algoritmo escrito. O sistema apresenta ao lado do algoritmo um fluxograma que representa visualmente o algoritmo desenvolvido. Este fluxograma ajuda o estudante a visualizar o seu algoritmo e entender como ele é executado. Mesmo durante a depuração do algoritmo o sistema indica a linha que está sendo executada no algoritmo e a ação respectiva no fluxograma. O Visualg é uma ferramenta simples, voltada para a aprendizagem de programação, por isso possui apenas as funcionalidades que o estudante precisa quando esta começando a programar. Nele o estudante consegue: digitar um algoritmo, compilar o algoritmo e verificar se possui erros de sintaxe e executar o algoritmo passo a passo. O ambiente de aprendizagem WEBportugol é a ferramenta mais simples de todas analisadas, que possui menos funcionalidades e uma

interface limitada. A única vantagem em relação às demais é que ela foi desenvolvida utilizando *applets*, podendo então ser executada dentro de um navegador e integrada a outros ambientes de aprendizagem que rodam na web. Todas estas funcionalidades destacadas serviram de ponto de partida para o desenvolvimento do software que se denominou Caderno de Algoritmos. Sendo um caderno, a proposta do ambiente é que ele sirva para armazenar soluções de algoritmos, mas também anotações pessoais e material didático. O único item da tabela 1 que não foi implementado na versão atual do Caderno de Algoritmos foi a visualização de problemas através de fluxogramas. As seções seguintes detalham características do software.

3. Apresentação do Caderno de Algoritmos

O Caderno de Algoritmos possui as funcionalidades e componentes descritos a seguir.

Cadastro de Turma e Professor. O primeiro passo para uso do ambiente consiste em criar uma turma e atribuir a ela um professor. O administrador informa o nome da turma e preenche o cadastro referente ao professor que será vinculado a ela.

Conteúdo Didático. Um usuário professor pode adicionar conteúdo disciplinar, enviar mensagens para seus alunos, cadastrar os exercícios e acompanhar o aprendizado do aluno. Ao adicionar um conteúdo disciplinar, é solicitado o tipo de inclusão que o professor deseja realizar. Adicionar Pasta ou Arquivo são as duas opções disponíveis. A pasta constitui uma forma de o professor organizar o conteúdo didático da melhor forma. O arquivo é o conteúdo que o professor deseja adicionar ao sistema.

Exercícios. Cabe ao professor o cadastro de exercícios para que os alunos possam resolver e aperfeiçoar o seu aprendizado. A tela de inclusão de um novo exercício é apresentada na Figura 1. A inclusão de um exercício se dá mediante a inserção de um título e uma descrição ou enunciado. O professor pode ainda adicionar um anexo (um arquivo ou imagem, por exemplo). O campo “dificuldade” serve para informar o grau de dificuldade do exercício. O campo “área do conhecimento” identifica em qual das três áreas do estudo de algoritmos esse exercício se enquadra. Os campos “limiar área 1”, “limiar área 2”, “limiar área 3” servem para configurar o sistema de diagnóstico do aluno. O limiar corresponde ao limite de erros que o estudante pode gerar naquela área específica quando estiver desenvolvendo sua solução. Acima destes limites, ele recebe mensagens especiais do componente de diagnóstico do aluno.

Cadastro de Aluno. Nesse momento, possuindo uma turma e exercícios o ambiente está apto a ser utilizado pelos alunos. O cadastro de aluno é livre. Qualquer pessoa que deseja ingressar ao ambiente como aprendiz pode se cadastrar através do botão presente na interface de autenticação. Quando solicitado o cadastro de um novo usuário o ambiente apresenta a seguinte tela de cadastro. Para o cadastro o aluno deve informar um email e uma senha, os quais serão utilizados como meio de autenticação, o seu nome, a turma a qual deseja ingressar e informar se ele aceita contribuir ajudando os demais alunos participantes, ou seja, se o aluno não desejar ajudar os demais o ambiente somente irá permitir que ele troque mensagens com o professor, caso contrário ele poderá se comunicar com o professor e os demais alunos.

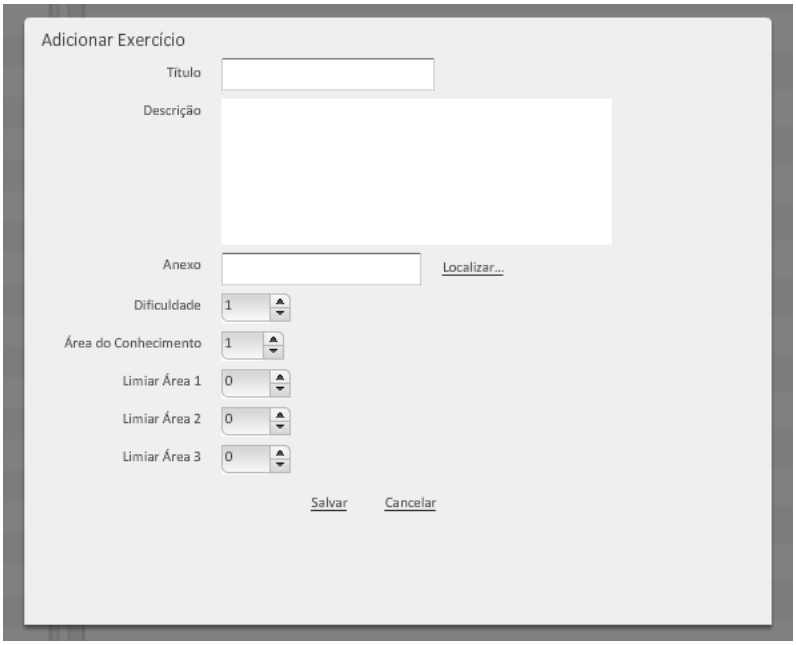

Figura 1 – Professor cria novo exercício

Mensagens. O estudante pode se comunicar com o professor ou com seus pares através de mensagens utilizando o sistema de recados. Ele informa um título para a mensagem, adiciona conteúdo e destinatários e envia. Todas as conversas nas quais o estudante estiver participando são exibidas em uma tela de Mural de Recados.

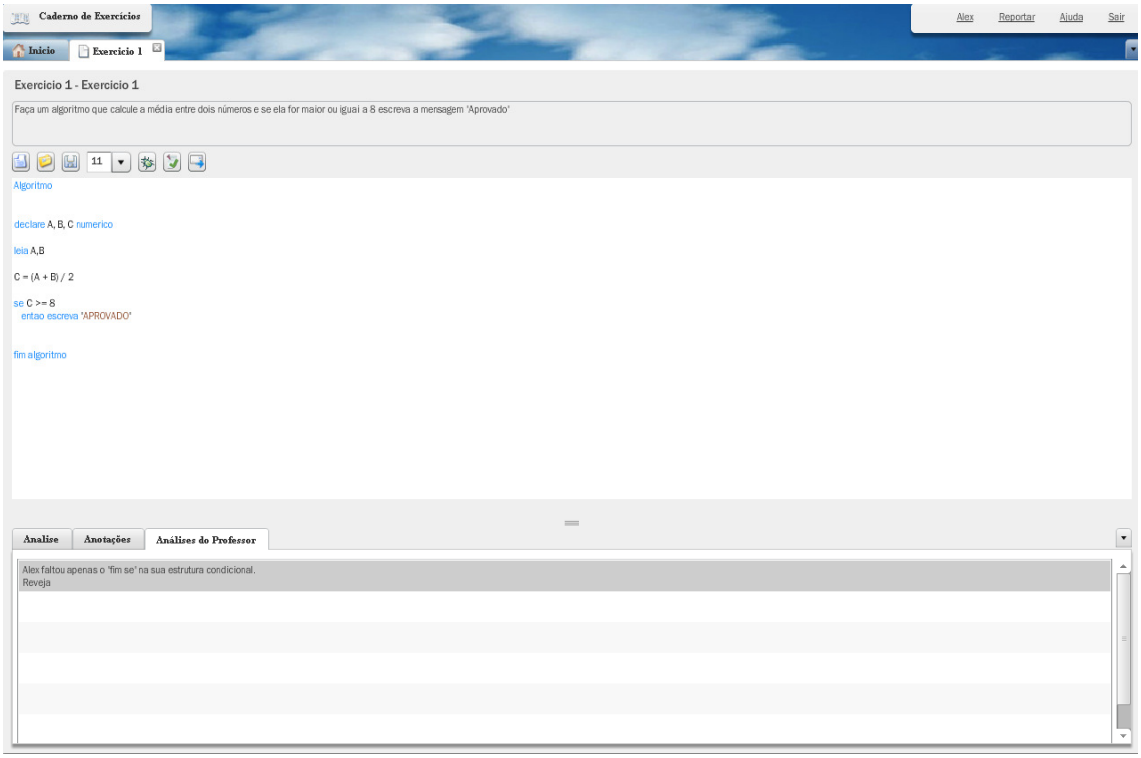

Figura 2 – Aluno resolve o exercício

Resolução de Exercícios. Uma interface específica (Figura 2) apresenta o enunciado do exercício, campo de edição e botões com ações: criar, abrir, salvar, enviar para análise (diagnóstico) e enviar para o professor. O aluno desenvolve a sua solução no editor e assim pode solicitar uma análise ao sistema SIATP (Webber, Viganó e Possamai, 2009), que lhe retorna um diagnóstico dos erros encontrados. O resultado dessa análise é apresentado na aba A*nálise*. A aba *Anotações* pode ser utilizada pelo aluno para colocar anotações úteis sobre o exercício. Ao enviar o algoritmo revisado para o professor, o editor bloqueia as alterações no algoritmo até que ele seja revisado e comentado pelo professor. A aba *Análises do Professor* contém as observações do professor sobre o exercício.

Acompanhamento de Aprendizado. Um usuário professor quando autenticado no sistema possui acesso ao material produzido pelos seus estudantes. O professor pode visualizar os exercícios propostos e verificar as solicitações de análise pendentes. O professor pode visualizar a solução do aluno, as análises que o aluno realizou utilizando o SIATP e as anotações do aluno (Figura 3).

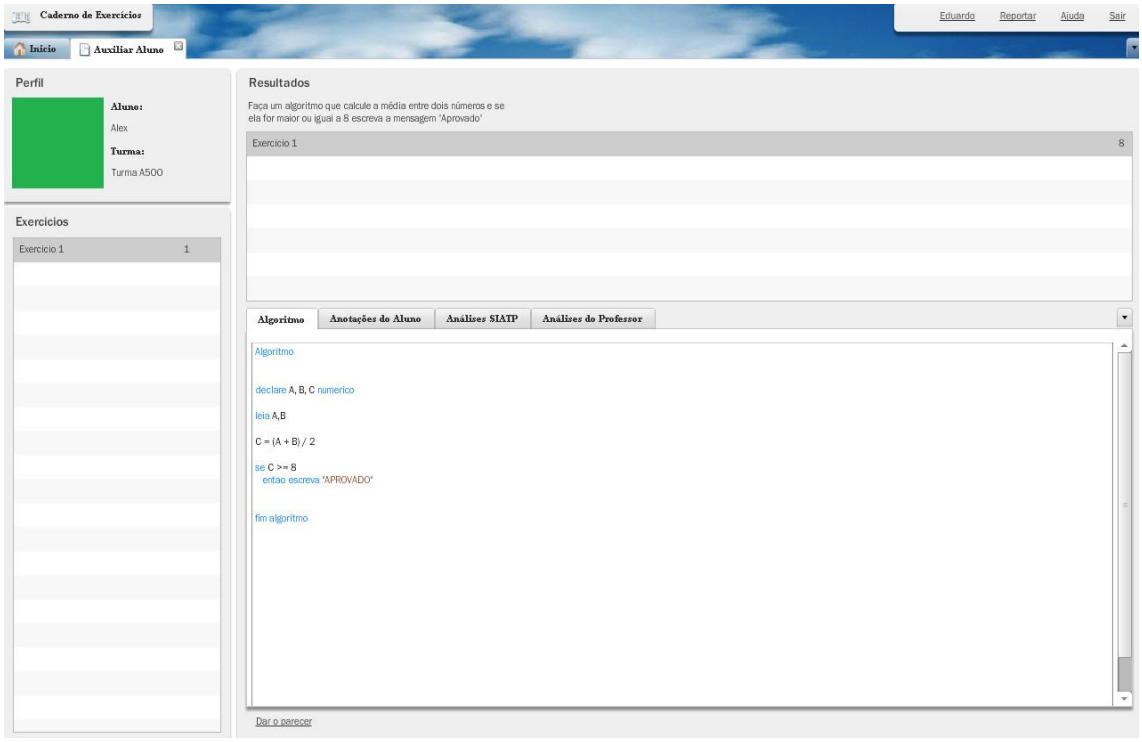

Figura 3 - Professor acompanha aprendizado

Com base nessas informações o professor pode escrever um parecer (Figura 4). Após o envio do parecer, o estudante recebe a informação e o ambiente libera a edição e correção do algoritmo.

Depurador de Algoritmos. Ao solicitar a depuração (disponível na Figura 2), o estudante é levado para outra tela, onde existem diversas ferramentas para realizar a depuração do algoritmo. A tela principal do depurador é dividida em três grandes partes:

uma janela com o algoritmo e a barra de ferramentas do depurador, a janela de inspeção de variáveis e a janela de mensagens de entrada e saída. O processo de depuração segue o formato tradicional encontrado em diversos ambientes de programação. O estudante pode controlar a execução, inspecionar variáveis, monitorar de mensagens de saída e fornecer dados de entrada de valores. Através da execução passo a passo o estudante pode verificar se o fluxo de execução está de acordo com o esperado. Existe também a possibilidade da inclusão de pontos de parada no código.

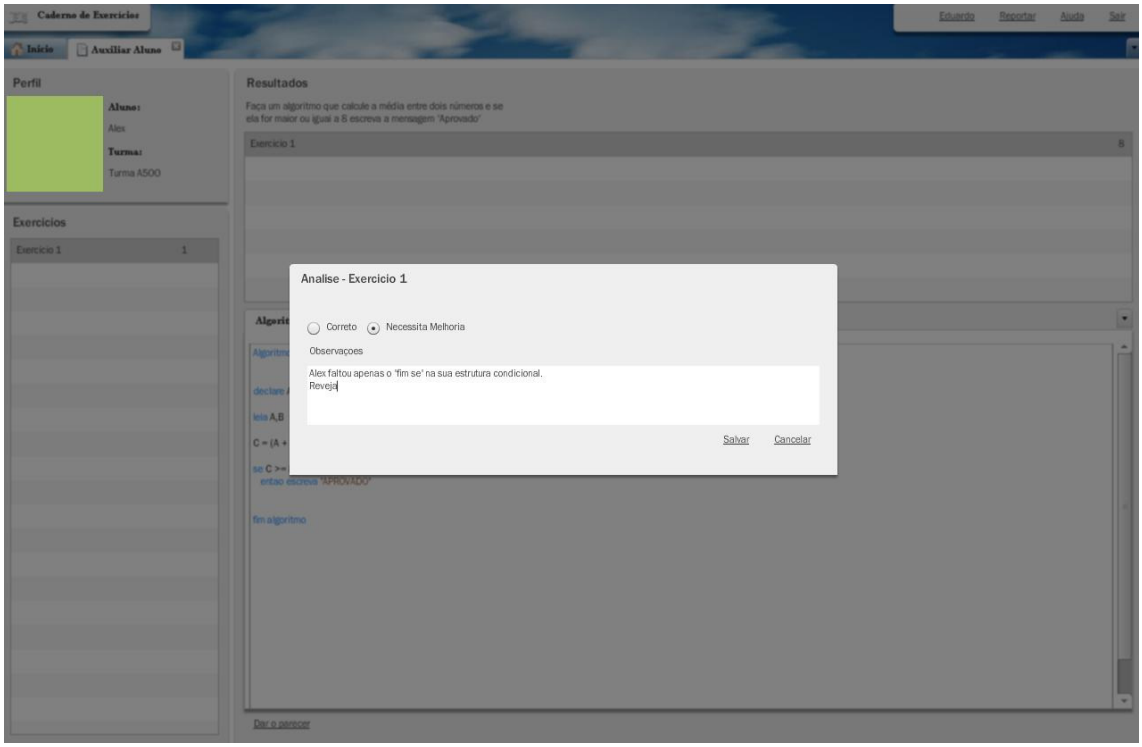

Figura 4 - Professor envia parecer para o estudante

4. Diagnóstico da solução do aluno

Integrou-se ao Caderno de Algoritmos um sistema multiagentes de diagnóstico das soluções do aluno. O diagnóstico do aluno foi desenvolvido a partir do conceito de Sistemas Imunológicos Artificiais. A metodologia imune se mostrou adaptada ao processo de diagnóstico do aluno pois identificou-se diversas similaridades entre o conceito de sinal de perigo do sistema imunológico natural e o sinal de erro no processo de aprendizagem de programação.

O sistema multiagentes desenvolvido é composto por três classes de agentes. Primeiramente, o agente macrófago analisa e decompõe o algoritmo do aluno. Agentes semanticamente distribuídos fazem o papel de linfócitos e são responsáveis por procurar por padrões de perigo (erro) no algoritmo decomposto, emitindo sinal de perigo quando necessário. Agentes de diagnóstico coletam os sinais de perigo e agrupam aqueles que forem relacionados. Estas etapas estão detalhadamente descritas em artigos precedentes (Webber, Viganó e Possamai, 2009).

Para a realização do processo de diagnóstico do aluno assume-se que a aprendizagem de programação é organizada em três áreas de conhecimento (AC)

progressivas (área 1, área 2 e área 3) descritas na tabela 2.

Para a tarefa de diagnóstico do aluno, estabeleceu-se que as três ACs correspondem à três zonas de perigo. Cada AC delimita uma zona de perigo onde os observáveis para o diagnóstico possuem pesos distintos à medida que o aluno avança em sua aprendizagem. Para que o acompanhamento da evolução da aprendizagem possa ocorrer, associou-se a cada AC uma variável de limiar (t) que corresponde a um número a partir do qual o sinal de perigo será disparado. Ele corresponde ao número máximo de erros que podem ser encontrados na resolução de um problema para que o sistema emita um alerta de perigo, o que sinaliza problemas de aprendizagem. Este limiar deve ser configurado pelo professor na inclusão do exercício (ilustrado na Figura 1), podendo ser alterado posteriormente.

Tabela 2 - Descrição das áreas de conhecimento

AC 1	AC 2	AC 3
Variáveis: declaração e tipos Expressões Operador de atribuição Algoritmos sequenciais Comandos Condicionais (se-então, se-então-senão, caso) Instruções de entrada e saída	Comandos de repetição (para, repita, enquanto) Condições de parada: - No início do bloco (enquanto) - No final do bloco (repita) Fluxo de controle Comandos aninhados	Estruturas de dados: - Vetores - Matrizes Manipulação de Índices

A tabela 3 permite visualizar, para cada combinação de AC, como erros são convertidos em sinal de perigo; *ns* indica que não há sinal de perigo e *na* indica que nenhum sinal se aplica para esta área. Para compreender a tabela 3, suponha um aluno no início da disciplina de algoritmos (primeira área de conhecimento – AC 1). Durante a resolução de problemas é normal que o aluno erre e corrija seus erros várias vezes. Cada erro na AC 1 gera um sinal de perigo fraco. Se a quantidade de erros exceder ou igualar o limiar estabelecido t_{AC1}, então um sinal de perigo forte será disparado.

Tabela 3 - Sinal de perigo em cada área de conhecimento

	AC 1		AC 2		AC 3	
AC corrente do aluno	$< t_{AC1}$	$>= t_{AC1}$	$< t_{AC2}$	$>= t_{AC2}$	$< t_{AC3}$	$>= t_{AC3}$
1	***fraco***	***forte***	*na*		*na*	
2	*ns*	***fraco***	***fraco***	***forte***	*na*	
3	*ns*	***fraco***	*ns*	***fraco***	***fraco***	***forte***

Quando o aluno alcança a segunda área de conhecimento, os erros da primeira área de conhecimento somente geram um sinal de perigo fraco se excedem ou igualam o valor do limiar t_{AC1} . Dessa forma prevê-se que o aluno poderia, por esquecimento ou erro de digitação, cometer poucos erros de conceitos ou comandos estudados na

primeira área. Considerando que a meta de aprendizagem corrente é a segunda área, assume-se para efeitos de diagnóstico que erros da primeira AC somente indicariam que o aluno precisa revisar conceitos já estudados e corrigir a sua solução. Para erros da segunda AC, o tratamento segue o mesmo conceito da primeira AC, visto anteriormente. Se o aluno cometer uma quantidade de erros que não atinja o limiar t_{AC2}, um sinal de perigo fraco será disparado. Porém, se o aluno comete um número de erros maior ou igual ao que o limiar t_{AC2} permite, uma sinal de perigo forte para a segunda área de conhecimento deve ser disparado. O mesmo ocorre para o tratamento da terceira área de conhecimento.

5. Avaliação do Caderno de Algoritmos

Com o objetivo de avaliar o ambiente Caderno de Algoritmos foi realizado um experimento baseado em princípios de avaliação de software educacional. Participaram da avaliação sete estudantes do curso de Licenciatura em Computação da Universidade de Caxias do Sul, com amplo conhecimento para realizar uma avaliação pedagógica e técnica em software educacional.

Cada avaliador individualmente respondeu um questionário sobre a sua percepção do ambiente Caderno de Algoritmos, simulando usuários do tipo aluno e professor. O instrumento de avaliação utilizado foi desenvolvido em trabalho prévio do grupo de pesquisa (Webber, Boff e Bonno, 2009), compreendendo vinte e quatro questões onde cinco (5) buscavam avaliar aspectos pedagógicos e dezenove (19) aspectos técnicos. Ao final do instrumento reservou-se um espaço para comentários e sugestões de melhoria.

Todos os avaliadores classificaram o ambiente como sendo do tipo Cooperativo e Interativo e ainda como um sistema de Exercícios e Prática. Outro ponto importante constatado foi que todos os avaliadores constataram que o sistema é adequado (80% totalmente e 20% parcialmente) ao nível do seu público-alvo. Em relação à variedade de atividades e níveis de complexidade das tarefas, os avaliadores confirmaram que o sistema possui suporte, mas que isso depende dos professores que serão os responsáveis por inserir no sistema as atividades que devem ser realizadas.

Em relação aos aspectos técnicos do ambiente de aprendizagem, os avaliadores entendem que o usuário possui controle sobre as suas atividades de forma a interrompê-las e retomá-las quando desejado. O ambiente de aprendizagem possui uma interface agradável e que estimula a sua utilização. Um ponto fraco assinalado foi o fato de que menus de ajuda ou dicas estão incompletos. Os avaliadores consideraram que o ambiente de aprendizagem pode ser utilizado pelos alunos sem a interferência do professor. Esse resultado é importante, pois um dos propósitos do sistema é que o aluno possa acessá-lo de qualquer lugar, sem a presença de um professor ao seu lado e que consiga utilizar o sistema sem encontrar dificuldades desmotivadoras da aprendizagem.

Uma sugestão de melhoria para o trabalho, sugerida por um avaliador, seria incluir um menu de ajuda com a sintaxe aceita pelo compilador do sistema. Durante a avaliação do ambiente de aprendizagem, foi detectado apenas um problema operacional O problema se tratou de execução de código no servidor da aplicação, quando uma solicitação de avaliação do algoritmo não obteve uma resposta do sistema. Os experimentos serviram para indicar o problema que foi analisado e corrigido pela equipe técnica.

6. Conclusões

O presente artigo propõe um ambiente para auxiliar o aprendizado de programação de computadores onde o estudante resolve problemas e o professor assume seu papel acompanhando os resultados e dificuldades do aluno. Embora ambientes similares já existam e estejam em uso, nossa proposta se baseia na aprendizagem baseada na resolução de problemas e no conceito de Caderno Virtual, onde o estudante encontra em um único ambiente funcionalidades de aprendizagem, de anotação e síntese da aprendizagem e de comunicação e colaboração entre pares.

O ambiente proposto abre mais espaço para o professor, permitindo que ele participe ativamente do processo de resolução de problemas. Desta forma o estudante pode contar com o auxílio 1) dos componentes de depuração e de diagnóstico automático do algoritmo, 2) dos pares para troca de informações e ajuda, e 3) do seu professor para a correção e observações. Estes três recursos integrados podem garantir o sucesso no processo de aprendizagem de programação. A avaliação preliminar por estudantes em Licenciatura em Computação mostrou que o ambiente tem recursos suficientes para o desenvolvimento de tarefas com acompanhamento presencial e à distância. Cabe agora o desenvolvimento de experimentos com alunos reais para demonstrar-se a sua real efetividade.

Referências

Baldwin, D. Three years experience with Gateway Labs. In: **Proceedings of Innovation and Technology in Computer Science Education Conference**, Barcelona, Spain, June 1996, ACM Press, 6-7, 1996.

Bridgeman, S.; Goodrich, M.T.; Kobourov, S.G.; Tamassia, R. PILOT: An Interactive Tool for Learning and Grading. In: **Proceedings of the SIGCSE Technical Symposium on Computer Science Education**, Austin, TX, March 2000, ACM Press, 139-143.

Cavalcante, R.; Finley T.; Rodger, S.H. A Visual and Interactive Automata Theory Course with JFLAP 4.0. In: **Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education**, Norfolk, VA, March 2004, 140-144.

Eclipse Foundation. Disponível em http://www.eclipse.org/. Acesso em Maio 2010.

Gomes, A.; Carmo L.; Bigotte, E.; Mendes, A. Mathematics and programming problem solving. In: **3rd E-Learning Conference,** Computer Science Education, Coimbra, September 2006.

Kumar, A. Generation of problems, answers, grade, and feedback case study of a fully automated tutor. In: **Journal on Educational Resources in Computing (JERIC)** volume 5 (issue 3), September 2005.

Schulze, K.G.; Shelby, R.N.; Treacy, D.J.; Wintersgill, M.C. Andes: A Coached Learning Environment for Classical Newtonian Physics. In: **Proceedings of the 11th International Conference on College Teaching and Learning**. Jacksonville, FL, April, 2000.

Scott, A.; Watkins, M.; McPhee, D. E-Learning For Novice Programmers - A Dynamic Visualisation and Problem Solving Tool. In: **Proceedings of the ICTTA**, Syrian Computing Society / IEEE, Damascus, Syria, 2008.

Souza, C. M. Visualg. Disponível em http://www.apoioinformatica.inf.br/visualg/. Acesso em Maio 2010.

Webber, C.; Viganó, E.; Possamai, R. Modelo Imunológico de Diagnóstico. In:

Encontro Nacional de Inteligência Artificial (ENIA 2009). Bento Gonçalves: SBC, 2009. p. 969-978.

Webber, C.; Boff, E.; Bonno, F. Ferramenta Especialista para Avaliação de Software Educacional. In: **Simpósio Brasileiro de Informática na Educação (SBIE 2009)**.

WEBportugol. Disponível em http://www.univali.br/webportugol. Acesso em Maio 2010.