

Um Sistema Tutor Inteligente para o Ensino no Domínio de Lógica de Programação

Idovaldo Cunha da Silva

Universidade Estadual do Maranhão
Cidade Universitária Paulo VI, s/n -
Tirirical, São Luís - MA, 65055-000
+55 99 982344968
idovaldo@gmail.com

Luís Carlos Costa Fonseca

Universidade Estadual do Maranhão
Cidade Universitária Paulo VI, s/n -
Tirirical, São Luís - MA, 65055-000
+55 98 981112222
lccfonseca@gmail.com

Reinaldo de Jesus da Silva

Universidade Estadual do Maranhão
Cidade Universitária Paulo VI, s/n -
Tirirical, São Luís - MA, 65055-000
+55 98 981007302
reinaldo.silvarrb@gmail.com

ABSTRACT

This work presents a computational model of an Intelligent Tutor System (ITS) to support teaching and learning in the programming logic domain. The STI used as artificial intelligence - IA technique, the Case Based Reasoning (Case-Based Reasoning - CBR), making use of a domain model knowledge base. Thus, this article discusses the design, modeling and prototyping of an Intelligent Tutor System to teach programming logic making use of a visual language. The investigation is in progress and on the assumption that the prototype completed will be enough to prove the concept that the STI BLOP, fully developed, will provide an interactive and rich learning environment for students which will result in a significant increase in their achievements.

RESUMO

Este trabalho objetiva apresentar uma modelo computacional de um Sistema Tutor Inteligente (STI), para suporte ao processo de ensino e aprendizagem no domínio de lógica de programação. O STI utiliza como técnica de Inteligência Artificial - IA, o Raciocínio Baseado em Casos (Case-based Reasoning - CBR), fazendo uso de uma base de conhecimento do modelo do domínio. Assim, este artigo aborda a concepção, modelagem e prototipação de um Sistema Tutor Inteligente para ensinar lógica de programação fazendo uso de uma linguagem visual. A investigação está em andamento e partir da hipótese de que o protótipo concluído será suficiente para provar o conceito de que o STI BLOP, totalmente desenvolvido, irá fornecer um ambiente de aprendizagem interativo e rico para os alunos o que resultará em um aumento significativo de suas conquistas.

Palavras Chaves

Algoritmo. Lógica de programação. Raciocínio Baseado em Casos. Sistema Tutores Inteligentes.

1. INTRODUÇÃO

Atualmente, é diverso o número de estudantes que se mostram interessados pelos cursos relacionados a área de computação. Porém, eles se diferem em muitos aspectos, tais como idade, sexo, nível de escolaridade, aptidão para resolver problemas lógicos e

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Conference'15, Month 1-2, 2015, City, State, Country.
Copyright 2015 ACM 1-58113-000-0/00/0010 ...\$15.00.

etc. Nesse contexto, observa-se a grande dificuldade em se criar um único curso que atenda a todas as necessidades de um público tão diferente. Apesar de uma metodologia na qual se trabalhasse com uma tutoria individualizada, de um tutor para um aluno, ser um meio adequado para abordar este problema, porém não é uma alternativa viável financeiramente. Uma solução muito melhor é usar Sistemas Tutores Inteligentes. Para [1], os STI se apresentam como os principais ambientes para auxiliar o processo de aprendizagem de programação a uma grande quantidade de pessoas com níveis e características tão distintos.

Os STI's são sistemas computacionais de ensino, que oferece tutoria um para um, que unem técnicas de IA com teorias pedagógicas para tutorar um aluno em determinado domínio, como por exemplo a programação de computadores, alterando sua interação com o aluno com base em características pessoais e individuais [10].

Um desafio importante aqui é que um problema de programação raramente tem uma solução única. Para que um STI seja eficaz, é necessário que seja capaz de lidar com muitas soluções alternativas para um determinado exercício de programação. Este artigo se concentra em atingir este objetivo utilizando as teorias de IA. Existem diferentes técnicas que implementam essas funcionalidades, como redes bayesianas, redes neurais, lógica fuzzy e também o raciocínio baseado em caso. O raciocínio baseado em casos é definido por [9], como um enfoque para a solução de problemas e para o aprendizado baseado em experiência passada. Sistemas CBR resolvem problemas ao recuperar e adaptar experiências passadas - chamadas casos - armazenadas em uma base de casos. Um novo problema é resolvido com base na adaptação de soluções de problemas similares já conhecidas [3].

Portanto, o objetivo desse artigo é propor o STI BLOP para auxiliar o processo de ensino e aprendizagem no domínio de lógica de programação a fim de facilitar o desenvolvimento de técnicas que cooperam para a produção de soluções logicamente válidas e coerentes, que resolva com qualidade os problemas que se deseja programar, contribuindo para identificação de dúvidas e erros dos alunos na construção de algoritmos.

2. REFERENCIAL TEÓRICO

Serão apresentados nessa seção alguns temas que darão fundamento teórico para o desenvolvimento do modelo computacional do STI, como: Dificuldade no ensino e aprendizagem de programação e técnicas utilizadas para o processo de ensino de lógica de programação.

2.1. Dificuldade no ensino e aprendizagem de lógica de programação

Nos cursos da área de computação as disciplinas relacionadas com o ensino e aprendizagem de programação de computadores são essenciais para todas as carreiras ligadas a área de informática. Essas disciplinas ocupam ponto de destaque e são essenciais para o bom desempenho dos alunos no decorrer do curso. Uma vez que a aprendizagem dos conceitos e da programação em si ocorrem praticamente, no decorrer de todo o curso, o baixo índice de assimilação dos estudantes nas disciplinas cujos requisitos exigem o conhecimento de programação tem sido um grande problema enfrentado em muitas instituições de ensino [7].

Apesar de várias metodologias propostas terem verificado melhorias nos índices de aprendizado no domínio de algoritmo, observou-se que a maioria das metodologias atuais não possibilitam tratar cada aluno de maneira diferenciada. Ou seja, as metodologias geralmente são aplicadas de maneira uniforme em turmas inteiras.

Tendo em vista a quantidade de alunos, a origem, as experiências e habilidades diferentes, tem-se que levar em conta que os alunos não são iguais. E isso é justificado, em partes, pelo fato de alunos de uma mesma classe, submetidos as mesmas condições de ensino, apresentarem resultados distintos [7], e reforça a necessidade do uso de técnicas variadas que permitam ampliar os resultados de ensino.

[7] afirma que, não é raro ouvir que, “algoritmo e programação não é para todos”. Esta é uma visão simplista daqueles que não questionam a maneira de ensinar. Porém, para uma razoável parcela da comunidade científica, surgem questionamentos sobre como ensinar algoritmos. São pesquisadores preocupados em entender o processo de aprender a programar, detectando falhas e dificuldades deste aprendizado e sugerindo alternativas, de modo a facilitar o aprendizado do aluno na disciplina de algoritmo.

2.2. Técnicas utilizadas para o ensino de lógica de programação

Para escrever programas de computador corretos, os alunos precisam entender conceitos abstratos, em seguida, convertê-los em soluções concretas [2]. O problema deve primeiro ser resolvido através de uma abordagem conceitual antes de um programa de computador pode ser escrito usando uma linguagem de programação particular. Ao fazê-lo, os alunos precisam utilizar as habilidades na concepção de programas, pensamento criativo e lógico. Ao criar uma solução, eles precisam concentrar-se simultaneamente sobre a sintaxe e a construção do algoritmo [2]. Isto significa que todo o processo requer a interação de muitas capacidades cognitivas, o que torna o processo muito difícil para principiantes.

O exemplo a seguir requer que o aluno faça um cálculo do fatorial de um número N inteiro. O cálculo do fatorial de um número N é obtido através da multiplicação de N pelos seus antecessores até se chegar ao número 1.

O fluxograma, é usado para mostrar de forma gráfica a lógica para resolver problemas, sendo que nesse processo destaca-se os passos individuais e o fluxo de execução [8]. A Figura 1 ilustra os passos usados no desenvolvimento de programa usando fluxograma para resolução do problema do fatorial de N.

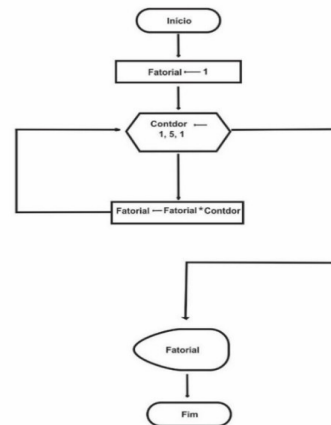


Figura 1 - Fluxograma: cálculo do fatorial de N

Porém, computadores não são programados usando desenhos. Existem, no entanto, algumas linguagens gráficas, mas são pouco utilizadas [7].

```

Program FATORIAL;
var
  I, N, FAT: integer;
Begin
  Writeln('Digite um número:');
  readln (N);
  FAT := N;
  For i:=1 to (N-1) do
    FAT := FAT * I;
  writeln('O fatorial de ', N, ' equivale a: ', FAT);
end;
    
```

Figura 2 - Linguagem Pascal fatorial de N

As linguagens de programação de auto nível, como mostrado na Figura 2, por outro lado, fazem uso de uma infinidade de palavras adicionais, como “program”, “var”, “begin” e “end”, e de um grande número de caracteres; e que podem, muitas vezes, acarretar um problema a mais para o aluno no aprendizado da disciplina. Esse é outro ponto que tem que ser levado em consideração quando se fala sobre o ensino e aprendizagem na disciplina de programação.

A linguagem visual em blocos, que consiste na criação de códigos onde o aluno pode arrastar e soltar blocos para escrever programa, mostra-se mais interessante e atraente para a finalidade do ensino na disciplina de lógica de programação, tendo em vista o pouco ou nenhum conhecimento da maioria dos alunos em programação, pois o aluno não precisará se preocupar com a sintaxe da linguagem quando estiver resolvendo os problemas. A linguagem visual consiste em que cada bloco corresponde a uma linha real em linguagem de código, sendo que depois o aluno pode utilizar esse código como base para o desenvolvimento de programas mais complexo. A Figura 3 mostra um código em linguagem visual em bloco para resolver o fatorial de N.

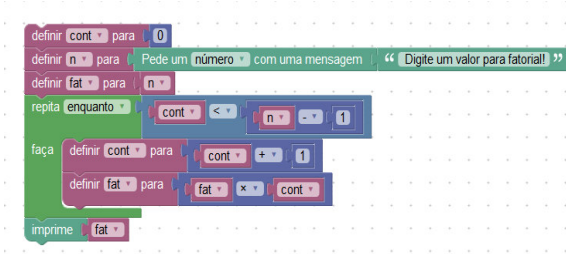


Figura 3 - Linguagem visual em blocos fatorial de N

Tendo em vista as características da linguagem visual e o foco da disciplina de lógica de programação, que está mais interessada na forma conceitual de organização das ideias dos alunos do que no aprendizado de sintaxe de linguagem é que optamos para fazer uso no desenvolvimento do STI BLOP dessa ferramenta visual de programação.

3. SISTEMAS TUTORES INTELIGENTES

Sistema Tutor Inteligente é um ambiente computacional de aprendizagem que possuem modelos de conteúdo instrucional que especificam o “que” ensinar e estratégias de ensino que especificam “como” ensinar [10]. Esses sistemas são capazes de acompanhar o aluno durante todo o processo de solução de um problema. Desta forma, enquanto o aluno aprende o conteúdo, o tutor aprende sobre o aluno levantando diversas informações sobre o mesmo, como, por exemplo, facilidades que ele apresenta sobre determinado assunto e dificuldades apresentadas quanto a outros. Com este aprendizado, o tutor pode prover um acompanhamento mais individualizado, apresentando explicações e exercícios mais bem relacionados às habilidades que o aluno não se sente tão seguro ou possui maiores dificuldades.

Geralmente, a arquitetura para um bom funcionamento de um Sistema Tutor Inteligente precisa ter muitos módulos. Uma classificação comum dos módulos que compõem um STI [6] é mostrada na Figura 4. A fim de compreender a funcionalidade de cada um destes módulos, vamos considerar uma situação em que o STI forneça um problema para um aluno resolver, como os passos a seguir. O problema é apresentado ao aluno através do módulo de comunicação que é o que lida com todas as interações entre o aluno e os STI. Em seguida, o aluno entra com sua solução do problema através do módulo de comunicação. O módulo pedagógico, em seguida, considera esta solução em um conjunto de informação obtida a partir dos módulos de estudantes e de domínio. O módulo de domínio contém detalhes sobre o assunto que é ensinado pelas STI e, portanto, contém informações sobre a solução correta para o problema. Com base nessas informações, o módulo pedagógico decide se a solução está correta ou não. O módulo de estudante contém informações sobre as características do aluno. O módulo pedagógico usa essa informação para decidir que tipo de retorno que deve fornecer ao aluno. Seja qual for a decisão do módulo pedagógico, o *feedback* é fornecido para o aluno através do módulo de comunicação. Enquanto isso, o sistema forma uma base de dados sobre o conhecimento do aluno em determinado assunto que está sendo ensinado pelo problema. Esta informação é atualizada para o módulo do estudante, a fim de ter um modelo mais preciso do aluno.

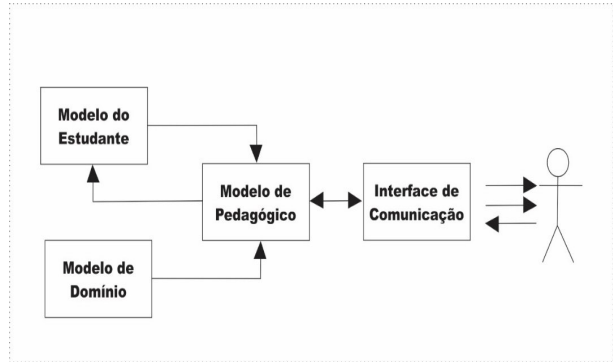


Figura 4 - Principais módulos de um Sistema Tutor Inteligente

Como podemos ver na arquitetura acima, um STI contém um módulo de estudante a fim de personalizar sua interação com o aluno. Para fazer isso, o sistema ideal precisa ter um bom conhecimento sobre o aluno, incluindo sua idade, sexo, capacidades, emoções e muitas outras características. Porém, o foco desta pesquisa não é sobre o projeto detalhado do módulo do estudante. Portanto, o módulo do estudante usado aqui considera apenas as características dos alunos que estão relacionadas diretamente com a aprendizagem: ou seja, o nível atual de conhecimento do aluno no assunto que está sendo ensinado.

4. STI PARA ENSINO NO DOMÍNIO DE PROGRAMAÇÃO

Muitos STI's para o ensino de programação de computador, tem sido desenvolvido em diferentes linguagens. No decorrer desse processo de desenvolvimento, diversos sistemas foram desenvolvidos para auxiliar alunos e professores no processo de ensino e aprendizagem de programação. Alguns desses sistemas são apresentados por [1], [5] e [4], que serão apresentados a seguir.

SQL-Tutor: que é um dos mais bem-sucedidos STI de todos os tempos e que ensina aos alunos conhecimento referentes ao domínio de teoria e design de banco de dados, procurando ensinar aos alunos como escrever consultas SQL.

Lisp Tutor: este STI serve para acompanhar o aluno durante o processo de resolução dos problemas e transcrição do código. O sistema possui uma sequência de regras de produção que é repassada para o aluno à medida que ele vai resolvendo o problema, caso haja deficiências durante a produção o aluno é informado e não é permitida a continuação até que seja resolvido aquele problema.

Algo-LC: nesse STI se diferencia dos outros Sistemas Tutores Inteligentes pelo fato de se ter um Companheiro de Aprendizagem, companheiro este que apoia o aluno durante a resolução dos exercícios, enviando mensagens o estimulando a verificar seus erros e corrigi-los e não demonstrando a resolução, como é feito em outros STI.

5. RACIOCÍNIO BASEADO EM CASO

Um sistema de CBR resolve problemas por adaptar soluções que foram utilizadas para resolver problemas anteriores. Segunda [3], o CBR é uma técnica de IA que tenta simular o funcionamento do cérebro humano, buscando solucionar um novo problema através da recuperação e adaptação de casos passados armazenados na

base de conhecimento. O raciocínio analógico reconhece similaridades entre diferentes domínios e, a partir delas, pode gerar novos conhecimentos. Essa técnica utiliza diferentes cálculos de medida de similaridade como meio de mensurar o quão semelhante um caso é de outro, considerando seus atributos e pesos associados a eles.

Ainda conforme [3], enquanto outras abordagens de IA utilizam conhecimento genérico na forma de regras e roteiros, o CBR utiliza exemplos específicos concretos para representar o conhecimento, que é utilizado como base para resolução de outros problemas similares.

O primeiro passo para utilizar uma solução já aplicada com sucesso anteriormente é determinar qual das experiências passadas mais se assemelha ao problema atual. Para ser possível realizar essa comparação, é necessário que as experiências sejam analisadas e armazenadas de forma organizada. Os aspectos importantes dos problemas devem ser isolados, rotulados e ordenados em uma base de casos de tal forma que possam ser utilizados para comparar a situação atual com as anteriores armazenadas.

Assim, a cada novo caso em um sistema de CBR deve reconhecer e definir o problema atual a fim de encontrar a nova solução. O sistema deve ser capaz de verificar a similaridade entre o caso atual e os casos armazenados na base de dados para, a partir deles, adaptar uma nova resolução para o problema dado.

6. BLOP PARA O ENSINO DE LÓGICA DE PROGRAMAÇÃO

O STI BLOP é uma ferramenta para dá suporte ao aluno a construir seus primeiros algoritmos e com isso adquirir conhecimento no domínio de lógica de programação. O STI utiliza uma linguagem visual em blocos que permite criar programas em blocos estruturados.

6.1. Descrição da Arquitetura BLOP

Na Figura 5, apresentamos o modelo de arquitetura genérica de processo de verificação de respostas fornecidas pelos alunos para BLOP.

A solução do aluno é analisada contra um conjunto de programas corretos armazenados na base de casos. Devido à complexidade envolvida com a análise semântica, é necessário restringir ao STI BLOP um pequeno subconjunto da linguagem de programação visual com a qual iremos trabalhar. A área de foco envolve a seguinte lista de conceitos básicos em linguagem visual em bloco: variáveis (declaração e uso local e global), operadores e estruturas de loops.

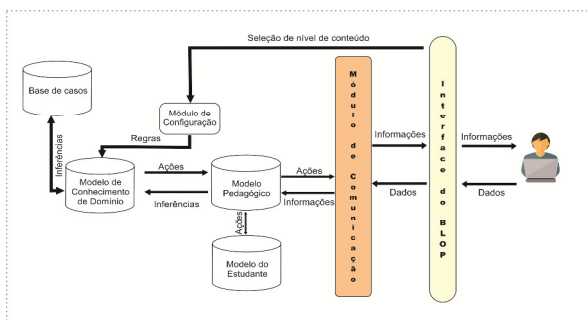


Figura 5 - Arquitetura de processos BLOP.

Este sistema armazenará várias soluções corretas para cada problema proposto. Quando um aluno apresentar uma solução para um dado problema de programação, o sistema verificará a similaridade entre as soluções proposta pelo o aluno e as solução armazenadas na base de casos, a fim de identificar os possíveis erros nas resoluções dos exercícios feitos pelos alunos. O *feedback* é então fornecido com base na análise de problemas anteriores. CBR pode ser visto como um ciclo de cinco tarefas seguintes: 1 - dado um novo caso, recuperar casos semelhantes a partir da base caso; 2 - comparação parcial dos casos da base com o problema atual; 3- ordenação dos casos selecionados na base de acordo com o valor da similaridade; 4 - avalie a solução e revê-la com base em quão bem ela funciona; 5 - Decida se deseja manter este novo caso na base de casos.

Problema: Escrever um programa chamado "soma", em que a soma de todos os números inteiros entre 1 e um número específico (N). Por exemplo, se N foram atribuído o valor 10, então a soma dos números de 1 a 10 é de 55.

Figura 6 mostra a especificações do programa:

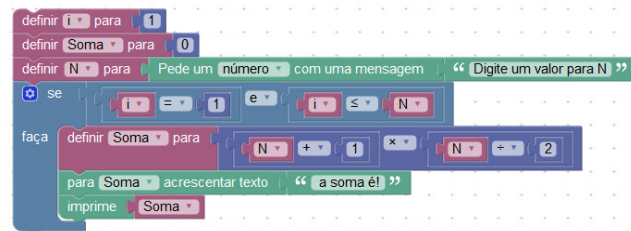


Figura 6 - Linguagem visual problema soma

Porém, pode-se reconhecer que existem várias possibilidades para as respostas dos alunos e o sistema não pode simplesmente listar respostas incorretas como mensagens de *feedback*. Mais tem que, baseado em casos anteriores sugerir possibilidades de formas de resolução do problema e apresentar para alunos. Por exemplo, a Figura 7 mostra como o aluno poderia escrever de forma diferente o mesmo problema:

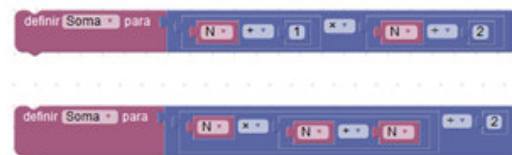


Figura 7 - Exemplos de resposta em linguagem visual

Ambas as respostas são completamente corretas e o sistema precisa reconhecer esses tipos de respostas e não apenas responder de volta para o aluno indicando uma falha. Testar a correção de um programa não é uma tarefa fácil, e não pode ser alcançado apenas dando um conjunto de respostas fixas, por isso o STI BLOP será desenvolvido para analisar experiências similares já vividas e apresentar novas formas de uma possível possibilidade de desenvolvimento dos exercícios.

Seja qual for o método de análise utilizado, o objetivo final de um sistema projetado para ensinar programação deve ser identificar corretamente os programas do aluno como correta ou incorreta e apresentar possibilidades, quando incorreto, de formas de alteração para que o aluno alcance êxito na resolução do problema. Os diferentes métodos de análise são bem-sucedidos em alcançar isso, em diferentes graus. Ao conceber o módulo do domínio de um STI para ensinar programação, é necessário selecionar um método que seja adequado para o sistema proposto.

6.2. STI BLOP

O STI BLOP é um sistema baseado na web que poderá ser acessado através de um navegador. Para utilizar o sistema, cada aluno deve criar uma conta de usuário com *login* e senha. Quando um aluno fazer *login* pela primeira vez, é solicitado para ele que realize um pré-teste para medir seu conhecimento atual sobre o domínio de lógica de programação. O pré-teste é um conjunto de questões de múltipla escolha, onde o aluno poderá deixar em branco as questões das quais não sabe a resposta para a pergunta. E é possível até mesmo não responder a qualquer pergunta se o aluno não tem conhecimento sobre o assunto.

Uma vez que um aluno tenha concluído o pré-teste, ele é direcionado para a página de seleção de exercícios. Esta página também é exibida no início de cada sessão seguindo uma ordem de nível desde o pré-teste. O aluno escolhe um exercício para tentar resolver e, em seguida, insere o código em linguagem visual com a solução do exercício como mostrando na Figura 8. Quando solicitado, o sistema fornece *feedback* para o aluno. Ao aluno, também é permitido abandonar o exercício atual e retornar para a página de seleção de exercício a qualquer momento.

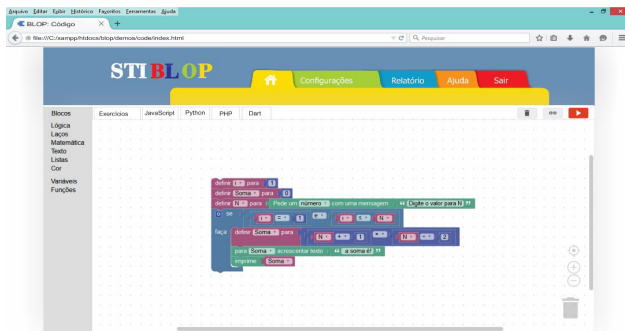


Figura 8 - Página de resolução de exercícios

Na parte superior da página o sistema exibe um menu, como mostrado na Figura 9. Nesse menu, é permitido ao aluno escolher entre algumas opções. Ele pode sair do sistema ou alterar as configurações do nível de exercícios e senha. O link "ajuda" traz algumas páginas de informação sobre como usar o sistema. Também será permitido ao aluno visualizar um relatório individual de desempenho, através de uma página que exibirá o seu conhecimento atual sobre o domínio dos temas abordados pelos STI BLOP, o relatório será aferido pelo sistema, com base nas resoluções de atividades realizada pelo aluno.

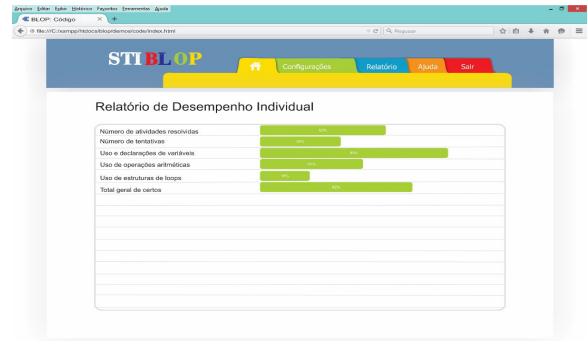


Figura 9 - Página de relatório de desempenho individual

Uma das principais vantagens do ITS BLOP é que ele orientará cada aluno para temas que são mais adequados para o seu nível atual de conhecimento. Esta orientação é feita através da lista de exercícios. O sistema exibe uma lista de exercícios que ele acha que são mais adequados para o aluno que está logado no sistema. Os exercícios são mostrados em ordem de adequação, com destaque aos exercícios mais adequados ao nível de conhecimento do aluno. Este é também o exercício que é selecionada por padrão. O estudante pode decidir tentar um outro exercício da lista, se ele desejar. A página de seleção de exercício das STI BLOP é mostrada na Figura 10.



Figura 50 - Página de seleção de exercícios

Embora o STI recomende ao aluno exercícios baseados em seu nível atual de conhecimento sobre um determinado assunto, pode haver momentos em que os alunos não concordar com as informações do sistema sobre o seu nível de conhecimento e pode querer ser encarregado de selecionar os próximos exercícios que queiram resolver. Nesses casos, eles podem selecionar um exercício diferente na lista exercícios, mas uma vez que esta lista pode ser muito extensa, eles podem achar que é difícil encontrar o que querem. Uma opção de pesquisa é fornecida para este propósito. Se o estudante decide procurar um exercício, a página mostrada na Figura 11 é exibida. O aluno pode agora escolher qual tema precisam ser cobertos pelos exercícios que ele queira tentar resolver. Também é possível escolher exibir os exercícios que já foram tentadas, ou os que ainda não foram tentados, ou ambos e exercícios desenvolvidos com êxito, ou os não desenvolvidos, ou ambos.

O sistema permitirá a liberação de exercícios em lotes para os alunos. Isso é útil porque muitos exercícios liberados de uma única vez, pode ser demais para alguns alunos. Em cada exercício pode ser atribuída uma data de lançamento. Exercícios que tenham sido libertados, mas que tem uma data de lançamento maior do que uma data especificada, são apresentados como novos exercícios. O aluno também pode selecionar a exibição de novos exercícios, que não são novos ou ambos. Uma vez que o aluno escolhe os critérios de pesquisa necessários, eles podem voltar à página de seleção de exercícios. Agora, esta página exibe uma lista de exercícios que correspondem aos critérios de pesquisa. O aluno pode escolher o exercício que ele quer tentar resolver.

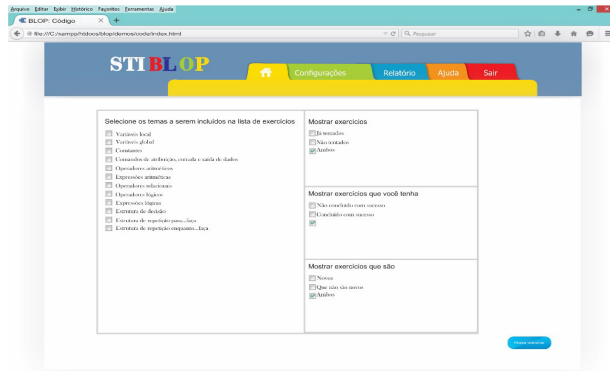


Figura 11 - Página de selecionar de exercícios

Uma vez que um aluno tenha escolhido se os próximos exercícios devem ser sugeridos pelo sistema ou se o próprio aluno deve procura-los, esta configuração desse modo de apresentação dos exercícios já permanecem ativas para a sessão de *login* atual, e para as sessões futuras, a menos que o aluno mude explicitamente as configurações novamente. Isto torna mais fácil para cada aluno trabalhar com base em sua preferência de apresentação dos exercícios sem ter que escolher o modo em outras vezes que fizer *login* no sistema.

Uma consideração importante neste STI está relacionada ao fato de como serão fornecidos, pelo sistema, o *feedback* ao aluno, se o *feedback* deverá ocorrer de forma proativa pelo sistema ou se o sistema deve esperar para que seja solicitado pelo aluno. Embora, muitas ideias diferentes terem sido apresentadas na literatura, este trabalho está baseado na ideia de que somente deverá ser fornecido o *feedback* ao aluno, se ele assim o deseja. Portanto, quando um aluno enviar uma solução a ser analisada pelo sistema, a única forma de *feedback*, inicialmente fornecida pelo sistema, é se a solução está correta ou incorreta.

7. RESULTADOS ESPERADOS

Ensinar programação introdutória é um grande desafio para os educadores, por muitas razões. Embora muitos métodos tenham sido sugeridos com a finalidade de transpor este desafio, que continua a ser um grande problema. Em particular, poucas pesquisas foram realizadas sobre os métodos de ensino de lógica de programação para alunos iniciantes na área com pouco ou nenhum conhecimento no assunto. Nesta perspectiva, uma solução para este problema, tal como sugerido pela presente investigação, é a utilização de sistemas tutores inteligentes para

esta finalidade. O STI BLOP é um sistema que se concentra em ensinar as noções básicas de lógica de programação para alunos iniciantes no curso da área de computação. Ele fornece exercícios para os alunos com base em suas necessidades específicas, a fim de maximizar sua aprendizagem.

Foi descrito nesse trabalho o modelo computacional do STI BLOP. O BLOP está sendo projetado para o ensino no domínio de lógica de programação com a finalidade de ajudar alunos a superar suas dificuldades na apropriação de tal conhecimento. O BLOP, além de possibilitar aos alunos apropriação de conhecimento no domínio de lógica de programação, tem a pretensão de administrar automaticamente problemas gerando dinamicamente e adaptando *feedback* em tempo de execução aos alunos.

8. REFERÊNCIAS

- [1] BOTELHO, Carlos Alberto. Sistemas Tutores no domínio da programação. Revista Informática Aplicada v IV nº1 jan/jun 2008.
- [2] GOMES, A., & MENDES, A. J. (2007). Learning to program - difficulties and solutions. International Conference on Engineering Education – ICEE. Disponível em: <<http://www.ineer.org/Events/ICEE2007/papers/411.pdf>>. Acesso em: 30 jul. 2015
- [3] MENDES, J. B. Um framework de raciocínio baseado em casos aplicados para estruturar a base de conhecimento em sistemas tutores inteligentes. / por Joice Barbosa Mendes. -- Itajubá (MG) : [s.n.], 2012. 147 p. : il.
- [4] NOBRE, I. A. M. N., MENEZES, C. S. “Suporte à Cooperação em um Ambiente de Aprendizagem para Programação (SAmbA)”. XIII Simpósio Brasileiro de Informática na Educação – SBIE 2002. São Leopoldo, 2002.
- [5] PETRY, Patrícia Gerent. Um sistema para o ensino e aprendizagem de algoritmos utilizando um companheiro de aprendizagem colaborativo. 2008. 86p. Tese (Mestrado em Ciência da Computação), Universidade Federal de Santa Catarina. Florianópolis, 2005.
- [6] PILLAY, N. (2003). Developing intelligent programming tutors for novice programmers. Disponível em: <<http://delivery.acm.org/10.1145/790000/782986/p78-pillay.pdf>>. Acesso em: 10 jun. 2015.
- [7] PIMENTEL, E. P.; FRANÇA, V. F.; OMAR, N. A Caminho de um Ambiente de Avaliação e Acompanhamento Contínuo da Aprendizagem em Programação de Computadores. II Workshop de Educação em Computação e Informática do Estado de Minas Gerais, 2003. Acessado em 29-12-2004.
- [8] VALENTIN, Henryethe, Koscianski, André. Um Estudo sobre o Ensino-aprendizagem de Lógica de Programação. Disponível em: <<http://posgrad.fae.ufmg.br/posgrad/viiienpec/pdfs/137.pdf>>. Acesso em: 05 abr. 2015.
- [9] WANGENHEIM, C. G.; WANGENHEIM, A. V. Raciocínio Baseado em Casos. 1. ed. [S.l.]: Manole, 2003.
- [10] Woolf, B. P. (2009). Building Intelligent Interactive Tutors: student-centered strategies for revolutionizing e-learning. Burlington, MA: Morgan Kaufman.