Ferramenta Computacional de Apoio ao Processo de Ensino-Aprendizagem dos Fundamentos de Programação de Computadores

Wilson Castello Branco Neto
Universidade do Planalto Catarinense
Departamento de Ciências Exatas e Tecnológicas
Lages, SC, Brasil, 88509-900
castello@uniplac.net

Aguinaldo Antonio Schuvartz
Universidade do Planalto Catarinense
Sistemas de Informação
Lages, SC, Brasil, 88509-900
castello@uniplac.net

Abstract. *The use of computational environments for teaching programming requires studies to learn the difficulties faced by students and teachers in the teaching-learning process, the definition of the main contents involved in it, as well as the different students' profiles to apply teaching strategies to assuage those difficulties. This article presents the proposal of a computational tool which makes use of an applied teaching methodology based on analogies between programming concepts and daily life common situations. Then, the Intelligent Tutoring Systems technology is used to make this tool more influential in the teaching-learning process.*

Resumo. *O uso de ambientes computacionais para o ensino de programação exige estudo para o conhecimento das dificuldades encaradas por alunos e professores no processo de ensino-aprendizagem, a definição dos principais conteúdos envolvidos, bem como dos diferentes perfis de alunos a fim de aplicar estratégias de ensino para amenizar tais dificuldades. Este artigo apresenta a proposta de uma ferramenta computacional que faz uso de uma metodologia de ensino baseada em analogias entre os conceitos de programação e situações comuns do dia-a-dia. Utiliza-se então a tecnologia de Sistemas Tutores Inteligentes visando tornar a ferramenta mais atuante no processo de ensino-aprendizagem.*

1. Introdução

Ambientes computacionais podem ser uma alternativa de auxílio ao processo de ensino-aprendizagem, pois facilitam a assimilação pelos alunos dos conteúdos ministrados e tornam esses processos mais dinâmicos, ágeis e prazerosos. A utilização de ferramentas computacionais no ensino prende mais a atenção do aluno, aproximando a teoria da prática e contribuindo para o aprendizado [Silva 2003].

Porém, um desafio existente na área da informática aplicada à educação é como fazer com que ferramentas computacionais realmente tenham um impacto no

aprendizado. Ainda há muito que se estudar e desenvolver nessa área, para que esses ambientes tenham maior aceitação por parte dos professores e alunos.

Os cursos da área de computação e informática enfrentam um grande problema com as disciplinas de introdução à programação de computadores, as quais visam ensinar como utilizar o computador para solucionar problemas. Acadêmicos iniciantes, ao se depararem com a disciplina, sentem-se incapazes de programar, devido ao conjunto de habilidades que a programação exige como capacidade para solucionar problemas, raciocínio lógico, habilidade matemática, capacidade de abstração, entre outras.

Os professores, diante disso, esforçam-se para aplicar a estratégia de ensino mais adequada que atenda às necessidades de aprendizagem de um grupo de alunos que, na maioria das vezes, está tendo que encarar tais habilidades e dominá-las pela primeira vez.

Tal problema acarreta em um alto índice de reprovação nas disciplinas iniciais de programação, a evasão do curso e a aversão à programação, onde o aluno, anos mais tarde, tende a procurar opções de trabalho que não envolvam esta atividade. Isto representa uma situação triste, pois se sabe que computadores são praticamente inúteis sem programas e sem programadores para desenvolvê-los [Jenkins 2002].

Dentre as tecnologias computacionais utilizadas como base para o desenvolvimento de ambientes virtuais de ensino, tem-se a Inteligência Artificial (IA), mais especificamente a área de Sistemas Tutores Inteligentes (STI).

Pretende-se então, nesse artigo, apresentar o desenvolvimento de uma ferramenta computacional, que utiliza técnicas de IA, para servir de apoio ao processo de ensino-aprendizagem dos fundamentos de programação de computadores.

O artigo está estruturado da seguinte forma: primeiramente, tem-se o referencial teórico do trabalho, onde é apresentada a conceitualização dos temas estudados, em seguida é apresentada metodologia pela qual a ferramenta foi desenvolvida, bem como a sua estrutura, faz-se então a apresentação do sistema descrevendo-se o ciclo do processo de ensino-aprendizagem presente no ambiente. Por fim, tem-se um estudo comparativo entre trabalhos relacionados e apresentam-se as considerações finais com os resultados esperados e sugestões para trabalhos futuros.

2. Informática no Ensino de Fundamentos de Programação

Na sala de aula, independente da disciplina, o professor depara-se com alunos apresentando graus de conhecimento e habilidades heterogêneas, ou seja, alunos com perfis de aprendizagem distintos. Considerando-se que alunos diferentes aprendem de formas diferentes, é preciso atender às necessidades de aprendizagem de todos, o que nem sempre acontece. Segundo [Pimentel, França e Omar 2003], "a mesma aula é dada para quem sabe muito, pouco, ou nada sobre determinado tópico. Isto gera um círculo de injustiças que condena muitos alunos à não-aprendizagem, gerando sucessivas reprovações e/ou colocando no mercado de trabalho profissionais sem uma formação consistente".

Por isto, é necessário que os professores tenham a sensibilidade para identificar estas diferenças, a fim de se adaptar, ou seja trabalhar de forma distinta com alunos que apresenta habilidades e conhecimentos diferentes sobre um dado assunto.

Ferramentas computacionais de ensino também devem adaptar-se às necessidades de cada perfil de aluno em particular e, para isso, é necessário separá-los em grupos de aprendizes que possuam mais ou menos as mesmas características, o que é considerado um grande desafio. Ou ainda, providenciar um atendimento individualizado, respeitando o ritmo de aprendizagem de cada usuário do ambiente. Segundo [Pimentel, França e Omar 2003], "é necessário determinar continuamente o que o aluno conhece e ensiná-lo de acordo, colocando-o numa situação de aprendizagem ótima".

Deve-se, ainda, considerar que o aluno não possui um perfil estático ao longo da utilização de um sistema educacional, sendo que em determinado momento pode migrar de um perfil para outro. Determinar os conhecimentos dos aprendizes antes, durante e após as sessões de treinamento é um fator crítico na construção de ambientes adaptáveis à heterogeneidade dos estudantes [Noronha *et al*. 2003].

Para cada perfil de aluno identificado é necessário aplicar ou desenvolver uma estratégia de ensino que melhor atenda às necessidades de aprendizagem daquele perfil em particular. Para o ensino de programação, [Rodrigues Júnior, 2004], propõe uma mudança na metodologia, onde temas como motivação, mudança na forma de avaliação, relacionamento professor-aluno, material utilizado e preparação das aulas devem ser revistos.

Já [Dunican 2002], propõe uma técnica baseada na utilização de exemplos que sejam familiares aos estudantes para explicar com mais facilidade os conceitos abstratos da programação, sendo esta a estratégia de ensino adotada e adaptada ao STI apresentado neste artigo.

Após a identificação de uma estratégia de ensino, é necessário realizar um levantamento dos principais conteúdos relacionados à disciplina, no caso deste trabalho os conceitos mais relevantes relacionados aos fundamentos de programação, e aplicar a estratégia de ensino escolhida para ensiná-los.

A ferramenta proposta é desenvolvida baseado na utilização de técnicas de IA, um ramo da ciência da computação que busca reproduzir nas máquinas a inteligência humana para solucionar problemas. Segundo [Bittencourt 2001], "o objetivo central da IA é simultaneamente teórico – a criação de teorias para a capacidade cognitiva – e prático – a implementação de sistemas computacionais baseados nestes modelos".

A área da IA utilizada como base para o desenvolvimento do ambiente são os STI, ferramentas de ensino computacioanis que buscam manipular algumas das capacidades cognitivas dos alunos e utilizar esses resultados como base para as decisões pedagógicas a serem tomadas. Os STI são ambientes educacionais adaptados ao aluno, na forma e no conteúdo, o que ajuda a superar um dos principais problemas encontrados no desenvolvimento de *softwares* educativos [Vicari e Giraffa 2003].

A arquitetura clássica de um STI é formada por três módulos distintos e complemetáveis que são: o Módulo Domínio, que constituí-se dos conteúdos e material que são trabalhados com o estudante; o Módulo Aluno, o qual apresenta as características e o conhecimento individual do aluno em um dado mometo; e o Módulo Tutor, que detém as estratégias e táticas de ensino que são selecionadas de acordo com as características de cada aluno. Além dos três módulos, tem-se ainda a interface que intermedia a relação entre o aluno e o sistema [Vicari e Giraffa 2003].

3. Desenvolvimento e Estrutura do Sistema

O ambiente educacional foi desenvolvido através da aplicação da estratégia de ensino baseada em analogias, ou seja, os exemplos dos conceitos de programação estudados são baseados em situações comuns ao dia-a-dia do aluno, apoiadas por imagens ilustrativas. Com isso, pretende-se instigar o usuário do sistema a fazer ligações dos conceitos abstratos da programação à situações às quais ele já está familiarizado, o que facilita a aprendizagem.

Como exemplo dessa estratégia de ensino baseada em analogias, pode-se apresentar o exemplo ilustrativo para a definição do conceito de programa de computador, que é associado à tarefa de trocar uma lâmpada. Essa situação é composta por diversos passos como: pegar uma escada, subir na escada, retirar a lâmpada queimada, colocar a lâmpada nova. Essa tarefa deve possuir um início e um fim e, para que a mesma seja realizada de forma efetiva, os passos devem estar ordenados de forma correta.

Da mesma forma funciona um programa computacional. Os passos para a realização dessa tarefa são realizados através de uma seqüência linear, de cima para baixo e da esquerda para a direita, exatamente como um programa é executado, exemplificando ainda o conceito de estrutura seqüencial.

A metodologia de ensino, deve então seguir o seguinte fluxo: primeramente é disponibilizado ao estudante a definição dos principais conceitos dos fundamentos de programação, em uma linguagem clara e objetiva que evite a citação de termos técnicos desconhecidos dos iniciantes. Os conceitos apresentados são sempre seguidos pelos exemplos apoiados por imagens. Em seguida são propostos exercícios de múltipla escolha que visam avaliar o entendimento do estudante no conceito corrente. Em seguida são propostos problemas computacionais, nos quais, durante a resolução, é permitido ao usuário selecionar e ordenar os passos, pré-determinados pelo sistema, necessários à solução.

3.1. Os Módulos Domínio e Aluno

Como visto anteriormente, o Módulo Domínio, de uma forma geral, compreende o conteúdo que o STI apresenta ao usuário. No caso do protótipo desenvolvido neste trabalho, o módulo domínio possui o conteúdo relacionado aos fundamentos de programação de computadores e é representado pelas entidades *Conceitos*, *Passos*, *Situações*, *Imagens*, *Exercícios* e *Alternativas* presentes no modelo conceitual da base de dados do STI (Figura 1).

Cada entidade do Módulo Domínio é responsável por armazenar um tipo de informação existindo também relacionamentos entre elas. A entidade *Conceitos* armazena a descrição dos conceitos principais e secundários possuindo uma relação consigo mesma, o que indica que cada conceito pode estar relacionado a outros conceitos sejam eles principais ou secundários.

A endidade *Passos* armazena os passos que compõe as situações apresentadas aos alunos, tais passos estão relacionados aos conceitos cadastrados. As situações por sua vez são armazenadas na entidade *Situações*, podendo ser selecionadas como exemplos ou como problemas a serem aplicados aos alunos. Cada situação possui uma ligação com um conjunto de passos que a constuitui.

A entidade *Imagens* é destinada a manter o nome e o caminho (diretório) das imagens que ilustram as situações selecionadas como exemplo, sendo que cada imagem é ligada a um passo. Já as entidades *Exercícios* e *Alternativas* armazenam os exercícios de múltipla escolha e suas alternativas respectivamente.

A figura 1 apresenta o modelo conceitual da base de dados do STI, o Módulo Domínio em conjunto com o Módulo Tutor com suas entidades e relacionamentos descritos anteriormente.

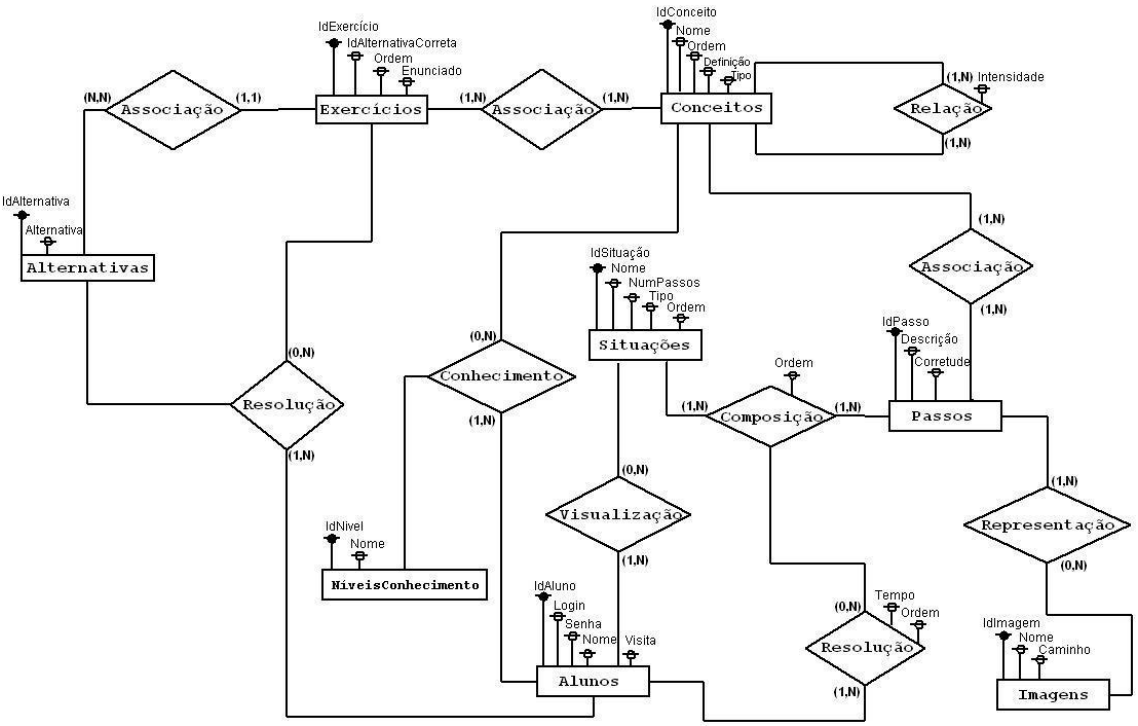

Figura 1. Modelo conceitual da base de dados do STI

O Módulo Aluno visa armazenar informações sobre os usuários, permitindo que eles sejam identificados pelo sistema e recebam tratamento adequado de acordo com seu nível de conhecimento. Este módulo está fortemente ligado ao Módulo Domínio e, em conjunto com ele, serve de base para que o Módulo Tutor possa determinar qual estratégia de ensino aplicar a cada estudante.

No Módulo aluno existe a entidade *Alunos* destinadas a armazenar informações sobre o aluno que são seu *login*, senha e nome. Ela está relacionada à endidade *Situações*, o que permite identificar quando um aluno visualiza uma determinada situação. Já o relacionamento com as entidades *Exercícios* e *Alternativas* permite armazenar informações pertinentes à resolução de exercícios pelos alunos, enquanto que a ligação com as entidades *Situações* e *Passos* armazena informações relativas à solução de problemas.

A entidade *NíveisConhecimento* tem por objetivo armazenar o nível de conhecimento do aluno sobre um dado conceito em um determinado momento de utilização do sistema, essa informação é obtida através da verificação das visualizações de conceitos e exemplos e da resolução dos exercícios e problemas realizados pelo aluno e permite ao STI guiar o estudante pelos conteúdos de acordo com o seu grau de aprendizagem. Para tornar isso possível, tem-se o relacionamento *Conhecimento* que faz

a ligação entre as entidades *Alunos*, *Conceitos* e *NiveisConhecimento*.

3.3. O Módulo Tutor

A função do Módulo Tutor é a de aplicar a estratégia de ensino mais adequada a cada usuário do STI. Para tanto, esse módulo deve trabalhar em conjunto com os demais Módulos, Aluno e Domínio, que retêm informações úteis sobre os usuários e sobre o conteúdo que está sendo ensinado pelo sistema respectivamente.

Para realizar a classificação do nível de conhecimento do aluno e aplicar a estratégia de ensino que o situe em uma situação de aprendizagem ótima, onde o sistema conhece o que o aluno estudou, o que ele aprendeu e qual deve ser o próximo conteúdo a ser apresentado, é utilizada a taxonomia de classificação de objetivos de aprendizagem desenvolvida por [Bloom 1976].

Essa classificação é dividida em seis categorias que são: conhecimento, compreensão, aplicação, análise, síntese e avaliação, cuja divisão ocorre em ordem crescente de complexidade e de acordo com as exigências das operações mentais que cada objetivo de aprendizagem requer. Das categorias apresentadas apenas as três primeiras são utilizadas.

O aluno é enquadrado na categoria Conhecimento quando visualizar um determinado conceito de programação e seu respectivo exemplo. Essa categoria é a menos exigente e compreende a fase inicial da aprendizagem onde o aluno tem apenas o conhecimento da existência de um determinado conceito.

Para ser enquadrado na segunda categoria Compreensão o aluno deve, através do alcance de um certo nível de aproveitamento na realização dos exercícios de múltipla ecolha, demonstrar entendimento sobre o conceito estudado, sendo capaz de realizar associações, ligações e elaborar explicações sobre o mesmo.

Para atingir a última categoria, Aplicação, o aluno deve ser capaz de prever, preparar e aplicar os conceitos adquiridos na resolução de novos problemas. Isso será avaliado por meio de seu desempenho na realização dos problemas propostos pelo ambiente.

4. Apresentação do Sistema

O sistema possui dois conjuntos de interfaces que são as interfaces para cadastro e as interfaces do Módulo Tutor. As interfaces destinadas aos cadastros são utilizadas pelo administrador do sistema com o intuito de cadastrar conceitos, situações, passos, imagens, exercícios e alternativas os quais compõem o Módulo Domínio do STI; para acessá-las o administrador deve possuir um *login* e uma senha.

As interfaces do Módulo Tutor são as utilizadas pelos alunos para o estudo dos fundamentos de programação de computadores, sendo que para acessar o sistema, o aluno deve possuir um *login* e uma senha. É neste módulo que o tutor entra em ação apresentando aos usuários conceitos e exemplos, corrigindo exercícios e problemas, verificando o nível de conhecimento de cada estudante e direcionando-o pelos conteúdos durante o processo de ensino-aprendizagem de acordo com seu desempenho. A figura 2 apresenta a janela *Estudando Conceito* presente no Módulo Tutor.

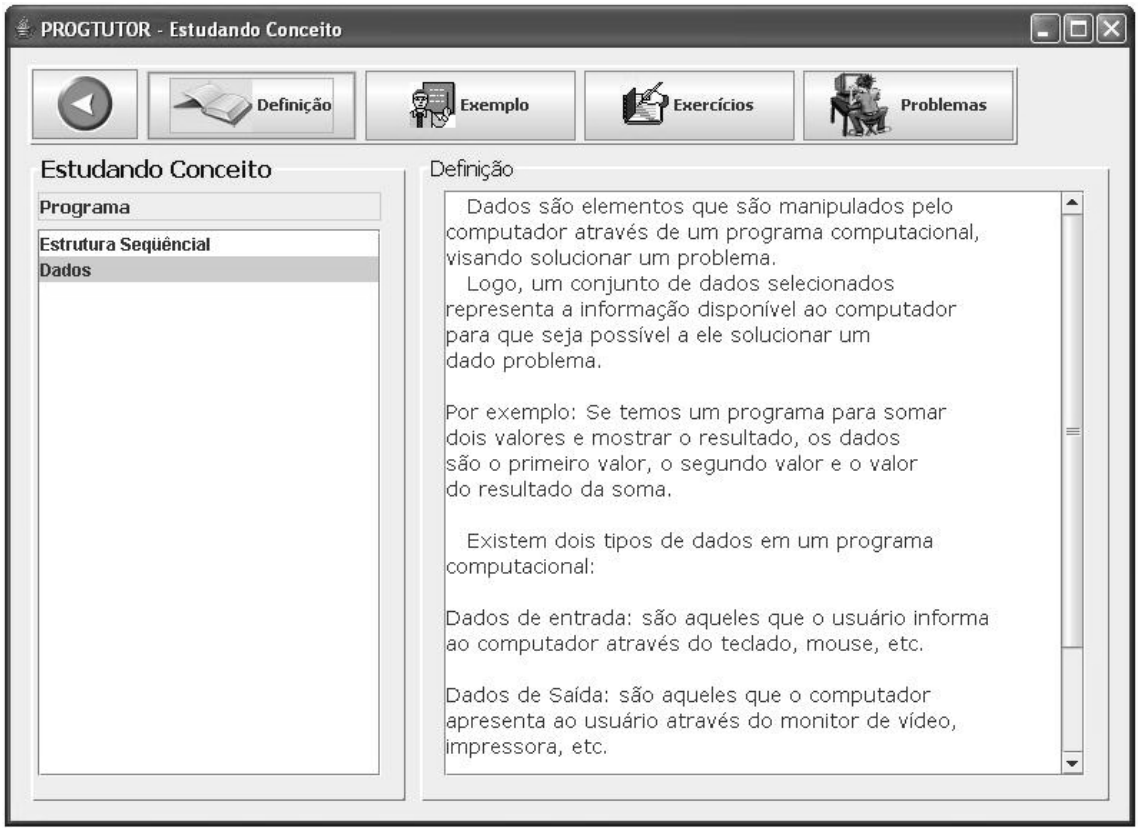

Figura 2. Janela Estudando Conceito

5. Trabalhos Relacionados

Nesta seção são apresentados alguns trabalhos relacionados, ou seja, exemplos de STI já desenvolvidos para auxiliar no processo de ensino-aprendizagem de programação.

[Tobar *et al*. 2001], projetou e desenvolveu uma arquitetura de ambiente colaborativo para o aprendizado de programação que tem por objetivo oferecer um conjunto de mecanismos que possam ser utilizados por estudantes e professores de forma colaborativa, oferecendo meios para resolução de problemas computacionais, acesso à informações *on-line* úteis e discussão de idéias, visando solucionar os problemas encontrados no processo de aprendizagem de programação nas disciplinas introdutórias.

A arquitetura SAAP (Sistema de Apoio à Aprendizagem de Programação), desenvolvido por [Castro *et al*. 2002], é direcionada a cursos de introdução à programação que utilizam o paradigma funcional como base. Trata-se de um ambiente cujo enfoque está na interação com os estudantes e na identificação de padrões de soluções de problemas com o intuito de melhorar o aprendizado e fornecer *feedback* adequado aos alunos.

[Menezes e Nobre 2002], projetou um ambiente cooperativo para apoio a cursos de introdução à programação, o qual utiliza a tecnologia de agentes e tem como principal objetivo prover a interação entre os alunos através de um ambiente que permite a colaboração e a cooperação, visando auxiliá-los na solução de problemas. É destinado a cursos orientados à resolução de problemas, no entanto, o foco é direcionado aos cursos de introdução à programação.

6. Considerações Finais

O objetivo principal do presente trabalho é o desenvolvimento de uma ferramenta computacional, que empregue em sua arquitetura técnicas de IA, para auxiliar alunos e professores no processo de ensino-aprendizagem dos fundamentos de programação de computadores. A utilização da tecnologia de STI visa tornar o ambiente de ensino aqui desenvolvido mais ativo, oferecendo suporte aos estudantes e auxiliando-os a amenizar as dificuldades encontradas no decorrer de seus estudos para a aquisição das habilidades que a programação requer.

Pretende-se então disponibilizar um ambiente computacional para ser utilizado em aula ou extra-classe que permita ao estudante aprender no seu próprio ritmo, disponibilizando definições de conceitos, exemplos ilustrativos e atividades, indicando ao aluno os pontos nos quais ele possui maior dificuldade e direcionando-o nos seus estudos.

Espera-se que tal ferramenta possa, no futuro, ser aplicada aos alunos, motivando-os no estudo de programação e contribuindo para a diminuição da aversão a mesma e a evasão dos cursos da área de computação e informática.

Para o desenvolvimento do STI, optou-se por aplicar a todos os alunos a mesma estratégia de ensino, baseada em analogias, considerando-se que a divisão dos mesmos em grupos homogêneos (definição de diferentes perfis de alunos) e a aplicação de uma estratégia de ensino distinta para cada grupo, tornaria a implementação da ferramenta muito complexa, o que demandaria um maior tempo para seu desenvolvimento.

O Módulo Tutor do STI ainda não foi desenvolvido devido à complexidade das técnicas de IA, as quais exigem prática, maior tempo para estudo e implementação e pelo fato de a tecnologia de STI ser uma área nova entre os desenvolvedores da ferramenta. No momento, o sistema encontra-se em fase de desenvolvimento, possuindo alguns exemplos de conceitos, situações, exercícios e problemas cadastrados e sendo apresentados aos usuários, bem como efetuando correções nos exercícios e problemas.

Para trabalhos futuros, além da implementação do Módulo Tutor, sugere-se uma pesquisa mais aprofundada sobre os diferentes perfis de alunos, estratégias de ensino distintas que atendam a tais perfis e a implementação das mesmas. Sugerem-se também estudos para a extensão da ferramenta, visando disponibiliza-la para estudantes que encontram-se em níveis mais elevados no estudo da programação como estrutura de dados e programação orientada a objetos.

Outra sugestão é um estudo na área de compiladores, com o intuito de incrementar o sistema para que este não permita somente selecionar passos e instruções lógicas, mas também possibilite aos estudantes escreverem seus próprios comandos. Isso permitiria ao STI identificar erros mais específicos dentro do código e realizar uma melhor avaliação e orientação do aluno pelos conteúdos disponíveis no Módulo Domínio.

7. Referências Bibliográficas

Bloom, B. S.; *et. al.* (1976). "Taxionomia de Objetivos Educacionais"**.** v. 1. Porto Alegre: Globo.

Bittencourt, G. (2001). “Inteligência Artificial – Ferramentas e Teorias”. 2. ed. Florianópolis: Editora da UFSC.

Castro, T. H. C. de. *et al*. (2002). “Arquitetura SAAP: Sistema de Apoio à Aprendizagem de Programação”. In: Workshop de Informática na Educação-WEI, 8. Florianópolis. Anais do XXII Congresso da Sociedade Brasileira de Computação. Florianópolis: SBC/UFSC. p. 267-276.

Dunican, E. (2002). “Making The Analogy: Alternative Delivery Techniques for First Year Programming Courses”. In: Workshop of the Psychology of Programming Interest Group, 2002. Proceedings… Carlow: Brunel University, 2002. p. 89-99.

Jenkins, T. (2002). “On the difficulty of learning to program”. In: Annual LTSN-ICS Conference. Proceedings**...** Leeds: Loughborough University. p. 53-57.

Menezes, C. S. de.; Nobre I. A. M. (2002). “Um Ambiente Cooperativo para Apoio a Cursos de Introdução à Programação”. In: Workshop Sobre Educação em Computação-WEI, 5. Florianópolis. Anais do XXII Congresso da Sociedade Brasileira de Computação. Florianópolis: SBC/UFSC.

Noronha, R. V. *et al*. (2003). “Classificação de Aprendizes através de Mapas Auto-Organizáveis”. In: Workshop Sobre Informática na Escola-WIE, 9. Campinas. Anais do XXIII Congresso da Sociedade Brasileira de Computação. Campinas: SBC, 2003. p. 127-134.

Pimentel, E. P.; França, V. F.; Omar, N. (2003). “A identificação de grupos de aprendizes no ensino presencial utilizando técnicas de clusterização”. In: Workshop de Educação em Computação, 14. Rio de Janeiro. Anais do XIV Simpósio Brasileiro de Informática na Educação. Rio de Janeiro: SBC.

Rodrigues Júnior, M. C. (2004). “Experiências Positivas para o Ensino de Algoritmos”. In: Workshop de Educação em Computação, 4. Feira de Santana. Anais do IV Escola Regional de Computação Bahia-Sergipe. Bahia: SBC.

Silva, M. A. da. (2003). “Protótipo de uma ferramenta para auxiliar no ensino de técnicas de programação”. 42 f. Trabalho de Conclusão de Curso (Bacharelado em Informática) – Departamento de Ciências Exatas e Tecnológicas, Universidade do Planalto Catarinense, Lages.

Tobar, C. M. e*t al*. (2001). “Uma Arquitetura de Ambiente Colaborativo para o Aprendizado de Programação”. In: Simpósio brasileiro de Informática na Educação, 12. Vitória, ES. Anais... Vitória: UFES.

Vicari, R.; Giraffa, L. M. M. (2003). “Fundamentos dos sistemas tutores inteligentes”. In: Barrone, D. Sociedades Artificiais: a nova fronteira da inteligência nas máquinas. Porto Alegre: Bookman. cap. 7.