# *APP-* Uma arquitetura pedagógica para aprendizagem de programação de computadores

**Lucinéia Barbosa da Costa Chagas[1], Márcia Gonçalves de Oliveira[2], Orivaldo de Lira Tavares[1]**

[1]Universidade Federal do Espírito Santo (UFES) – Avenida *Fernando Ferrari,* 514 – Goiabeiras, Vitória – ES - Brasil

[2]Centro de Referência em Formação e EaD (Cefor/IFES) – Rua *Barão de Mauá*, 30 – Jucutuquara – Vitória – ES – Brasil

**cliklucineia@gmail.com, clickmarcia@gmail.com, tavares@inf.ufes.br**

***Abstract.*** *This article presents a pedagogical architecture developed in order to support the programming learning process. This pedagogical architecture allows each teacher create problems and lists of exercises, create pair of students based on the similarities of the solutions developed and recommend exercises based on the performance achieved by students in exercises developed previously. The student, in turn, has the possibility of developing the lists of exercises, create new solutions programmed collaboratively and solve problems recommended.*

***Resumo.*** *Este artigo apresenta uma arquitetura pedagógica para apoiar o processo de aprendizagem de programação. Nesta arquitetura pedagógica o docente tem a possibilidade de cadastrar problemas, criar listas de exercícios, formar pares de alunos com base nas similaridades das soluções desenvolvidas e recomendar exercícios com base no desempenho obtido pelos alunos em exercícios desenvolvidos anteriormente. O aluno, por sua vez, tem a possibilidade de desenvolver as listas de exercícios, criar novas soluções programadas de forma colaborativa e resolver os problemas recomendados.*

## 1. Introdução

A disciplina relacionada à programação de computadores é uma das principais razões pelas quais há evasão e reprovação nas primeiras fases dos cursos de informática da maioria das escolas de ensino técnico ou superior [Tavares, 2013; Chagas, 2013].

Vários fatores contribuem para a dificuldade de aprendizagem de programação, como: a inexperiência dos alunos egressos da educação básica quanto à resolução de problemas [Kazimoglu, 2012; Ater-kranov, 2010], o uso de habilidades cognitivas e raciocínio lógico [Casey, Cegielski & Diane, Hal, 2006; Qualls & Sherrell, 2010; Wing, 2006].

[Moons, 2013] relata que do ponto de vista de muitos estudantes, os cursos introdutórios de programação são considerados difíceis, no que resulta em baixos desempenhos e muitas reprovações.

Um dos fatores que contribui para a dificuldade de aprendizagem de programação está relacionado às múltiplas habilidades e múltiplos processos necessários para desenvolver um programa [Govender, 2009], além da dificuldade compreensão do problema a ser resolvido [Assis, 2008].

Para [Fessakis *et alli*, 2013] o aprendizado de programação é benéfico no desenvolvimento da cognição, além de ser útil em diversas áreas de conhecimento, por desenvolver habilidades lógicas de planejamento, definição, uso de critérios de comparação de teste e avaliação de soluções.

Na busca de propor melhorias no processo de ensino e de aprendizagem de programação, apresenta-se neste trabalho uma arquitetura pedagógica especialmente desenvolvida para auxiliar professores e alunos no processo de ensino e de aprendizagem de programação de computadores.

Este trabalho está organizado conforme a ordem a seguir. Na seção 2, são apresentados os problemas de aprendizagem de programação de computadores. A seção 3 descreve a metodologia aplicada. Na seção 4 é apresentada a *APP - Arquitetura Pedagógica para Aprendizagem de Programação de Computadores*. A seção 5 apresenta aplicações da arquitetura pedagógica e, finalmente, são apresentadas as considerações finais.

## 2. O complexo processo de aprendizagem de programação

A aprendizagem de programação é estudada por vários pesquisadores em todo o mundo. Todavia, mesmo com vários estudos feitos, a aprendizagem de programação continua a ser um grande desafio [Tavares, 2013; Costa, 2011, Chagas, 2013].

Outras habilidades importantes para o programador, a serem desenvolvidas pelos aprendizes são a resolução de problemas de forma colaborativa [Cámara, 2013]; o desenvolvimento de várias soluções para o mesmo problema [Tavares, 2013]; trabalho em equipe e espírito crítico [Wang & Lin, 2007]; codificação de algoritmos em linguagens de programação [Lau, 2011; Khvilon & Patru , 2002; Cohen & Haberman, 2007] dentre outras.

O professor como ator do processo de ensino e de aprendizagem, precisa estar ciente de seu papel e é necessário que ele consiga perceber o que o aluno não consegue assimilar. O professor precisa refletir sobre sua prática, criticar e ajustar constantemente seu método pedagógico para que possa atingir o seu objetivo maior: criar condições favoráveis à aprendizagem do aluno [Costa, 2013].

Caminhando nessa direção, apresenta-se uma arquitetura pedagógica para aprendizagem de programação de computadores que tem o objetivo de promover melhorias no processo de ensino e de aprendizagem.

Com essa arquitetura pedagógica o professor avalia os níveis de aprendizagem dos alunos de forma contínua, acompanhando e incentivando o desenvolvimento das

habilidades por meio de estratégias de avaliações diagnósticas e formativas e da proposição de atividades que promovam a autoria, colaboração e o espírito crítico.

## 3. Metodologia

Para o desenvolvimento da proposta de solução, fez-se uma pesquisa qualitativa e quantitativa, com o método de coleta de dados, pesquisa bibliográfica e de campo.

A pesquisa bibliográfica fundamentou-se principalmente na busca de soluções similares já desenvolvidas para minimizar os problemas de aprendizagem de programação e ferramentas para o desenvolvimento. Também foi aplicado como instrumento de pesquisa, um questionário constituído por dez questões objetivas.

O principal objetivo das questões foi compreender quais os problemas de aprendizagem enfrentados pelos alunos na disciplina de programação de computadores.

Foi observado que a maioria dos alunos tinha dificuldades na compreensão dos problemas propostos, além das dificuldades de uso da linguagem de programação. Esse questionário foi aplicado a uma turma de trinta alunos de uma instituição de ensino superior com faixa etária entre 17 e 34 anos. Embora o tamanho da amostra seja pequeno, ela foi útil para a avaliação inicial dos resultados da pesquisa.

Para o desenvolvimento da proposta de solução, foi usado o *software Xampp* 3.2.1 que é um servidor independente de plataforma, que consiste de uma base de dados *MySQL*, servidor *web Apache* e os interpretadores para linguagens de *script: PHP e Perl.* O programa está liberado sob a licença *GNU* e atua como um servidor *web* livre, fácil de usar e capaz de interpretar páginas dinâmicas.

Os seguintes recursos adicionais foram usados:

*Moodle* - ambiente usado para instalar a ferramenta computacional desenvolvida e para os discentes realizarem as atividades.

*R* - é um *software* usado para calcular o grau de similaridade dos códigos desenvolvidos pelos alunos. Com os cálculos gerados foi possível formar os pares de alunos para o desenvolvimento de atividades da programação em pares.

*Sed* –ferramenta computacional usada com objetivo de remover as *stop words* dos códigos criados pelos discentes. Com o *Sed* foi possível retirar dos códigos as palavras frequentes que não representam nenhuma informação de maior relevância para a identificação de palavras-chave.

O ambiente de aprendizagem de programação possui integrado ao sistema o compilador da linguagem *Haskell*, o que dá condições aos alunos de testarem suas soluções no próprio ambiente, programar em pares e receber recomendações de atividades de programação baseadas nos resultados obtidos nos exercícios anteriores.

## 4. *APP* - Arquitetura Pedagógica para Aprendizagem de Programação

O *AAP* é um ambiente de aprendizagem de programação desenvolvido com o propósito de auxiliar professores e alunos no processo de ensino e de aprendizagem de programação de computadores.

Com esse ambiente o aluno aprende enquanto desenvolve suas atividades de forma individual ou colaborativa, com a opção de compartilhar suas soluções e dúvidas sobre determinados problemas e propor diversas soluções para cada problema.

O procedimento computacional foi desenvolvido em *PHP* e segue a estrutura apresentada na Figura 01. O protótipo computacional possui duas visões sendo uma do professor e outra do aluno. A Figura 02 apresenta a visão do Professor.

Na visão do professor, mostrada na Figura 02, o docente tem a possibilidade de cadastrar problemas, criar uma lista de exercícios com os problemas cadastrados, acompanhar o desenvolvimento das atividades dos alunos, formar pares de alunos para desenvolvimento de atividades e recomendar atividades.

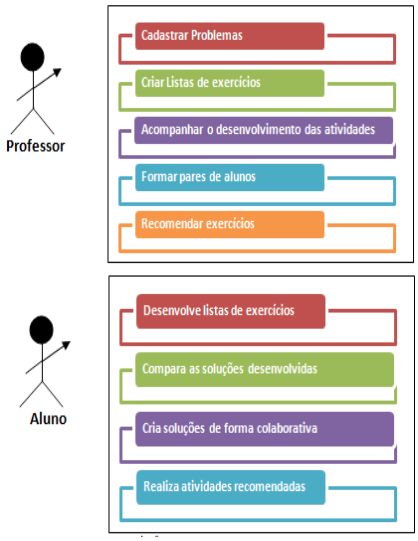


**Figura 01 – Estrutura computacional do *APP***


**Figura 02 – Visão do professor**

O procedimento computacional apresentado na Figura 01, inclui funções como: *Cadastrar problemas* - nesta função atribuída ao professor é feito o cadastramento de todos os problemas a serem propostos aos alunos.

Na segunda função - *criar listas de exercícios*, o docente tem a possibilidade de criar listas de exercícios, inserir nas listas os problemas cadastrados, estabelecer prazos para a entrega das atividades, informar se as atividades serão desenvolvidas de forma individual ou em grupos.

Na terceira função - *acompanhamento de atividades*, o professor tem a possibilidade de acompanhar as atividades desenvolvidas pelos alunos. Neste acompanhamento o docente visualiza a compreensão do problema por parte do estudante, a especificação da solução e o código da solução desenvolvido.

Na função *Programação em Pares,* o professor usa um recurso digital especialmente desenvolvido para analisar as soluções dos alunos, no intuito de buscar similaridades ou diferenças entre as soluções desenvolvidas para a formação dos pares. A Figura 03 apresenta o fluxo das etapas da Programação em Pares.

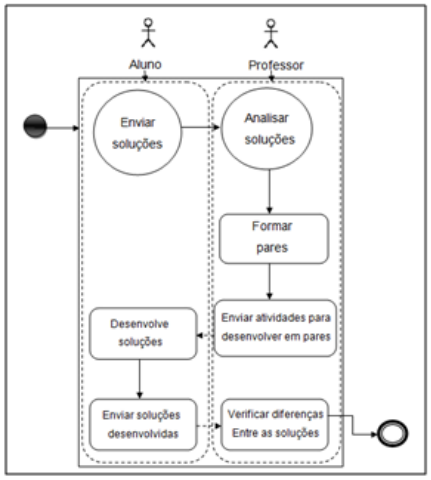


**Figura 03: Fluxo das etapas da formação de pares**

Após a formação dos pares é enviado aos alunos às mesmas atividades para que possam desenvolver em pares, novas soluções para o problema proposto. As novas soluções devem possuir diferenças em relação àquelas desenvolvidas anteriormente de forma individual. A análise das soluções é feita através do modelo de espaço vetorial.

O modelo de espaço vetorial representa cada documento como um vetor de termos. Cada termo possui um valor associado que indica seu grau de importância [Bonfin, 2009]. Os termos são ocorrências únicas nos documentos, neles são atribuídos pesos que especificam o tamanho e a direção de seu vetor de representação.

Os pesos são usados para calcular o grau de similaridade entre consulta e documento. Cada elemento do vetor de termos é considerado uma coordenada dimensional. Assim, os textos podem ser colocados em um espaço euclidiano de *n* dimensões e a posição do texto em cada dimensão é dada pelo seu peso [Paysan, 2006]. A Figura 04 mostra o espaço euclidiano com textos em *n* dimensões.

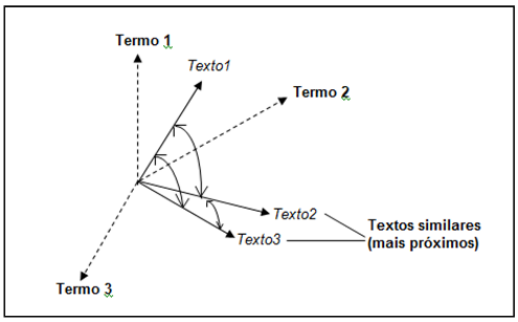


**Figura 04. Espaço euclidiano com os textos em *n* dimensões**

Os textos que possuem os mesmos termos são colocados em uma mesma região do espaço, pois se tratam de assuntos similares. Para o cálculo dos pesos dos vetores é usada a fórmula do *TF-IDF* (*term frequency–inverse document frequency*). O *TF-IDF* é uma estatística numérica que reflete o quão importante uma palavra é, considerando um documento dentro de uma coleção de documentos [Baeza & Ribeiro, 2011].

O valor *TF-IDF* aumenta proporcionalmente com o número de vezes que uma

palavra aparece no documento, mas é compensada pela frequência da palavra na coleção de documentos, o que ajuda a controlar o fato de que algumas palavras são geralmente mais comuns do que outras. Abaixo é apresentado a equação usada para calcular o *TF-IDF*(*term frequency–inverse document frequency*).

$$tf = \frac{freq_{ti,dj}}{max\ freq_{dj}} \quad \textbf{(1)}$$

Nesta equação, $freq_{ti,dj}$ indica a frequência do termo $t_i$ no documento $d_j$ e $max\ freq_{dj}$ indica a frequência do termo de maior frequência no documento $d_j$. Quanto mais frequentemente um termo ocorrer no texto de um documento, maior será a sua frequência. Para calcular o termo *IDF*, foi usada a seguinte equação:

$$idf = log\frac{N}{n_{ti}} \quad \textbf{(2)}$$

O número total de documentos da coleção é dado por N e $n_{ti}$ é o número de documentos da coleção que contêm o termo de indexação $t_i$. Após calculados os pesos dos termos, é feito o cálculo da medida de similaridade entre documentos e consulta. Para este cálculo, fez-se uso da fórmula do *cosseno*, apresentada a seguir:

$$similaridade(Q,D) = \frac{\sum_{k=1} w_{qk} \bullet w_{dk}}{\sqrt{\sum_{k=1}^{n}(w_{qk})^2 \bullet \sum_{k=1}^{n}(w_{dk})^2}} \quad \textbf{(3)}$$

O vetor de termos da consulta é representado por Q, *D* é o vetor de termos do documento, $W_{qk}$ são os pesos dos termos da consulta e $W_{dk}$ são os pesos dos termos do texto. O modelo de espaço vetorial foi escolhido devido a sua capacidade de mensurar um grau de similaridade parcial entre os documentos e pela não necessidade de técnicas probabilísticas avançadas para a recuperação de informação.

As Figuras 05 e 06 mostram as tabelas de similaridades com os pares formados. A Figura 05 apresenta os pares de alunos formados com os maiores graus de similaridade encontrados nos códigos das atividades desenvolvidas. A Figura 06 apresenta um exemplo de pares formados considerando soluções menos parecidas, em uma amostra de cinco alunos que desenvolveram códigos de uma mesma atividade.

| Aluno | A | B | C | D | E |
|---|---|---|---|---|---|
| A | 1 | 0.9905 | 0.9137 | 0.4986 | Par com "A" |
| B | 0.9905 | 1 | Par com "B" | 0.5034 | 0.9816 |
| C | 0.9137 | Par com "B" | 1 | 0.4650 | 0.9055 |
| D | 0.4986 | 0.5034 | 0.4650 | 1 | 0.4942 |
| E | Par com "A" | 0.9816 | 0.9055 | 0.4942 | 1 |

Pares formados: Aluno "A" com aluno "E"
Aluno "B" com aluno "C"
Aluno "D" Ficou sem par

**Figura 05 – Pares formados por maior grau de similaridade**

| Aluno | A | B | C | D | E |
|---|---|---|---|---|---|
| A | 1 | 0.9905 | 0.9137 | Par com "A" | 0.9910 |
| B | 0.9905 | 1 | Par com "B" | 0.5034 | 0.9816 |
| C | 0.9137 | 0.9224 | 1 | 0.4650 | 0.9055 |
| D | Par com "A" | Par com "B" | 0.4650 | 1 | 0.4942 |
| E | 0.9910 | 0.9816 | 0.9055 | 0.4942 | 1 |

Pares formados: Aluno "A" com aluno "D"
Aluno "B" com aluno "C"
Aluno "E" Ficou sem par

**Figura 06 – Pares formados por menor grau de similaridade**

Na Figura 05, a técnica usada neste experimento faz a leitura dos códigos e os

agrupa em pares de alunos, conforme a similaridade dos códigos desenvolvidos, assim que um par é formado, este é separado e novamente o sistema busca a formação de outros pares. O processo é repetido até que se tenha formado todos os pares.

A Figura 06 apresenta os pares formados através das atividades desenvolvidas com menor grau de similaridade. A técnica usada neste experimento faz a leitura dos códigos e os agrupa em pares de alunos conforme a menor similaridade possível dos códigos desenvolvidos. Assim que um par é formado, ele é separado e novamente o sistema busca a formação de outros pares. Este processo também é repetido até que se tenha formado todos os pares.

A funcionalidade *Recomendar atividades* tem como objetivo a aprendizagem de programação, por meio de recomendações de exercícios adequados ao perfil de aprendizagem do estudante. Além disso, armazena o histórico de aprendizagem do aluno para permitir o acompanhamento de sua aprendizagem pelo professor.

A recomendação de exercícios é feita de forma automática, em que o próprio ambiente imita o docente no processo de recomendar exercícios para os alunos com base no seu perfil de aprendizagem. A abordagem pedagógica proposta na *Recomendação de Exercícios* teve como base:

- ✓ resolução de problemas de modo individual;
- ✓ recomendação de problemas, parcialmente ou totalmente resolvidos, equivalentes aqueles anteriormente recomendados e não resolvidos;
- ✓ autorregulação – o próprio estudante vê imediatamente o resultado da avaliação da sua solução;
- ✓ espírito crítico – o estudante acessa as possíveis soluções elaboradas por seus colegas, no intuito de facilitar sua tomada de consciência sobre o processo de resolução de problemas.

Na *Visão do Aluno*, a arquitetura pedagógica proposta trás as seguintes possibilidades:

- ✓ *escolher a lista de exercícios que deseja resolver* – o discente tem a possibilidade de escolher a lista de exercícios que quiser resolver, dentre as opções disponibilizadas pelo professor;
- ✓ *desenvolver os problemas propostos em cada lista de exercícios* – após escolher o problema a ser solucionado, o aluno é direcionado a uma tela no qual consta o enunciado do problema, o local para digitar o código da solução e o interpretador da linguagem. Em seguida, o estudante usa o plano d*e teste para testar o código e apresentar os resultados obtido*s no processamento.

O plano de teste é um recurso do ambiente em que é editado pelo usuário os dados de entrada e a respectiva saída esperada. O ambiente ao processar o código Haskell desenvolvido como solução, apresenta a saída obtida para cada dado de entrada.

Caso a saída obtida seja igual à saída esperada pelo usuário, o ambiente informa que o resultado está correto; caso contrário o ambiente informa que as respostas não são iguais ou apresenta o erro encontrado na execução do código do aluno;

- ✓ *criar uma solução colaborativa para um determinado*

*problema proposto* – a criação da solução colaborativa para um determinado problema proposto acontece na Programação em Pares, em que o discente tem a possibilidade de criar uma nova solução para o mesmo problema desenvolvido anteriormente, desta vez de forma colaborativa. Ao desenvolver uma atividade em par, o estudante é remetido a um menu de opções, conforme mostra a Figura 07.

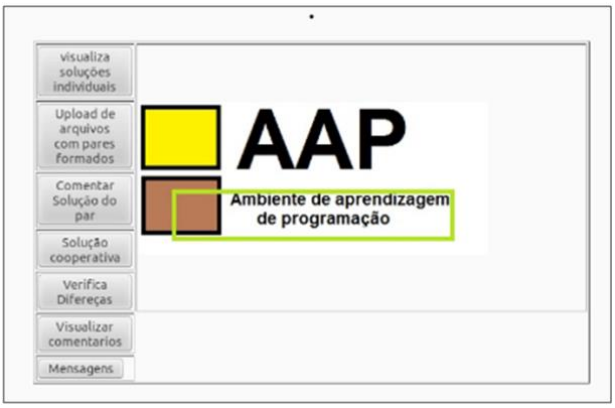


**Figura 07 – Programação em Pares**

A Figura 07 apresenta opções como: visualizar as soluções desenvolvidas de forma individual, fazer *upload* de arquivos com pares formados, comentar a solução do par formado, desenvolver uma solução de forma cooperativa, verificar as diferenças entre as soluções desenvolvidas, visualizar os comentários das soluções e receber alertas de novas mensagens são outras opções apresentadas no menu.

Ao desenvolver a solução de forma colaborativa, os pares formados pelo sistema visualizam as soluções desenvolvidas de forma individual e são convidados a criar uma nova solução para o mesmo problema proposto. Essa solução a ser criada deve ter no mínimo 20% de diferença das soluções desenvolvidas anteriormente.

Após o desenvolvimento da solução colaborativa, os próprios alunos através do ambiente de aprendizagem faz a comparação das soluções desenvolvidas para analisar os percentuais de diferença entre elas.

O processo de desenvolver mais de uma solução para o mesmo problema, faz com que o aluno evolua no seu processo de aprendizagem, aprendendo novas formas de resolver um problema.

## 5. Aplicação da Arquitetura Pedagógica

A *APP*- Arquitetura Pedagógica para aprendizagem de programação de computadores está sendo inicialmente aplicada por uma turma de programação introdutória do curso de Ciência da Computação da Universidade Federal do Espírito Santo.

Usando essa AP os alunos resolvem exercícios de programação em Linguagem *Haskell,* de forma individual ou colaborativa, além de receber atividades recomendadas com base nos códigos desenvolvidos anteriormente.

Esses exercícios são executados pelo ambiente de aprendizagem e as saídas são apresentadas ao professor e aos alunos. A *APP* mapeia as variáveis de avaliação que interessam ao professor para ter um diagnóstico da turma e fornecer *feedbacks* aos alunos. Até o final de 2016, existirão novos resultados da aplicação da Arquitetura pedagógica com outra turma de programação.

## 6. Considerações finais

Os objetivos da arquitetura pedagógica apresentada neste artigo visam estender as práticas do professor na promoção da aprendizagem de programação por seus alunos.

Essas práticas poderão ser aperfeiçoadas pelo fato de que as estratégias implementadas possibilitam o controle contínuo do desenvolvimento de habilidades de programação nos aprendizes.

Foi apresentado o processo de desenvolvimento de todos os recursos digitais usados, de modo a permitir a continuidade deste trabalho.

Na questão da viabilidade da arquitetura pedagógica, a implementação de um ambiente computacional que dê suporte a essa arquitetura foi determinante. Através dele foi possível avaliar os pontos fortes e os fracos, bem como oportunidades de melhoria.

Em geral, foram encontradas respostas para as principais questões norteadoras desta pesquisa, mostrando arquiteturas pedagógicas úteis para a aprendizagem de programação de computadores. Como trabalhos futuros destacam-se:

- ✓ a inserção de novos recursos computacionais pedagógicos na arquitetura pedagógica apresentada;
- ✓ inclusão um agente pedagógico que auxilie o estudante no desenvolvimento de suas atividades;
- ✓ inserção de recursos computacionais para o desenvolvimento de atividades de programação por pessoas com deficiência visual ou auditiva.

Espera-se que este trabalho venha promover transformações na aprendizagem de programação como: análise dos problemas propostos e reflexão sobre possíveis soluções, desenvolvimentos de soluções de forma colaborativa, autorreflexão do aprendizado, além do interesse em aprendizagem de outras linguagens de programação.

## 7. Referências

Ater-Kranov, et al., Developing a community definition and teaching modules for computational thinking: accomplishments and Paper presented at the Proceedings of the 2010 ACM conference on Information technology education, pp. 143-148, 2010.

Assis, João Francisco de. Evasão escolar no ensino profissionalizante: Um estudo de Caso do Colégio Francisco Carneiro Martins. UNICENTRO: Revista Eletrônica de pós-graduação Lato Sensu, 2008.

Bonfin, M. E. Recuperação de documentos texto usando modelos probabilísticos estendidos. In: VI - EPCC Encontro Internacional de Produção Científica Cesumar 27 a 30 de outubro de 2009.

Cabral, M. I. C. et al. Perfil dos cursos de computação e informática no Brasil, XXVII Congresso da SBC - XV WEI, Rio de Janeiro, 2007.

Cámara, L. M. S.; Velasco, M. P.; Alcover, C. M.; Iturbide, J. A. V.: An evaluation of students' motivation in computer-supported collaborative learning of programming concepts. In: Computers in Human Behavior, 2013.

Casey, G. Cegielski and Dianne J. Hal(2006): What Makes a good programmer?: In Communications of the ACM October 2006/Vol. 49, No. 10.

Chagas, L.B.C; Oliveira, M.G.: Metodologia ANEA para avaliação Online de Lógica de Programação. In: Anais do XXII SBIE - XVII WIE. Aracaju, 2011.

Costa, L. B. ; Tavares, O. L. ; Menezes, C. S. ; Aragon, R. . Pedagogical architectures and web resources in the teachinglearning of programming. In: TISE 2013 - XVIII Conferencia Internacional sobre Informática na Educação, 2013, Porto Alegre - RS. Nuevas Ideas en Informática Educativa, 2013. v. 9. p. 430-434.

Cohen, A., & Haberman, B. (2007). Computer science: a language of technology. ACM SIGCSE Bulletin, 39(4), 65–69.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. Computers & Education, 63, 87-97. doi: 10.1016/j.compedu.2012.11.016

Govender, I. (2009). The learning context: Influence on learning to program.Computers & Education, 53(4), 1218-1230.

Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2012). Learning Programming at the Computational Thinking Level via Digital GamePlay.Procedia Computer Science, 9, 522-531.

Khvilon, E., & Patru, M. (2002). Information and communication technology in education: A curriculum for schools and programme of teacher development. http://unesdoc.unesco.org/images/0012/001295/129538e.pdf. Acessado em 17 de setembro de 2015.

Lau, W. W.F. , Yuen, A. H.K.. Modelling programming performance: Beyond the influence of learner characteristics. In: Computers & Education, 2011.

Moons, J.; Backer, C. D.: The design and pilot evaluation of an interactive learning environment for introductory programming influenced by cognitive load theory and constructivism. In: Computers & education 60 (2013) 368-384.

Paysan, T. G. Sistema WEB Gerenciador de Perfis. In: Monografia de conclusão de curso. Ufes-2006.

R. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley, Wokingham, UK, 1999. Second edition published in 2011.

Qualls, J. A., & Sherrell, L. B. (2010). Why computational thinking should be integrated into the curriculum. J. Comput. Small Coll., 25(5), 66-71.

Tavares, O. L.; Menezes, C.S.; Aragón, R.; Costa, L. B.: Uma arquitetura pedagógica auxiliada por tecnologias para ensino e aprendizagem de programação. In XXXIII Congresso as sociedade brasileira de computação - CSBC 3013.

Wang, S.-L., & Lin, S. S. J. (2007). The effects of group composition of self-efficacy and collective efficacy on computer-supported collaborative learning. Computers in Human Behavior, 23, 2256–2268.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(2), 33-35.