Concepção de Ensino-Aprendizagem de Algoritmos e Programação de Computadores: A Prática Docente

Andreia de Jesus[1], Gláucia Silva Brito[2]

[1]UNIBRASIL – Faculdades Integradas do Brasil
Coordenação de Sistemas de Informação
Rua Konrad Adenauer, 442. Tarumã.
CEP 82820-540 Curitiba, PR

sistemas@unibrasil.com.br

[2]UFPR - Universidade Federal do Paraná
Departamento de Comunicação Social
Rua Bom Jesus, 610. Juvevê.
CEP 80035-010 Curitiba, PR

glaucia@ufpr.br

Resumo *O objetivo deste artigo é apresentar uma visão geral sobre a concepção de ensino-aprendizagem de algoritmos e programação de computadores, e pontuar as principais dificuldades enfrentadas pelos alunos e apresentar o perfil dos professores, tendo como base literaturas relacionadas com o assunto. Após a conclusão da análise destes itens, propõe-se uma metodologia de plano de aula que visa auxiliar o professor no processo de ensino-aprendizagem de algoritmos e programação de computadores.*

1. Introdução

Disciplinas como algoritmo e programação de computadores exigem dos alunos habilidades e competências como: raciocínio lógico, resolução de problemas e a capacidade de abstração da solução em uma representação formal e/ou em uma linguagem computacional.

Autores como Raabe e Silva [1] citam a abstração como um dos principais fatores da desmotivação e do elevado grau de desistência dos alunos. Além disso, essas habilidades e competências são pré-requisitos à maioria das disciplinas trabalhadas em cursos de Computação e Informática. Portanto, é primordial que sejam desenvolvidos aplicativos e pesquisadas metodologias que venham, efetivamente, contribuir para a melhoria da qualidade do processo de ensino-aprendizagem destes alunos.

Em geral, os professores utilizam exemplos hipotéticos para demonstrar o comportamento da máquina (computador) no momento da execução das soluções abstraídas em linguagem computacional. Mas isto não é suficiente, pois os alunos não

conseguem visualizar efetivamente como as mudanças de estado ocorrem no equipamento. Por isso, pesquisas estão sendo realizadas e ferramentas computacionais estão sendo desenvolvidas com este objetivo ([1], [2], [3], [3] entre outros).Vale ressaltar que tanto as metodologias, como os aplicativos educacionais devem ser desenvolvidos de tal forma que durante a sua utilização os alunos se sintam motivados e consigam ter nessas ferramentas um auxílio a mais no desenvolvimento das competências necessárias à construção de algoritmos e programação de computadores.

Portanto, é de suma importância que os professores que lecionam algoritmos e programação de computadores tenham uma definição bem clara da concepção do processo de ensino-aprendizado desses conteúdos, para que os mesmos possam desenvolver metodologias que realmente contribuam para o desenvolvimento das competências e habilidades exigidas, além de fazerem uso adequado de aplicativos destinados a este fim.

2 Contexto do Processo Ensino-Aprendizagem de Algoritmos e Programação de Computadores

2.1. Natureza das Disciplinas de Algoritmos e Programação de Computadores

Para se compreender a natureza das disciplinas de algoritmos e programação de computadores é preciso, primeiramente, conceituar três termos bastante utilizados nestas disciplinas: lógica, algoritmos e linguagem de programação.

Para Forbellone [4] (2000, pág. 1) a lógica é a “arte do bem pensar”. Segundo o autor o raciocínio é a forma mais complexa do pensamento e a lógica estuda a correção do raciocínio, colocando coerência e ordem no pensamento. O autor também define o que é lógica de programação, descrevendo que além de visar à ordem da razão, também se preocupa com a simbolização formal do raciocínio para a programação dos computadores. Já o conceito de algoritmos é definido, em geral por vários autores [3][4][6], como uma seqüência lógica de passos que visa atingir um objetivo bem definido. Por fim, as linguagens de programação constituem ferramentas para a implementação de software [5].

Com as definições apresentadas anteriormente é possível fazer a seguinte relação: para se construir um algoritmo é preciso primeiramente desenvolver o raciocínio lógico, pois esta habilidade é fundamental nesta tarefa. Além disso, é preciso compreender o funcionamento das linguagens de programação, quanto à sintaxe e à semântica, para que seja possível traduzir uma solução algorítmica para um programa de computador. Logo, a construção de algoritmos é uma das várias etapas da atividade de programação.

Delgado et al.[6], em seu trabalho de pesquisa, identificou subcompetências que devem ser desenvolvidas antes de atingir a competência de construção de algoritmos e, consequentemente, programação de computadores: **(1)** Interpretação e compreensão de texto; **(2)** Resolução de problemas; **(3)** Formalização da solução proposta; **(4)** Construção de algoritmos.

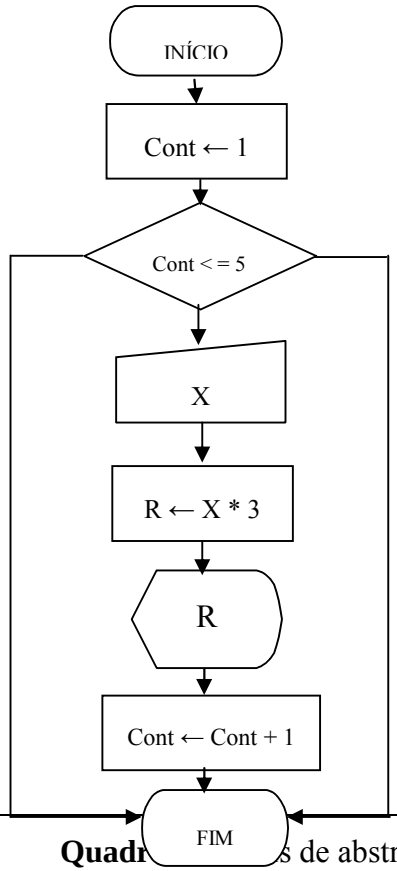
Outra característica importante dos algoritmos é que não existe uma única solução para um determinado problema. Como é necessário aplicar lógica na construção de um algoritmo e/ou programa, a solução é bastante subjetiva, pois o raciocínio lógico é particular de cada pessoa. Logo, para um determinado problema é possível apresentar várias soluções com caminhos diferentes, mas todas alcançarem o mesmo resultado. Devido a isto, é necessário saber formalizar as soluções propostas de forma a abstraí-las em diversos níveis, como demonstrado no Quadro 1, e permitir assim uma compreensão universal da solução. Vários autores ([3], [4], [6], entre outros) mencionam as diversas abstrações isoladamente. Portanto, o Quadro 1 tem como objetivo apresentar, paralelamente, as diversas abstrações para um determinado problema e, com isso, mostrar que quanto mais alto o nível de abstração, mais a solução se aproxima da linguagem computacional. Além disso, o quadro possibilita reforçar que a formalização, independente do nível, se preocupa com o processo de solução e não mais com a solução em si.

Portanto, com o Quadro 1 é possível verificar as diversas formas de representação de um algoritmo. A abstração 1 apresenta um algoritmo descrito em português coloquial. Esta forma de representação é mais natural, porém não possibilita uma interpretação única da solução. No caso de não se ter acesso ao enunciado do problema, quando é dito para repetir os passos 3, 4 e 5 enquanto o contador não for cinco, isto significa que o processo será executado no máximo cinco vezes ou os passos serão executados um número indeterminado de vezes até que o contador venha conter o valor 5? Já a abstração 2 trás o algoritmo representado em sua forma gráfica, os chamados fluxogramas, onde cada forma geométrica representa uma ação diferente. Estas formas gráficas são mais puras, pois substituem um grande número de palavras por convenções de desenhos que permitem visualizar mais claramente o fluxo do processo. Com o fluxograma tem-se a garantia de que os passos 3, 4 e 5 serão executados no máximo 5 vezes. Para isto basta seguir o fluxo do processo no desenho gráfico para chegar a esta conclusão. A abstração 3 mostra um algoritmo em portugol. Esta forma permite representar mais fielmente o raciocínio envolvido na lógica de programação, sem para isso ter que se preocupar com os detalhes computacionais. Ou seja, o objetivo aqui é se preocupar com o processo de solução – a construção do algoritmo. Por fim, a abstração 4 trás o algoritmo traduzido para uma linguagem de programação (neste exemplo, a linguagem C). Nesta forma de representação os detalhes computacionais estão intrínsecos a solução do problema. Neste momento é necessário ter conhecimentos sobre a sintaxe e a semântica da linguagem de programação utilizada para poder compreender e interpretar a solução proposta. Portanto, um algoritmo pode ser traduzido para qualquer linguagem de programação e ser executado em um computador. Esse processo é denominado codificação da solução.

Como visto acima a tarefa de construção de algoritmos é bastante complexa e é necessário que diversas atividades sejam desenvolvidas para que o aluno seja capaz de compreender soluções algorítmicas prontas, propor as suas próprias soluções e realizar relações entre diferentes soluções para um mesmo problema.

Enunciado do problema: Desenvolver um programa que lê um valor para a variável X, multiplica este valor por 3, armazena o resultado em uma variável R e imprime o conteúdo desta variável. O programa deve repetir 5 vezes esta sequência de comandos.

Abstração 1 Linguagem Natural	Abstração 2 Representação Gráfica	Abstração 3 Linguagem Algorítmica	Abstração 4 Linguagem de Programação C

1. Criar uma variável para ser o contador iniciando em 1;
2. Enquanto o valor do contador não for 5, executar os passos 3, 4 e 5;
3. Ler um valor para a variável X;
4. Efetuar a multiplicação do valor de X por 3, armazenando o resultado em R;
5. Apresentar o conteúdo de R;
6. Acrescentar mais um valor na variável contador;
7. Quando o contador for maior do que 5, encerrar o processamento repetição (looping).
		Algoritmo Looping
		Var:
		X, R, I: Inteiro;
		Início
		I ← 1;
		Enquanto (I <= 5)
		Leia X;
		R ← X * 3;
		Escreva R;
		I ← I + 1;
		Fim-Enquanto;
		Fim
			include<studio.h>
			#include<conio.h>
			
			void main( )
			{
			int X, R, I=1;
			
			clrscr( );
			while(I <= 5)
			{
			printf(“Digite um valor inteiro:”);
			scanf(“%d”, &X);
			R = X * 3;
			printf(“%d”, R);
			I++;
			}
			}

Quadr s de abstração de um algoritmo

2.2. Perfil dos Professores

Os professores precisam ter em mente que, para se alcançar o sucesso no ensino de algoritmos e programação de computadores, a competência de interpretar e de analisar problemas é essencial para que os alunos consigam projetar suas próprias soluções. Além disso, os algoritmos não são modelos prontos e que devem ser estudados e decorados. Um algoritmo é a descrição de um caminho para se alcançar a solução de um determinado problema, e este caminho pode ser descrito de diversas maneiras de acordo com o raciocínio lógico de cada indivíduo.

Mas estudos como o de Raabe e Silva [1] coloca que os professores não estão preparados para ensinar os alunos a resolver problemas e que o professor tem dificuldades de compreender a lógica do aluno. Em geral, a teoria é apresentada através de modelos prontos, ou seja, através de técnicas, estratégias e soluções do próprio professor. Isto leva o aluno, ao se deparar com um problema a ser solucionado e uma solução a ser formalizada, a reproduzir o que foi apresentado em sala. Neste momento esse aluno tem a percepção de que ele entendeu a solução apresentada pelo professor, mas que não compreendeu como esse processo se desenvolveu. Isto dificulta a sua habilidade de desenvolver as suas próprias soluções, pois o que lhe foi passado foi um modelo de algoritmo pronto e não o processo de desenvolvimento deste modelo.

Tendo em vista o que foi descrito, percebe-se que nas disciplinas de Algoritmos e Programação de Computadores há uma forte ênfase no ensino da construção de algoritmos e programas computacionais, mas que as atividades de interpretação e compreensão de texto, resolução de problemas e formalização da solução proposta, que são habilidades essenciais para a compreensão algorítmica, estão sendo deixadas de lado ou em segundo plano neste processo de ensino.

Portanto, as propostas de metodologias e de softwares para o ensino de algoritmos e programação de computadores, encontrados na literatura, só terão uma aplicação efetiva no processo ensino-aprendizagem desses conteúdos se os professores: (1) levarem em consideração as diferentes formas de raciocínio e interpretações que se pode ter sobre um determinado problema; (2) incentivarem os alunos a desenvolver as suas próprias soluções, ao invés de tentar fazer com que eles absorvam as soluções propostas pelo professor; (3) considerarem as atividades de interpretação de texto, resolução de problemas e formalização como etapas primordiais para a compreensão do raciocínio algorítmico, além de reforçar para os alunos que estas atividades são etapas iniciais do processo de construção de algoritmos.

Logo é de suma importância que os professores tenham bem claro: (1) a concepção de ensino-aprendizagem de algoritmos e (2) as subcompetências necessárias para a construção de algoritmos. Desta forma, os professores poderão propor metodologias compatíveis com as necessidades da área e fazer uso de softwares destinados ao ensino de algoritmos e programação de forma adequada, além de auxiliar na avaliação dessas ferramentas. Isso também possibilita uma re-análise das metodologias tradicionais de

ensino-aprendizagem de algoritmos e programação, evitando que estas metodologias sejam repassadas no desenvolvimento e no uso desses aplicativos.

2.3. Perfil dos Alunos

Em geral os alunos não sabem por onde começar quando um problema lhe é apresentado e uma solução para esse problema lhe é solicitada. Isto se deve, principalmente, pelo fato de que os alunos apresentam dificuldades na interpretação do enunciado do problema. Eles não conseguem identificar no texto quais são as variáveis de entrada, o que precisa ser processado e quais são as variáveis de saída.

Outras duas características da maioria dos alunos que ingressam nos cursos da área de computação, conforme descreve Raabe e Silva [1] são: (1) a falta de perfil para a solução de problemas e a (2) base operatório-formal fraca (o raciocínio operatório formal é base para a compreensão do raciocínio lógico). Isto acontece porque muitos alunos não desenvolveram adequadamente as estratégias de resolução de problemas durante o ensino médio, e por isso apresentam dificuldades para desenvolver soluções algorítmicas para os problemas. Quanto à base operatório-formal os autores colocam que faltam dados empíricos que comprovem que esta habilidade não foi desenvolvida adequadamente no ensino médio.

Além disso, Lister e Leany [7] colocam em seu trabalho que os alunos mais preparados não são desafiados pelo professor, enquanto que com os alunos mais fracos o professor encontra dificuldades para compreender e identificar as suas deficiências. Esta atitude contribui para o grande número de evasão dos alunos dessas disciplinas e, conseqüentemente, dos cursos da área de computação. Devido a isto, os autores colocam a necessidade de se trabalhar com uma metodologia de ensino-aprendizam que atinja todos os níveis de conhecimento e habilidades dos alunos. Para isso, eles aplicam a chamada Taxionomia de *Bloom* para propor uma metodologia de ensino-aprendizado e avaliação desses conteúdos. A Taxionomia de Bloom possui seis níveis, sendo que os níveis subseqüentes dependem das habilidades desenvolvidas nos níveis anteriores. Os níveis em ordem crescente são os seguintes: conhecimento, compreensão, aplicação, análise, síntese e avaliação. Com esta proposta os alunos são ensinados a analisar uma descrição vaga de um problema, refiná-lo, decompô-lo em partes e determinar métodos apropriados para cada parte. A solução do problema neste caso é a tarefa em nível de síntese.

A proposta descrita anteriormente reforça o estudo de [6] que apresenta as diversas habilidades que os alunos precisam desenvolver para alcançar a habilidade maior de construção de algoritmos.

E como colocado no início desta seção os alunos apresentam maiores dificuldades na análise e síntese do problema. Ou seja, não basta ter conhecimento da sintática e semântica das linguagens de programação se os alunos não conseguem compreender e propor soluções para os problemas, etapa esta que antecede a construção de algoritmos e programas computacionais. Mas, também, como coloca Neto e Cechimel [8], a simples

ignorância dos aspectos sintáticos básicos também dificulta o aprendizado e a motivação dos alunos para a programação de sistemas computacionais.

3. Proposta Metodológica de Plano de Aula para Algoritmos e Programação de Computadores

Tendo como base a concepção de ensino-aprendizagem de algoritmos e programação de computadores e as dificuldades de aprendizado dos alunos, segue uma proposta de plano de aula que visa auxiliar o professor no processo de ensino de algoritmos e programação de computadores.

Esta proposta de plano de aula tem como 1º etapa a resolução de problemas. O objetivo é desenvolver as habilidades referentes às fases 1 e 2 da metodologia proposta por [3]. Sendo que a fase 1 não trata de algoritmos e programação e sim de técnicas para a resolução de problemas. E a fase 2 busca a formalização, em linguagem natural, do método utilizado para a solução, além da prática de resolução de problemas mais complexos.

A 2ª etapa do plano de aula proposto tem como objetivo abordar a terceira fase da metodologia de [3], a construção de algoritmos. Nesta fase é preciso trabalhar os conceitos das diversas estruturas computacionais necessárias a prática de programação de computadores, bem como a sintaxe utilizada para transcrever a solução do problema em uma linguagem algorítmica ou computacional.

Nesta fase de construção de algoritmos, deverão ser trabalhados os conceitos de cada estrutura de forma isolada. Para cada conceito abordado o professor deverá definir atividades que permitam o aluno desenvolver os níveis de conhecimento, compreensão e aplicação da Taxonomia de Bloom. Além disso, com base nestas categorias selecionar aplicativos que possam ser utilizados para apoiar o processo ensino-aprendizagem em cada um dos níveis. Para esta seleção propõe-se aplicar a metodologia apresentado por Jesus [9] que integra vários tipos de classificação de softwares: tipo geral de software, ciclo de desenvolvimento, tipo de ambiente de aprendizado, o nível de aprendizado, tipo de ferramenta para o ensino de algoritmos e pela classificação dos objetivos de aprendizado.

A seguir segue um roteiro para a elaboração das aulas nesta **segunda etapa**:

Objetivo de aprendizado e suas respectivas atividades (para cada conceito abordado): **(1)** Em nível de conhecimento (verbos que refletem este nível de conhecimento: citar, nomear, listar, reproduzir, repetir e apresentar): neste nível deve ser abordada a definição do conceito apresentado, bem como a expressão sintática que expressa o conceito no algoritmo ou programa. Exemplos devem ser apresentados aos alunos. **(2)** Em nível de compreensão (verbos que refletem este nível de conhecimento: explicar, relacionar, traduzir, transformar, descrever e associar): neste nível os alunos devem ser estimulados a explicar o funcionamento lógico do conceito estudado, bem como identificar este conceito em solução de problemas já modelados na linguagem natural. **(3)** Em nível de aplicação (verbos que refletem este nível de conhecimento: aplicar, prever, demonstrar, preparar, resolver e modelar): neste nível os alunos devem

ser estimulados a resolver problemas que necessitam do conceito estudado para a sua solução; demonstrar aos alunos como a aplicação do conceito pode melhorar um algoritmo ou programa que já foi desenvolvido por eles; e como o conceito, se aplicado adequadamente a uma determinada solução, pode contribuir para que o algoritmo, quando implementado em uma linguagem de programação, exija menos recursos da máquina.

Definir aplicativos que possam ser utilizados como apoio no processo de desenvolvimento de cada nível. Neste ponto é importante que o professor realize uma análise do(s) aplicativo(s) proposto(s) para identificar, conforme metodologia proposta por [9], em qual ou quais dos níveis da Taxonomia de Bloom a ferramenta(s) poderá(ão) ser utilizada(s) como apoio no processo.

A seguir um exemplo de aplicação do roteiro proposto utilizando um conceito operacional de programação modular:

Conceito: Passagem de Parâmetro

a) **Em nível de Conhecimento**: (1) Objetivos de Aprendizado: definir o conceito de passagem de parâmetro por valor e por referência, suas respectivas sintaxes e conceitos intrínsecos a sua utilização, como variável global e local. (2) Atividades Propostas: identificar, segundo a sintaxe apresentada nos algoritmos e/ou programas, se é uma passagem de parâmetro por valor ou por referência; corrigir a sintaxe dos algoritmos e/ou programas no que diz respeito à passagem de parâmetro. Este exercício deve reforçar os conceitos de variável global e local. (3) Definir Aplicativo: Através da análise e classificação do ambiente A4, realizada por [9], foi possível identificar que este ambiente auxilia o desenvolvimento em nível de conhecimento através do tutorial e da lista de links interessantes. Além disso, o editor de algoritmos permite que os alunos corrijam a sintaxe de algoritmos pré-definidos e editem os seus próprios algoritmos, os quais podem ser submetidos ao professor para a correção da sintaxe. E por tratar de um ambiente distribuído socialmente possibilita, através do chat e do fórum, a socialização e discussão dos conteúdos em momentos extra classe.

b) **Em Nível de Compreensão:** (1) Objetivos de Aprendizado: discutir com os alunos quando e por que a passagem de parâmetro deve ser utilizada. Quais as vantagens que este conceito trás para a atividade de programação de computadores. (2) Atividades Propostas: descrever a lógica de funcionamento da passagem de parâmetro por valor e por referência. Esta descrição deve ser socializada com os demais colegas de turma para analisar as diferentes percepções; propor aos alunos que traduzam as soluções de problemas já formalizadas na linguagem natural para a linguagem algorítmica, utilizando o conceito de passagem de parâmetro; apresentar aos alunos programas que utilizam conceito de passagem de parâmetro e que apresentam erros de lógica. Os alunos devem *debugar* (executar passo a passo) estes programas para identificar o problema e propor solução para eles. (3) Definir Aplicativos: Neste nível de aprendizado duas ferramentas podem apoiar o processo, segundo [9]: o **Ambiente A4**, que além de proporcionar ao aluno a socialização e a discussão sobre a sua compreensão do conceito, possibilita o aluno continuar utilizando o tutorial e os links interessantes

para estar pesquisando novos exemplos e reforçando a sintaxe; já o simulador de algoritmos **WebPortugol** possibilita os alunos debugar os programas propostos. Esta prática proporciona uma melhor compreensão, por parte do aluno, sobre o comportamento da máquina quando um determinado conceito é aplicado e executado. Além de facilitar a compreensão de erros de lógica relacionados a um determinado conceito.

c) **Em nível de Aplicação:** (1) Objetivos de Aprendizado: discutir com os alunos quando e por que aplicar este conceito para a resolução de problemas computacionais. (2) Atividades Propostas**:** apresentar, aos alunos, diversos enunciados de problemas e solicitar que eles identifiquem quando e porque a passagem de parâmetro deve ser aplicada; solicitar que os alunos apresentem suas próprias soluções algorítmicas com passagem de parâmetro; demonstrar como a passagem de parâmetro pode melhorar a organização de algoritmos e programas já desenvolvidos por eles e, com isso, contribuir para o reaproveitamento de código. É importante que nesta atividade sejam utilizados algoritmos que os alunos já desenvolveram, pois desta forma eles irão direcionar a atenção para o conceito que está sendo abordado e não para uma solução de problemas que ainda não estudaram; modelar, junto com os alunos, situações em que o uso de parâmetros diminuirá o número de comandos (linhas) de um algoritmo e/ou programa; ou em casos em que diminuirá o número de variáveis e, com isso, exigirá menos recurso da máquina, quando implementado em uma linguagem de programação. (3) Definir Aplicativo: Para [9] o ambiente A4 poderá ser utilizado nesta fase como uma ferramenta de apoio para socializar e discutir as demonstrações e modelagens trabalhadas em sala de aula e para os alunos pesquisarem nos links interessantes outras aplicações de passagem de parâmetro. Com o simulador WebPortugol os alunos poderão implementar as suas soluções e debugá-las. Isto fará com que os alunos pensem sobre suas próprias idéias, reavaliando a sua percepção sobre o que é passagem de parâmetro e por que utilizá-la.

Após trabalhar o nível de conhecimento, compreensão e aplicação de cada conceito têm como objetivo desenvolver nos alunos o nível de análise, síntese e avaliação. No caso da análise e síntese é necessário propor atividades que instiguem os alunos a analisar como cada conceito se comporta individualmente e como diferentes conceitos se relacionam na solução de um problema único. Nesta fase, os problemas propostos devem exigir para as suas soluções a integração de diferentes conceitos. Desta forma o aluno passa a ter uma visão macro e não micro das estruturas de programação. Com relação a avaliação deve-se propor aos alunos que façam julgamentos sobre as soluções propostas, para verificar a exatidão, a efetividade, a economia de recursos computacionais e/ou a satisfação da atividade proposta.

4. Conclusão

É de suma importância que professores que lecionam algoritmos e programação de computadores tenham uma definição bem clara da concepção de ensino-aprendizagem desses conteúdos, para que os mesmos possam desenvolver metodologias que realmente

contribuam para o desenvolvimento das competências e habilidades exigidas, além de fazerem uso adequado de aplicativos destinados a este fim.

Logo o objetivo deste artigo foi apresentar uma visão geral sobre a concepção de ensino-aprendizagem de algoritmos e programação de computadores, pontuar as principais dificuldades enfrentadas pelos alunos e apresentar o perfil dos professores, tendo como base literaturas relacionadas com o assunto. Como conclusão da análise destes itens foi proposta uma metodologia de plano de aula que visa auxiliar o professor no processo de ensino-aprendizagem de algoritmos e programação de computadores.

Um próximo passo é validar a proposta na prática, pois desta forma será possível avaliar a sua eficácia e propor melhorias no processo. Além disso, há diferenças de perfis entre alunos e entre turmas e isto, geralmente, implica em adaptações das propostas de ensino-aprendizagem às características de seus públicos alvos.

Referências

[1] Raabe, André Luis Alice; Silva, Júlia Marques Carvalho da. Um Ambiente para Atendimento as Dificuldades de Aprendizagem de Algoritmos. XIII Workshop sobre Educação em Computação, São Leopoldo-RS, 2005.

[2] 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, RS, Brasil, 2002.

[3] Delgado, C., Xexeo, J. A. M., Souza, I. F., Campos, M., Rapkiewicz, C. E. "Uma Abordagem Pedagógica para a Iniciação ao Estudo de Algoritmos". XII Workshop de Educação em Computação (WEI'2004). Salvador, BA, Brasil, 2004.

[4] Forbellone, André Luiz Villar; Eberspächer, Henri Frederico. Lógica de Programação – 2° edição. São Paulo: Makron Books, 2000. ISBN: 85.346.1124-6.

[5] Xavier, G. M. C. et al. Estudo dos Fatores que Influenciam a Aprendizagem Introdutória de Programação. In: Escola Regional de Computação Bahia-Sergipe, Feira de Santana. Anais... Feira de Santana: Universidade Estadual de Feira de Santana, 2004.

[6] Delgado, C., Xexeo, J. A. M., Souza, M., Rapkiewicz, C. E., Junior, J.C.P. "Identificando Competências associadas ao Aprendizado de Leitura e Construção de Algoritmos". XXV SBC. Unisinos - São Leopoldo – RS, 2005.

[7] Lister, Raymond E Leany, John. First Year Programming: Let All the Flowers Bloom. Fifth Australasian Computing Education Conference, Australia, 2003.

[8] Neto, Wilson Castello Branco E Cechinel, Cristian. Uma Análise dos Problemas Enfrentados no Ensino-Aprendizagem de Fundamentos de Programação à Luz da Taxionomia de Bloom. XIV Workshop sobre Educação em Computação, 2006.

[9] Jesus, Andreia de. O processo de Ensino-Aprendizado de Algoritmos e Programação: Uso de Aplicativos. Monografia para a obtenção do titilo de especialista em Informática na Educação. Instituto Brasileiro de Pós-Graduação e Extensão – IBPEX, 2008.