
Interpretações da Taxonomia de Bloom no Contexto da Programação Introdutória

Elieser Ademir de Jesus, André Luis Alice Raabe

Universidade do Vale do Itajaí - UNIVALI
Itajaí – Santa Catarina – Brasil
{elieser, raabe}@univali.br

Abstract. *Applying Bloom's taxonomy in introductory programming is not trivial and some authors have reported difficulties in using it in their assessments. This article discusses the interpretation of each category of the taxonomy and how to use them in program assessments. Moreover, it presents an example assessment where the assessment items are classified according to Bloom's taxonomy. The goal is that the ideas discussed and the example assessment developed can be used as tools to aid in measuring the effectiveness of educational practices on programming learning.*

Resumo. *A aplicação da taxonomia de objetivos de aprendizagem de Bloom no contexto da programação introdutória nem sempre é trivial e alguns autores têm relatado dificuldades em utilizá-la na elaboração de avaliações. Neste artigo discute-se como cada uma das categorias da taxonomia vêm sendo interpretadas e utilizadas em avaliações de programação. Além disso, apresenta-se um instrumento de avaliação onde cada uma das questões foi classificada segundo a taxonomia de Bloom. O que se espera é que as idéias discutidas e o instrumento elaborado possam ser utilizados como ferramentas de auxílio na mensuração da efetividade das práticas educacionais relacionadas ao ensino de programação.*

Palavras-chave: Taxonomia de Bloom. Programação Introdutória. Algoritmos.

1. Introdução

As disciplinas de programação introdutória são consideradas como fundamentais nos cursos relacionados à computação. Em 2008, por exemplo, aconteceu pela segunda vez no Brasil o Workshop de Ambientes de Apoio à Aprendizagem de Algoritmos e Programação, workshop vinculado ao Simpósio Brasileiro de Informática na Educação (SBIE). A própria existência deste workshop mostra que mesmo com todas as pesquisas já realizadas na área as dificuldades na aprendizagem e os altos índices de desistência e repetência continuam existindo nas disciplinas relacionadas.

Entre os muitos desafios enfrentados por um professor de programação introdutória está a *avaliação*. Como um professor poderia mensurar objetivamente a aprendizagem de seus alunos? Supondo que os alunos obtenham ótimo desempenho, como saber, por exemplo, se o desempenho não foi causado por testes demasiadamente fáceis? O professor poderia comparar o desempenho de sua turma com uma *turma de referência*. Mas, nesse caso, como saber se a turma de referência é realmente referência?

McCracken *et al* (2001) realizaram um estudo conjunto entre universidades de alguns países com o objetivo de avaliar os alunos de programação introdutória. Nesta pesquisa foi elaborado um instrumento de avaliação através de um trabalho conjunto entre os professores envolvidos. De forma muito semelhante, Lister *et al* (2004) realizaram experimentos em sete países com o objetivo de testar alunos iniciantes na leitura e entendimento de código de programas. Novamente, os pesquisadores utilizaram um instrumento padrão de avaliação, de maneira que a comparação entre o desempenho das diferentes turmas pôde ser realizada. O mesmo instrumento de avaliação utilizado por estes autores foi reutilizado e aprimorado no trabalho de Whalley *et al* (2006). Os instrumentos de avaliação utilizados nas pesquisas de Whalley *et al* (2006) e Lister *et al* (2004) foram elaborados segundo a taxonomia de Bloom, uma estrutura conceitual concebida para auxiliar a definição de objetivos de aprendizagem.

A taxonomia de Bloom é bastante conhecida e utilizada em várias áreas do conhecimento. Para Whalley *et al* (2006), muitas das descrições dos níveis da taxonomia são difíceis de serem interpretados no contexto dos exercícios de programação. Thompson *et al* (2008) relatam uma situação onde foram observadas discrepâncias significativas entre as classificações sugeridas por diferentes professores para uma mesma questão de um teste. Neste sentido, Johnson e Fuller (2007) publicaram um trabalho com um título intrigante: *Is bloom's taxonomy appropriate for Computer Science?* Os autores colocam que, de forma geral, os professores de Ciência da Computação não consideram os termos *síntese* e *avaliação* úteis para a descrição dos objetivos de aprendizagem nas disciplinas de programação introdutória, e que consideram a *aplicação* como a habilidade mais importante a ser desenvolvida.

O objetivo deste artigo é discutir a interpretação da taxonomia de Bloom no contexto específico da programação introdutória e aplicar as idéias discutidas na elaboração de um instrumento de avaliação que possa ser utilizado por outros professores e pesquisadores.

A seguir discute-se a taxonomia de objetivos de aprendizagem de Bloom. Em seguida apresenta-se a taxonomia revisada e a interpretação dos seus níveis. Por fim, apresenta-se um instrumento de avaliação elaborado segundo a taxonomia e as conclusões deste trabalho.

2. Taxonomia de Bloom

Segundo Krathwohl (2002), Benjamin S. Bloom e seus colegas criaram a taxonomia buscando uma forma de facilitar a troca de questões de testes entre professores de várias universidades, cada questão avaliando o mesmo objetivo de aprendizagem. O autor diz que a taxonomia original de Bloom provê definições cuidadosas para as seis principais categorias do domínio cognitivo: conhecimento, compreensão, aplicação, análise, síntese e avaliação. Segundo o autor, estas categorias são ordenadas da mais simples para a mais complexa. Além disso, a taxonomia é uma hierarquia cumulativa, onde uma categoria mais simples é pré-requisito para a próxima categoria mais complexa.

Existem verbos associados a cada um dos níveis da taxonomia mencionados anteriormente. Estes verbos auxiliam na classificação de uma questão de avaliação em um dos níveis da taxonomia. Mais adiante apresenta-se estes verbos e como eles são especificamente interpretados no contexto da programação introdutória.

3. Taxonomia de Bloom Revisada

Apesar das revisões, segundo Fuller *et al* (2007) a versão da taxonomia de Bloom mais utilizada ainda é a original. Por outro lado, Fuller faz algumas críticas à taxonomia original, dizendo que as categorias nem sempre são fáceis de serem aplicadas, que existe uma sobreposição significativa entre elas e que existe algum debate sobre a ordem em que as categorias *análise*, *síntese* e *avaliação* aparecem na hierarquia.

Krathwohl (2002) diz que na taxonomia revisada existem duas dimensões. A dimensão do *Conhecimento* engloba as subcategorias da categoria *conhecimento* na taxonomia original. Já a dimensão dos *Processos Cognitivos* abrange as seis categorias da taxonomia original, porém renomeadas, em alguns casos apenas para suas formas verbais. A categoria *Conhecimento* tornou-se *Lembrar*, *Compreensão* tornou-se *Entender*, *Síntese* tornou-se *Criar* (e foi promovida para a categoria mais alta da hierarquia), *Aplicação*, *Análise* e *Avaliação* tornaram-se respectivamente *Aplicar*, *Analisar* e *Avaliar*. Os verbos associados a cada um dos níveis da taxonomia são apresentados na Tabela 1.

Tabela 1. Níveis da taxonomia revisada e seus respectivos verbos

1-Lembrar	2-Entender	3-Aplicar	4-Analisar	5-Avaliar	6-Criar
Reconhecer	Interpretar	Executar	Diferenciar	Verificar	Gerar
Relembrar	Exemplificar	Implementar	Organizar	Criticar	Planejar
	Classificar		Atribuir		Produzir
	Sumarizar				
	Inferir				
	Comparar				
	Explicar				

4. Interpretação da Taxonomia em Programação Introdutória

Em Whalley *et al* (2006) os autores descrevem seus esforços para categorizar as questões de um instrumento de avaliação de acordo com a taxonomia de Bloom. O instrumento criado pelos autores foi utilizado para mensurar as habilidades de leitura e compreensão de código de programadores iniciantes. Os autores dizem que mesmo para eles, um grupo experiente de professores de programação, muitas das descrições dos níveis da taxonomia são difíceis de ser interpretados no contexto dos exercícios de programação. Por conta deste tipo de dificuldades, a seguir descreve-se cada uma das categorias da taxonomia de Bloom e exemplos de como essas categorias são interpretadas e utilizadas no contexto da programação introdutória.

4.1 Interpretação da Categoria *Lembrar*

Thompson *et al* (2008) dizem que o processo cognitivo nesta categoria é definido como *recuperar conhecimento relevante da memória de longo termo*. Scott (2003) cita exemplos de tarefas que seriam classificadas no nível *Lembrar* da taxonomia: 1) Citar os nomes dos três tipos de *loops* em C++; 2) Listar três operações de entrada/saída em um computador; e 3) Escrever cinco coisas que são verdadeiras sobre uma arquitetura RISC.

4.2 Interpretação da Categoria *Entender*

Segundo Forehand (2009), esta categoria caracteriza-se pela *construção de significados através de linguagem oral, escrita ou gráfica, usando para isto a interpretação, exemplificação, classificação, sumarização, inferência e explicação*.

Hernán-Losada, Pareja-Flores e Velázquez-Iturbide (2008) exemplificam o nível da *Compreensão* na taxonomia original (*Entender* na taxonomia revisada) através de uma tarefa onde deve-se construir um programa que implementa a descrição de um algoritmo conhecido, representado em pseudo-código, fluxograma ou linguagem natural. Outro exemplo citado pelos autores é a utilização de tarefas onde os alunos recebem um trecho de código com partes faltantes que devem ser preenchidas com fragmentos de códigos.

Scott (2003) cita exemplos de questões de avaliação onde o aluno deve explicar em palavras o comportamento de um trecho de código e prever os valores de algumas variáveis a cada iteração de um *loop*.

4.3 Interpretação da Categoria *Aplicar*

Scott (2003) diz que na taxonomia original os verbos *aplicar, computar, demonstrar, manipular, modificar, produzir* e *resolver* estão associados à categoria *Aplicação*, renomeada para *Aplicar* na taxonomia revisada.

Thompson *et al* (2008) fornecem um exemplo onde o aluno deve executar mentalmente uma expressão como: $2 + 4 / 7 * 5 \% 3 = 7$. Segundo os autores, o que classifica uma questão como esta na categoria *Aplicar* é o fato de que o aluno aplica um processo conhecido (as regras de precedência das operações) para resolver um problema familiar, porém os dados do problema não são familiares para o aluno. Os autores ainda chamam a atenção para o fato de que embora a palavra *avaliar* seja utilizada na descrição do exemplo (*avaliar uma expressão*) o seu significado neste caso não é o mesmo do processo cognitivo *Avaliar* da taxonomia, que é descrito mais adiante.

4.4 Interpretação da Categoria *Analisar*

Thompson *et al* (2008) fornecem exemplos de duas questões que poderiam ser utilizadas para avaliar os alunos na categoria *Analisar*. Dado um código para uma classe de nome *Circulo*, as questões sugeridas pelos autores são: 1) O que é o método *Circulo* na classe? 2) Como ele se diferencia dos outros métodos da classe? Neste exemplo, as respostas esperadas seriam: 1) é um construtor; e 2) ele é invocado quando um novo objeto é criado. Segundo os autores, a primeira questão (o que é) envolve apenas lembrar que um método com o mesmo nome da classe é um construtor. Entretanto, na segunda questão os alunos devem diferenciar o construtor dos demais métodos da classe, o que enquadra a questão como um todo na categoria *Analisar*.

4.5 Interpretação da Categoria *Avaliar*

Thompson *et al* (2008) dizem que a categoria *Avaliação* pode ser definida como *a realização de julgamentos baseados em critérios e padrões*. Scott (2003) cita alguns exemplos de questões que enquadram-se na categoria *Avaliar*. O autor propõe que os alunos: encontrem um erro de lógica presente em um trecho de código dado; forneçam conjuntos de dados de teste para verificar um código e expliquem o que está sendo

testado em cada caso; e critiquem um código dado levantando seus pontos positivos e negativos.

4.6 Interpretação da Categoria Criar

Thompson *et al* (2008) definem a criação como *o ato de juntar elementos para formar um todo coerente e funcional*. Os exemplos desta categoria fornecidos pelos autores são: 1) propor um novo algoritmo alternativo ou hipotetizar que uma nova combinação de algoritmos resolverá o problema; 2) conceber um processo alternativo ou estratégia para resolver um problema; e 3) construir um segmento de código ou programa utilizando algoritmos inventados ou aplicando algoritmos conhecidos em uma combinação ainda não utilizada pelo aluno.

Na Tabela 2, é apresentada uma síntese das principais interpretações de cada um dos níveis da taxonomia de Bloom no contexto da programação introdutória.

5. Estudo de caso: Um Instrumento de Avaliação

Nesta seção é apresentado um instrumento de avaliação elaborado segundo as interpretações da taxonomia de Bloom discutidas neste artigo. Foram utilizadas questões de múltipla escolha para minimizar a subjetividade na correção. Como sugerem Lister *et al* (2004), o problema em se utilizar questões discursivas onde, por exemplo, o aluno descreve o funcionamento de um trecho de código é que o desempenho pode ser ruim não por falta de entendimento sobre o funcionamento de programas, mas sim pela própria dificuldade de escrever.

O instrumento de avaliação é focado em apenas três dos tópicos gerais de programação frequentemente citados pelos alunos iniciantes como alguns dos mais difíceis: os laços de repetição, os *arrays* e os desvios condicionais. Estes tópicos aparecem na lista dos mais difíceis nos trabalhos de Lahtinen, Ala-Mutka e Järvinen (2005), Dale (2006), Robins, Roundtree e Roundtree (2003) e Schulte e Bennedsen (2006).

Sete das dez questões do instrumento de avaliação são baseadas no trabalho de Whalley *et al* (2006), que por sua vez já é um aprimoramento do trabalho de Lister *et al* (2004), e três questões foram adicionadas. Os trechos de código foram traduzidos para o Português, uma pseudo-linguagem de programação. A maioria das questões são apenas descritas neste artigo, já que o espaço não possibilita que todos os detalhes e códigos sejam apresentados.

5.1 Questão 1 (Nível da taxonomia: Aplicar)

Nesta questão os alunos recebem um trecho de código onde um *loop* modifica o valor de um *contador* até que uma determinada condição seja satisfeita. Os alunos devem indicar qual o valor do contador ao final da execução do *loop*. Neste caso, a predição do valor de uma variável encaixa-se perfeitamente na categoria *Entender* da taxonomia. Entretanto, para fazer a predição os alunos precisam seguir a lógica de execução do algoritmo, respeitar a precedência dos operadores, etc, ou seja, precisam seguir um processo bem conhecido. Sendo assim, a atividade cognitiva dos alunos não resume-se a prever, pois exige que eles *apliquem* o conhecimento sobre a execução das instruções.

Tabela 2. Síntese da interpretação da taxonomia de Bloom em programação

Categoria	Interpretação da categoria em programação
	<i>Recuperação de conhecimento relevante da memória de longo termo.</i>
Lembrar	<p>Identificar elementos específicos em um trecho de código. Reconhecer a implementação de um determinado conceito. Reconhecer a descrição mais apropriada para um determinado conceito. Lembrar de um conceito, processo, algoritmo, etc. Listar operadores de acordo com a ordem de precedência. Definir o propósito de um método construtor. Descrever um determinado padrão de projeto. Citar os nomes dos tipos de <i>loops</i> em uma linguagem de programação. Listar <i>N</i> métodos que executem operações de entrada e saída de dados.</p>
	<i>Construção de significados através de diferentes tipos de linguagens.</i>
Entender	<p>Escrever em pseudo-código, fluxograma ou linguagem natural um programa que calcule uma fórmula bem conhecida. Completar partes faltantes de um programa utilizando fragmentos de código. Explicar com palavras o comportamento de um trecho de código. Predizer valores de variáveis depois da execução de um trecho de código. Traduzir um algoritmo de uma forma de representação para outra. Explicar um conceito, algoritmo ou padrão de projeto. Apresentar exemplos de um conceito, algoritmo ou padrão de projeto.</p>
	<i>Utilização de processos conhecidos para executar ou implementar.</i>
Aplicar	<p>Implementar um programa utilizando como exemplo um código que resolva um problema semelhante. Implementar ordenação de vetores não numéricos com alunos que já tenham ordenado vetores numéricos. Executar mentalmente expressões seguindo as regras de precedência. Resolver um problema familiar, mas com dados ou ferramentas não familiares. Modificar o código de um <i>loop</i> do tipo <i>for</i> para um do tipo <i>while</i>.</p>
	<i>Decomposição de um problema em suas partes constituintes e determinação das relações entre as partes e o todo.</i>
Analisar	<p>Dividir uma tarefa de programação em suas partes componentes. Organizar as partes componentes para atingir um objetivo geral. Identificar componentes críticos para o desenvolvimento. Identificar componentes ou requisitos não importantes. Diferenciar um método construtor dos demais métodos de uma classe.</p>
	<i>Realização de julgamentos baseados em critérios e padrões.</i>
Avaliar	<p>Determinar se um código satisfaz os requisitos definindo uma estratégia de teste apropriada. Criticar a qualidade de um código baseando-se em boas práticas de programação ou critérios de eficiência do código. Avaliar qual de dois algoritmos que resolvem a mesma tarefa é mais adequado. Encontrar um erro de lógica em um trecho de código dado.</p>
	<i>Juntar elementos para formar um todo coerente e funcional.</i>
Criar	<p>Propor algoritmo, processo ou estratégia alternativa para um problema. Hipotetizar que uma nova combinação de algoritmos resolverá o problema. Construir um programa utilizando algoritmos inventados. Aplicar algoritmos conhecidos em uma combinação não familiar para o aluno.</p>

5.2 Questão 2 (Nível da taxonomia: Entender)

Nesta questão os alunos recebem um trecho de código que apenas exibe o valor do contador de um laço de repetição. Em seguida são apresentados 4 fluxogramas. O aluno deve dizer qual dos fluxogramas possui a mesma lógica do trecho de código apresentado.

5.3 Questão 3 (Nível da taxonomia: Entender)

Nesta questão os alunos recebem o diagrama apresentado na Figura 2 e quatro trechos de código semelhantes ao apresentado na Questão 1. Os alunos devem indicar qual dos trechos de código possui a mesma lógica do diagrama. Como pode-se observar nesta questão o aluno faz o processo inverso daquele que é feito na questão anterior.

5.4 Questão 4 (Nível da taxonomia: Entender)

Nesta questão os alunos recebem um trecho de código e uma descrição textual do seu comportamento. O trecho de código verifica se o valor de uma variável numérica está dentro de um determinado intervalo usando um desvio condicional. A condição do desvio é justamente a parte faltante do código. São apresentadas quatro expressões condicionais que poderiam ser usadas para completar o código, e cabe aos alunos indicar qual das opções permite que o trecho comporte-se de acordo com a sua descrição textual.

Neste caso o aluno deve completar a parte faltante do algoritmo com um trecho de código dado, o que enquadra a questão na categoria *Entender* da taxonomia de Bloom. Entretanto, é possível argumentar que a questão enquadra-se na categoria *Analisar* da taxonomia, que entre outras coisas trata da *organização de partes componentes para atingir um objetivo geral*. Neste caso, é preciso atentar para o fato de que o aluno apenas *indica* qual parte completa o algoritmo. A questão trabalharia o processo cognitivo da *Análise* se, por exemplo, o aluno recebesse vários trechos de códigos embaralhados entre si e precisasse ordená-los para formar um algoritmo coerente.

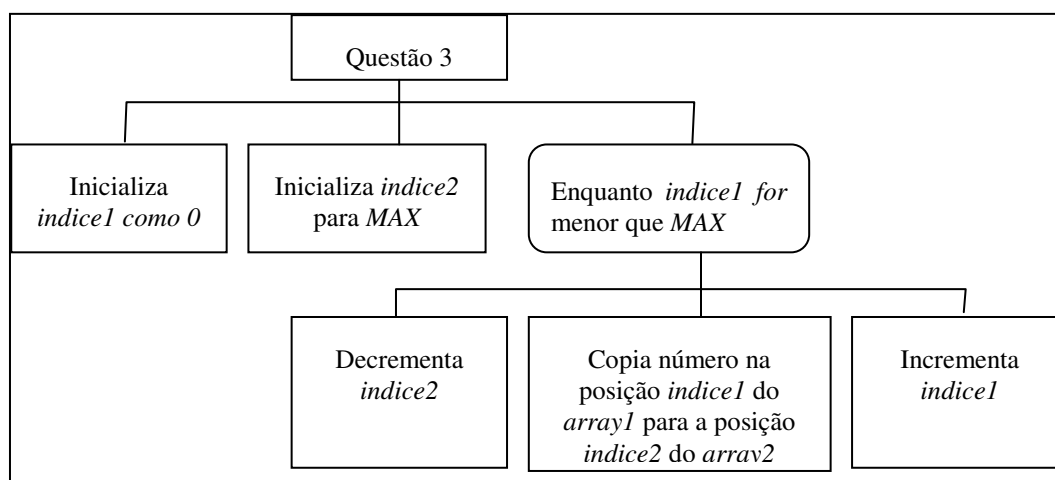


Figura 1. Diagrama da questão 3

5.5 Questão 5 (Nível da taxonomia: Aplicar)

Nesta questão os alunos também recebem um trecho de código onde um valor é solicitado ao usuário e em seguida dois desvios condicionais fazem testes lógicos com o valor solicitado. Um dos desvios é aninhado no outro. O primeiro desvio condicional verifica se o valor solicitado é par e o segundo, o aninhado, verifica se o mesmo valor é positivo. Dentro do desvio condicional mais interno existe um comando que exibe uma mensagem que está omitida, pois os alunos devem indicar qual dentre 4 mensagens é a correta para a situação. As opções possíveis são: **a)** “par e positivo”; **b)** “ímpar e positivo”; **c)** “maior que dois”; e **d)** “positivo ou zero”.

5.6 Questão 6 (Nível da taxonomia: Aplicar)

Esta questão assemelha-se à questão 5. O enunciado contém um trecho de código que contém dois desvios condicionais do tipo “IF”, um aninhado na seção “else” do outro. Duas variáveis são lidas através da entrada padrão. A condição do primeiro desvio é uma expressão relacional envolvendo as duas variáveis lidas. Dentro dos desvios existem comandos que exibem mensagens. O aluno recebe quatro opções de mensagens e deve indicar qual delas é a mensagem exibida no trecho de código.

5.7 Questão 7 (Nível da taxonomia: Aplicar)

Nesta questão os alunos vêem um trecho de código que utiliza um laço de repetição para somar todos os valores contidos em um *array*, com exceção do primeiro valor. Os alunos recebem quatro descrições sobre o comportamento do código e devem indicar qual das opções descreve corretamente o que o código executa.

5.8 Questão 8 (Nível da taxonomia: Analisar)

Nesta questão os alunos vêem um trecho de código que soma os elementos de um *array* até que a soma ultrapasse o valor armazenado em uma determinada variável. No enunciado é dito ao aluno que o código contém um erro. São apresentadas quatro opções que seriam as supostas correções para o trecho de código. Cabe ao aluno indicar qual das opções faz com que o trecho de código passe a executar corretamente de acordo com a descrição textual apresentada.

5.9 Questão 9 (Nível da taxonomia: Analisar)

Nesta questão os alunos recebem um trecho de código que ordena um *array* em ordem ascendente. O algoritmo está incompleto e o aluno deve indicar a seqüência de instruções que completa o algoritmo corretamente. Para isso o aluno deve ser capaz de indicar qual dentre as opções seguintes é a seqüência de instruções correta.

- a)** 4, 2, 6, 3, 1, 5 **b)** 2, 4, 6, 5, 1, 3 **c)** 4, 2, 6, 5, 1, 3 **d)** 2, 4, 6, 1, 5, 3

```

programa ex11
declarações
  defina TAM 5
  inteiro temp, i, j
  inteiro array[TAM]
inicio
  array <- {4, 3, 2, 1, 0}
  ???
  ???
  ???
  ???
  ???
  fimse
  fimpara
  fimpara
fim

```

Instruções que completam o algoritmo:

- 1) array[i] <- array[j]
- 2) para j <- i ate TAM-1 passo 1
- 3) array[j] <- temp
- 4) para i <- 0 ate TAM-1 passo 1
- 5) temp <- array[i]
- 6) se (array[j] < array[i]) entao

5.10 Questão 10 (Nível da taxonomia: Aplicar)

Nesta questão o aluno recebe um trecho de código que verifica se um *array* está ordenado ascendentemente. São apresentadas quatro descrições textuais de comportamento de algoritmos e o aluno deve indicar qual dos comportamentos se adequa ao trecho de código apresentado no enunciado.

6. Conclusões

Este artigo tratou da interpretação das categorias da taxonomia de Bloom em um contexto bastante específico: a programação introdutória. Alguns autores colocam que a aplicação da taxonomia neste contexto nem sempre é fácil. Sendo assim, foram apresentados exemplos de como as categorias da taxonomia podem ser interpretadas e utilizadas em testes de programação. Por fim, foi apresentado um instrumento de avaliação onde cada uma das questões foi classificada segundo a taxonomia de Bloom, de acordo com as idéias discutidas no artigo.

O instrumento de avaliação elaborado e apresentado neste artigo possibilitará que outros professores e pesquisadores tenham um instrumento padrão para utilizar em seus experimentos, principalmente quando comparando desempenho entre diferentes turmas de alunos. É pouco provável que o instrumento de avaliação elaborado se adéque a todas as realidades educacionais possíveis. Entretanto, mesmo nos casos onde o instrumento não seja adequado, total ou parcialmente, ele servirá como base para a construção de outros instrumentos.

É fundamental que utilizemos a experiência adquirida de outros pesquisadores e professores na elaboração de instrumentos e critérios de avaliação. O próprio instrumento que foi apresentado neste artigo não é outra coisa senão o aprimoramento do instrumento elaborado por Whalley *et al* (2006), que por sua vez também é uma melhoria do trabalho de Lister *et al* (2004). Ao reutilizar criticamente as contribuições deixadas por outros pesquisadores caminhamos sempre para resultados mais sólidos. Espera-se que este artigo desperte alguma crítica quanto ao uso da taxonomia de Bloom, seja no caso específico da programação introdutória ou no caso mais geral, pois é muito provável que também existam dificuldades de interpretação da taxonomia em outros contextos.

Referências

- Dale, Nell B. (2006) “Most difficult topics in CS1: results of an online survey of educators”, *ACM SIGCSE Bulletin*, v. 38, n.2, p. 49-53.
- Forehand, Mary. (2009) “Bloom's taxonomy”, <http://www.coe.uga.edu/epltt/bloom.htm>, maio.
- Fuller, Ursula *et al.* (2007) “Developing a Computer Science-Specific Learning Taxonomy”, In: *SIGCSE Bulletin, USA*, v. 39, n. 4, p. 152-170.
- Hernán-Losada, Isidoro; Pareja-Flores, Cristóbal; Velázquez-Iturbide, J. Ángel. (2008) “Testing-Based Automatic Grading: a proposal from Bloom's taxonomy”, In: *VII IEEE International Conference on Advanced Learning Technologies, IEEE Computer Society*, p. 847-849.
- Johnson, Colin. G.; Fuller, Ursula. (2007) “Is Bloom's taxonomy appropriate for computer science?”, In: *VI Baltic Sea Conference on Computing Education, ACM*, p. 120-123.
- Krathwohl, David R. (2002) “A revision of bloom's taxonomy: an overview”, In: *Theory into Practice*, n. 41, v. 4, p. 212-218.
- Lahtinen, Essi; Ala-Mutka, Kirsit; Järvinen, Hannu-Matti. (2005) “A study of the difficulties of novice programmers”, In: *X Annual SIGCSE conference on Innovation and Technology in Computer Science Education, ACM*, p. 14-18.
- Lister, Raymond *et al.* (2004) “A multi-national study of reading and tracing skills in novice programmers”, In: *ACM SIGCSE Bulletin*, v. 36, n. 4, p. 119-150.
- McCracken, Michael *et al.* (2001) “A Multi-National, Multi- Institutional Study of Assessment of Programming Skills of First-year CS Students”, In: *SIGCSE Bulletin*, n. 33, v. 4, p. 125-140.
- Robins, Anthony; Roundtree, Janet; Roundtree, Nathan. (2003), “Learning and Teaching Programming: a review and discussion”, In: *Computer Science Education*, v. 13, n. 2, p. 137-172.
- Scott, Terry. (2003) “Bloom's taxonomy applied to testing in computer science classes”, In: *Journal of Computing Sciences in College, USA*, v. 19, n. 1, p. 267-274.
- Schulte, Carsten; Bennedsen, Jens. (2006) “What do teachers teach in introductory programming?”, In: *International workshop on Computing education research, Canterbury, ACM*, p. 17-28.
- Thompson, Erol *et al.* (2008) “Bloom's taxonomy for CS assessment”, In: *X Australasian Computing Education Conference - ACE, Australian Computer Society*, p. 155-161.
- Whalley, Jacqueline L. *et al.* (2006) “An Australasian Study of Reading and Comprehension Skills in Novice Programmers, using the Bloom and SOLO Taxonomies”, In: *VIII Australasian Computing Education Conference (ACE2006), Computer Society*, p. 243-252.