Um Modelo de Avaliação do Pensamento Computacional na Educação Básica através da Análise de Código de Linguagem de Programação Visual

Nathalia da Cruz Alves

Orientadora: Dr. rer. nat. Christiane Gresse von Wangenheim, PMP

Coorientador: Dr. Jean C. R. Hauck

CBIE 2019

COMPUTAÇÃO NA ESCOLA

GQS Grupo de Qualidade de Software

UNIVERSIDADE FEDERAL DE SANTA CATARINA

Contextualização

Home › News & Events › Computer programming and coding in schools

Computer programming and coding in schools — an emerging trend

News | 06.03.2015 | 3 | 6 | 150

Inclusão da programação nos currículos escolares avança no exterior

Enquanto outros países já adotaram a área de conhecimento como base curricular, Brasil está na fase da experimentação

Why Computer Science Education in K-12 Settings Is Becoming Increasingly Essential

By ACM, the Association for Computing Machinery

EXAME | Revista EXAME | Reforma trabalhista | Black Friday | Ag

TECNOLOGIA

Escolas da Inglaterra ensinam alunos de 5 anos a programar

Reino Unido reformulou a maneira de ensinar computação às crianças do país adicionando aulas obrigatórias de programação

Por Sam Chambers
16 out 2014, 16h15

Ministério da Educação

Buscar no portal

Contato | Serviços do MEC | Área de imprensa

EDUCAÇÃO BÁSICA

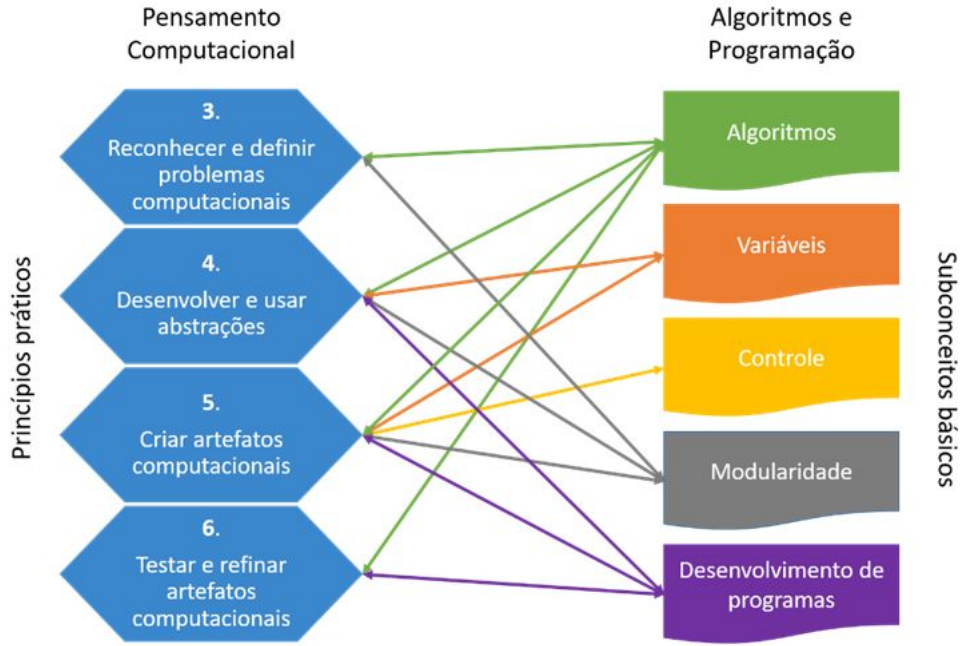
CNE debate incluir computação na Base Nacional Comum Curricular

Segunda-feira, 30 de julho de 2018, 18h56

Ensino de computação na Educação Básica

Ensino de computação envolve o ensino do **pensamento computacional**

K-12 Computer Science Framework - CSTA



Base Nacional Comum Curricular

"A área de Matemática, no Ensino Fundamental, centra-se na compreensão de conceitos e procedimentos em seus diferentes campos e no **desenvolvimento do pensamento computacional**, visando à resolução e formulação de problemas em contextos diversos." (MEC; BNCC, 2018, p. 471).

Linguagem de programação visual App Inventor

Ensino de **algoritmos e programação** pode ser por meio do **desenvolvimento de aplicativos** com App Inventor

App Inventor é uma linguagem de programação visual que permite:

- **Criar aplicativos** para Android
- **Programar** encaixando blocos visuais de comandos
- **Visualizar** em tempo real o aplicativo em qualquer *smartphone* Android

Interface do App Inventor

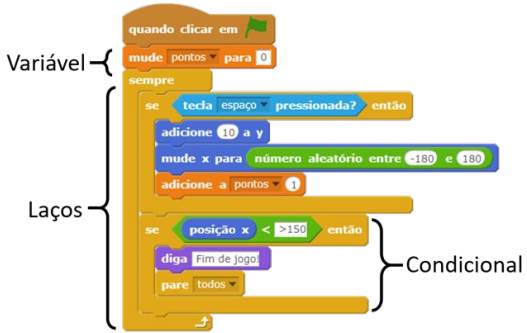
Processo de avaliação via código-fonte

Avaliação do **Pensamento Computacional**

O professor define um **tema** em uma **atividade aberta** → O aluno **programa** um aplicativo sobre o tema → O **código-fonte** do aplicativo criado pelo aluno é **analisado** → O aplicativo do aluno é **avaliado** com base nos dados da análise

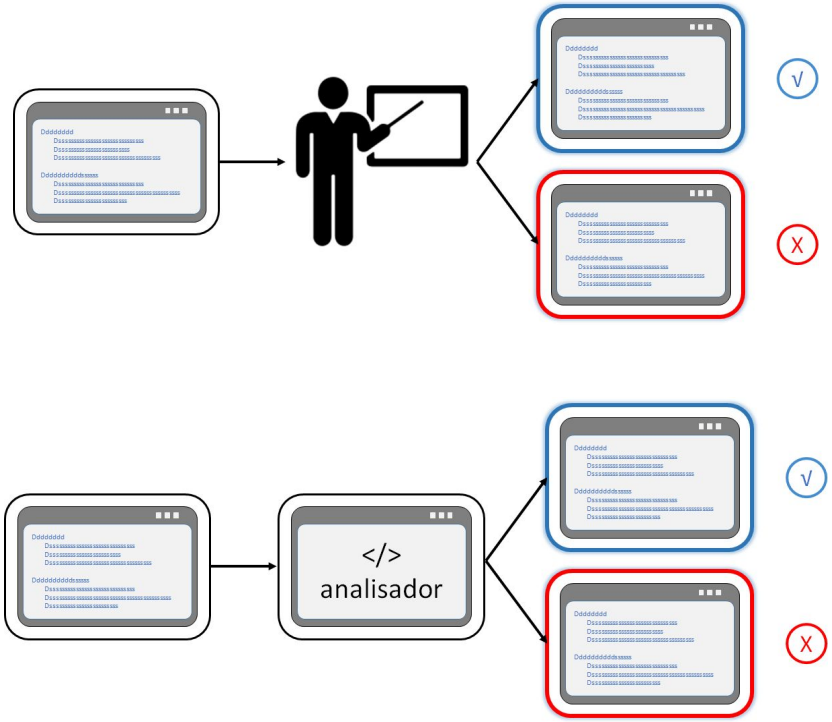
Exemplo:

Conscientização do combate à Dengue



Critério	Pontuação
Variável	3
Laços	2
...	...
Nota final:	**8**

Avaliação de atividades de programação



Avaliação pelo professor

- Tarefa repetitiva
- Diferentes professores enfatizam características diferentes
- Consome esforço e tempo considerável
- Requer conhecimento substancial da computação

Avaliação por um analisador automatizado

- Consistência
- Rapidez
- Libera o professor para avaliar outros pontos (criatividade) e/ou ajudar o aluno.

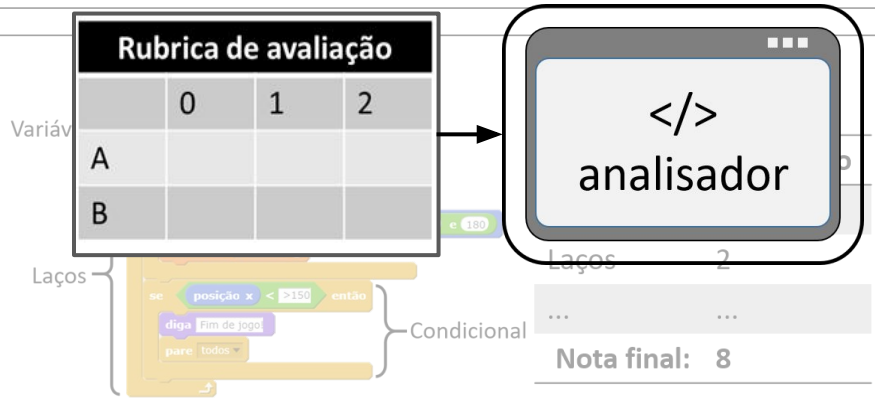
Processo de avaliação automatizado

Avaliação do **Pensamento Computacional**

O professor define um **tema** em uma **atividade aberta** → O aluno **programa** um aplicativo sobre o tema → O **código-fonte** do aplicativo criado pelo aluno é **analisado** → O aplicativo do aluno é **avaliado** com base nos dados da análise

Exemplo:

Conscientização do combate à Dengue



Pergunta de pesquisa

Como fornecer uma avaliação automatizada da qualidade de trabalhos práticos de programação com VPL baseada em blocos, dentro do contexto educacional da Educação Básica, de forma confiável e válida?

Objetivos

Objetivo geral: Desenvolver e avaliar um modelo conceitual de análise estática de código de linguagens de programação visual (VPL) baseada em blocos dentro do contexto educacional da Educação Básica

Objetivos específicos:

I. **Analisar o estado da arte** em relação a modelos de análise e avaliação de código referente a VPL baseada em blocos no contexto da Educação Básica.
II. **Conduzir uma avaliação do avaliador de código CodeMaster** em termos de confiabilidade, validade e qualidade.
III. **Desenvolver um modelo conceitual de avaliação genérico**, identificando de forma sistemática características do código a serem analisadas a partir da análise qualitativa da literatura existente e da análise do avaliador de código CodeMaster.
IV. **Evoluir a implementação do avaliador de código CodeMaster** com base no modelo genérico desenvolvido.
V. **Avaliar uma instância do modelo** desenvolvido em um estudo de caso avaliando a sua confiabilidade, validade e qualidade.

Método de pesquisa

Etapa	Atividades	Métodos	Resultados
Etapa 1 Síntese da fundamentação teórica	Sintetizar conceitos teóricos de análise estática Sintetizar o ensino de programação no ensino básico Sintetizar a metodologia de avaliação de trabalhos práticos de programação	Pesquisa bibliográfica (GIL, 2010)	Fundamentação teórica
Etapa 2 Levantamento do estado da arte	Analisar modelos de análise de código para atividades de programação no contexto educacional	Mapeamento Sistemático da Literatura (PETERSEN et al., 2008)	Análise do estado da arte
Etapa 3 Avaliação do modelo existente CodeMaster v1.0	Planejar a avaliação ↓ Avaliar o modelo existente usando programas provenientes da base de dados Galeria App Inventor ↓ Analisar a validade e confiabilidade do modelo	Estudo de caso (YIN, 2001) GQM (BASILI et al., 1994) - Alfa de Cronbach (CRONBACH 1951) - Análise fatorial (GLORFELD, 1995)	Avaliação do modelo CodeMaster v1.0
Etapa 4 Desenvolvimento do modelo	Indicar o propósito da avaliação e os objetivos ↓ Desenvolver critérios de avaliação para cada objetivo ↓ Revisar os critérios e objetivos ↓ Instanciar o modelo por meio de uma rubrica para App Inventor e definir a nota e *feedback* instrucional ↓ Implementar o modelo	Modelo ADDIE (BRANCH, 2009) Scoring Rubric Development (MOSKAL et al., 2000) Expert Panel (BEECHAM et al., 2005) Iterative and Incremental Development (LARMAN & BASILI, 2003)	Modelo desenvolvido
Etapa 5 Avaliação do modelo desenvolvido CodeMaster v2.0	Planejar a avaliação ↓ Avaliar o modelo desenvolvido usando programas provenientes da base de dados Galeria App Inventor ↓ Analisar a validade e confiabilidade do modelo	Estudo de caso (YIN, 2001) GQM (BASILI et al., 1994) - Alfa de Cronbach (CRONBACH 1951) - Análise fatorial (GLORFELD, 1995) - TRI (ANDRADE et al. 2000)	Avaliação do modelo CodeMaster v2.0

Estado da arte - Definição

Quais abordagens existem para analisar o desempenho do aluno a partir da análise do código de programas criados com VPL baseada em blocos no contexto da Educação Básica e quais as suas características?

Análise do programa	**PA1**. Quais abordagens existem e quais as suas características? **PA2**. Quais conceitos de algoritmos e programação relacionados à prática do pensamento computacional são analisados? **PA3**. Como os conceitos de algoritmos e programação relacionados à prática do pensamento computacional são analisados?
Avaliação e feedback instrucional	**PA4**. Se e como é gerada uma nota? **PA5**. Se e como o feedback instrucional é apresentado?
Automatização da avaliação	**PA6**. Se e como a abordagem foi automatizada?

Definição da busca

- Utilização da ferramenta SCOPUS
- Artigos na língua inglesa publicados entre 1997 e 2018

Execução da busca

- Em agosto de 2018
- Analisados 2550 resultados
- Identificados 23 artigos relevantes
- Os artigos descrevem 14 abordagens diferentes

Estado da arte - Panorama

PA1			PA2		PA3	PA4	PA5		PA6		
Nome da abordagem	VPL	Tipo de atividade	Conceitos do CSTA	Outros conceitos	Análise	Pontuação	Avaliação	Feedback instrucional	Público-alvo	Plataforma	Língua
Abordagem de Kwon e Sohn	AgentSheets	aberta	4	5	estática	Composta	Somativa		Professor	Não identificado	Inglês
Hairball	Scratch		2	1	estática	Politômica	Somativa e Formativa	via rubrica	Professor	Scripts em Python	Inglês
Dr.Scratch	Scratch	aberta				Politômica e Composta	Somativa e Formativa	via rubrica e dicas	Professor, aluno e instituição	Aplicação web	Espanhol Inglês, etc.
CTP/PBS/ REACT	AgentSheets	fechada				Politômica e Composta	Formativa		Professor	Aplicação web	Inglês
ITCH	Scratch	fechada	4	1	estática e	Não atribui	Somativa		Aluno	Scripts em Python	Inglês
PECT	Scratch	aberta	4	1			Somativa		Professor	Rubrica	Inglês
Ninja Code Village	Scratch	aberta	4	1			Somativa e Formativa		Aluno e professor	Aplicação web	Inglês e Japonês
Fairy Assessment	Alice	fechada	4	1	manual via rubrica					Rubrica	Inglês
Abordagem de Denner et al.	Stagecast Creator	fechada	4	5	manual via rubrica					Rubrica	Inglês
Scrape	Scratch	aberta	4	1	estática	Não atribui	-		Professor	Aplicação	Inglês
Quizly	App Inventor	fechada	4	1	dinâmica	Dicotômica	Formativa	via dicas	Prof adm		
Lambda	Snap!	fechada	4	0	estática e dinâmica	Dicotômica	Formativa		Aluno e professor	web	Inglês
CodeMaster	App Inventor e Snap!	aberta	4	1	estática	Politômica	Somativa e Formativa	via rubrica	Professor, aluno e administrador	Aplicaç web	
Abordagem de Funke et al.	Scratch	aberta	4	3	manual via rubrica	Politômica	Somativa		Professor	Rubric	

Maioria para Scratch

Maioria avalia apenas nuances dos conceitos

Muitas vezes não indicam de onde foram derivados

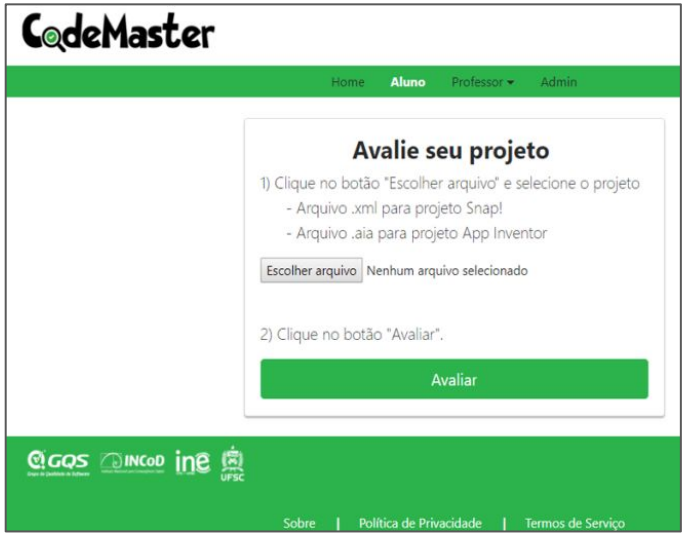
Análise estática => atividades abertas
Análise dinâmica => atividades fechadas

Ferramenta para professor e aluno

Maioria em inglês

CodeMaster v1.0

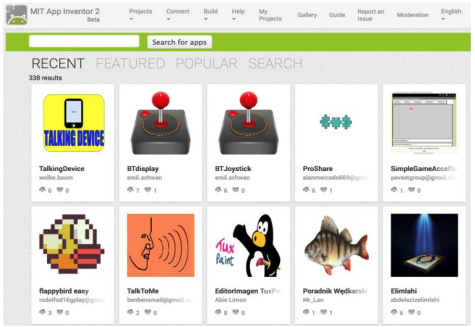
- Dimensão: pensamento computacional
- 2 subdimensões:
  - Conceitos e práticas de pensamento computacional
  - Pensamento computacional móvel
- Voltado para atividades abertas

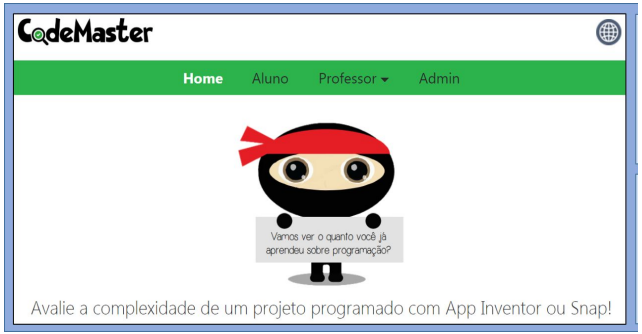


Avaliação do CodeMaster v1.0 em larga escala

A versão inicial do CodeMaster permite avaliar programas criados com App Inventor de forma **válida** e **confiável**?

- **Dados**: 88.864 aplicativos reais da Galeria AppInventor, 88.608 analisados
- **Itens**: avaliação de 15 critérios pelo CodeMaster
- **Pontuação**: escala ordinal 0 a 3, na qual cada item tem 4 possíveis respostas





Aplicativo	I01	I02	...	I15
1	3	2	...	0
2	1	0	...	2
...				
88.608	0	1	2	0

Resultados da avaliação do CodeMaster v1.0

Confiabilidade

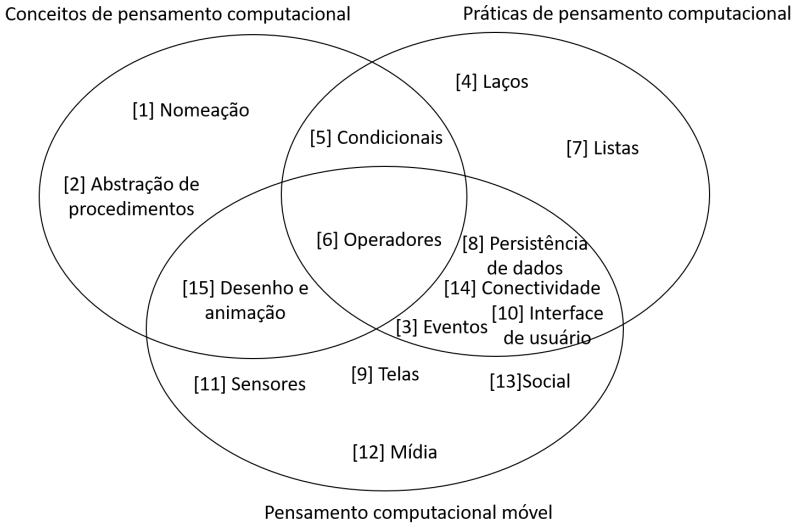
- Análise da consistência interna: Alfa de Cronbach (α=0,79)

Validade

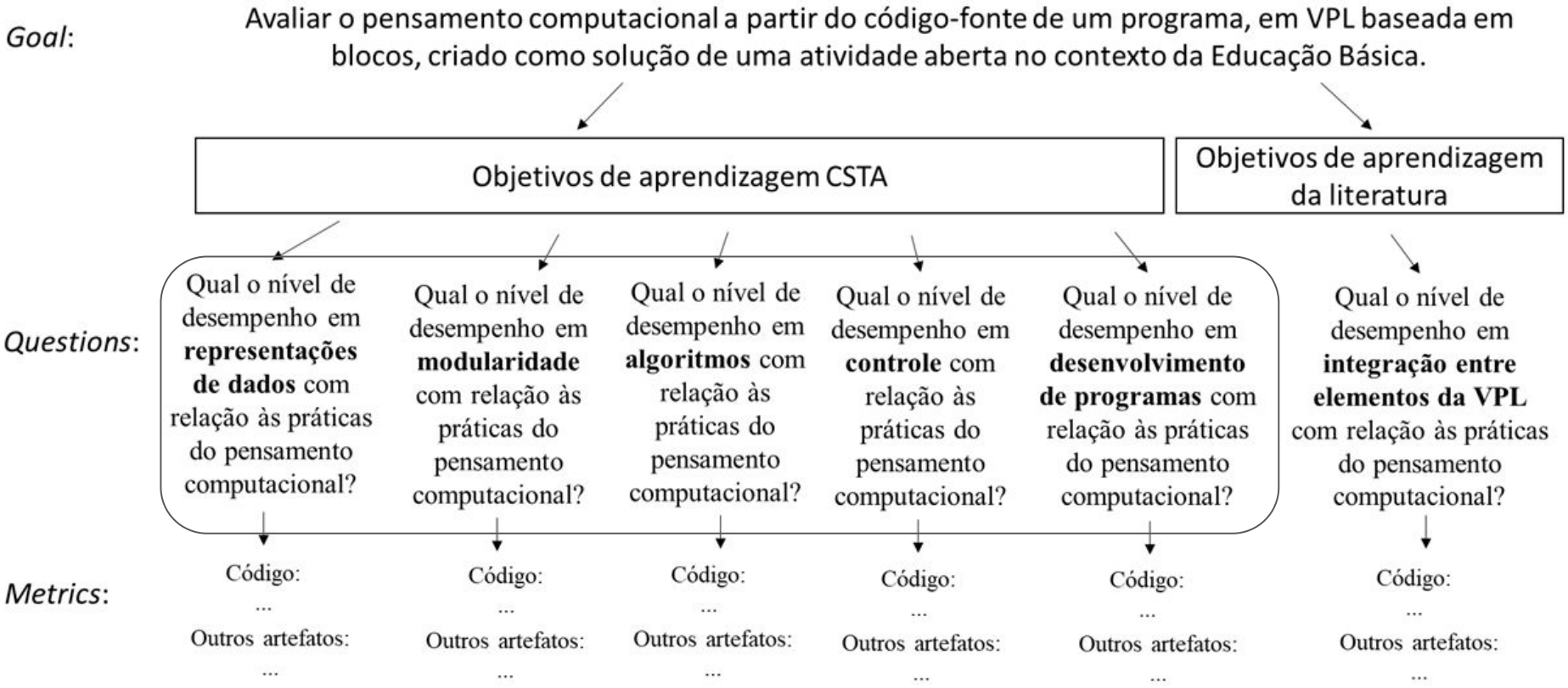
- Correlação entre os itens:
  - Itens de **conceitos e práticas do pensamento computacional** apresentam alta correlação
  - Itens de **pensamento computacional móvel** não apresentam validade convergente
  - Alguns apresentaram baixa correlação item-total: Mídia, Social e Conectividade
- Análise fatorial:
  - 3 subdimensões como resultado ≠ Proposto no modelo CodeMaster v1.0

Outros pontos fracos:

- Não está alinhado a um **guia de currículo**
- **Ausência** de alguns **itens essenciais** no contexto de desenvolvimento de aplicativos com App Inventor, como strings



Modelo proposto



Instanciação do modelo para App Inventor - Extrato

...

...

P2. Qual o nível de desempenho em **representação de dados** com relação às práticas do pensamento computacional? (CSTA, 2016; 2017)

Item	0 pontos	1 ponto	2 pontos	3 pontos
Variáveis: verificar se criou ou modificou valores de variáveis.	Sem uso de variáveis.	Modificação ou uso de variáveis predefinidas.	Criação e operação com variáveis	
Strings: verificar se criou ou modificou valores de strings.	Sem uso de strings.	Uso do comando de criação de string para alterar textos de elementos.	Criação e operação com strings.	
Nomeação: verificar se os nomes de variáveis são alterados do padrão.	Nenhum ou poucos nomes são alterado do padrão. (menos do que 10%)	De 10 a 25% dos nomes são alterados do padrão.	De 26 a 75% dos nomes são alterados do padrão.	Mais de 76% dos nomes são alterados do padrão.
Listas: verificar se são usadas listas.	Não usa listas.	Usa uma lista unidimensional.	Usa mais de uma lista unidimensional.	Usa uma lista de tuplas (map).

...

...

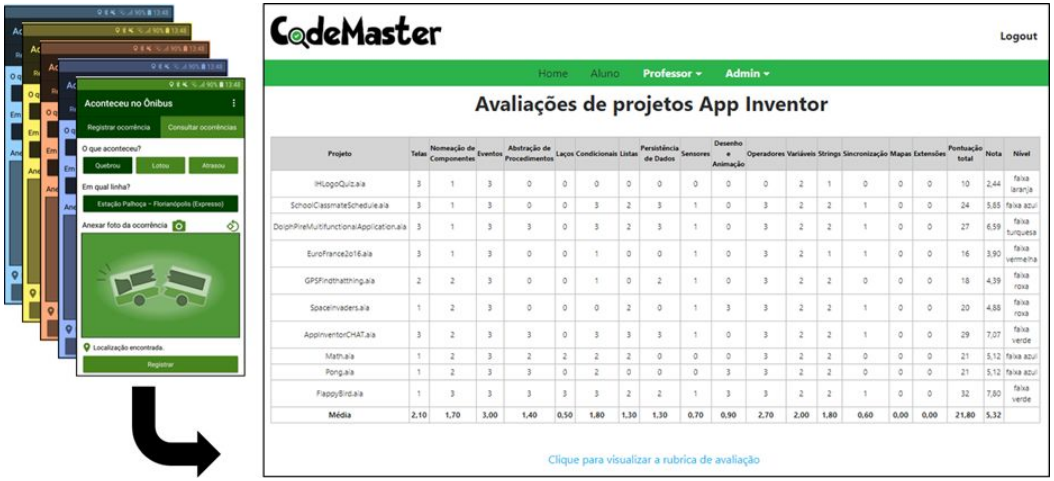
...

Modelo proposto - Implementação

- Desenvolvimento iterativo e incremental
- Linguagens de programação
  - Java com JSP (linguagem Java para desenvolvimento Web)
- Módulo de apresentação com JSP, JavaScript, HTML5 e CSS3

Avaliação de um aplicativo (aluno)

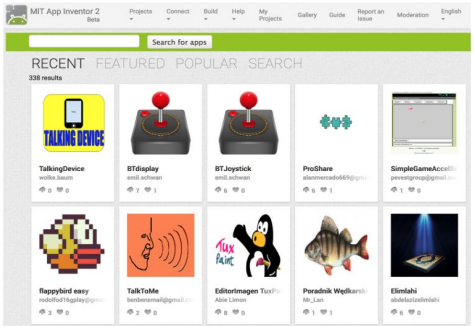
Avaliação de vários aplicativos (professor)

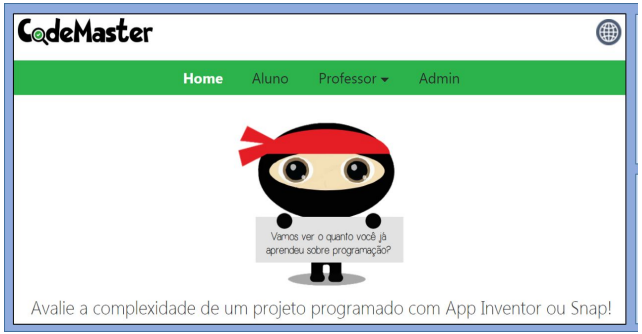


Avaliação do modelo CodeMaster v2.0

A versão 2.0 do CodeMaster permite avaliar programas criados com App Inventor de forma **válida** e **confiável**?

- **Dados**: 88.864 aplicativos reais da Galeria AppInventor
- **Itens**: avaliação de 16 critérios pelo CodeMaster v2.0
- **Pontuação**: escala ordinal 0 a 3, 0 a 2 e itens dicotômicos.





Aplicativo	I01	I02	...	I16
1	3	2	...	0
2	1	0	...	2
...				
88.812	0	1	2	0

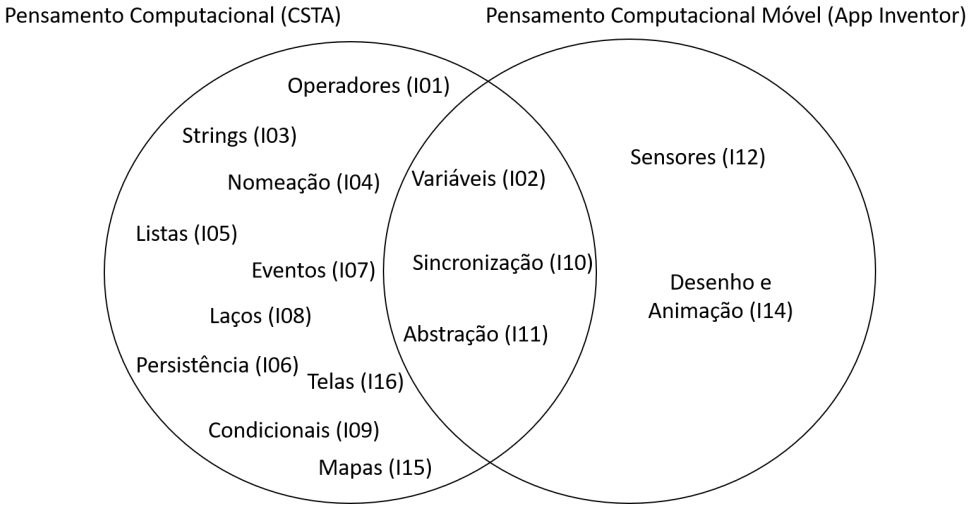
Resultados da avaliação do CodeMaster v2.0

Confiabilidade

- Análise da consistência interna: Alfa de Cronbach (α=0,84)

Validade

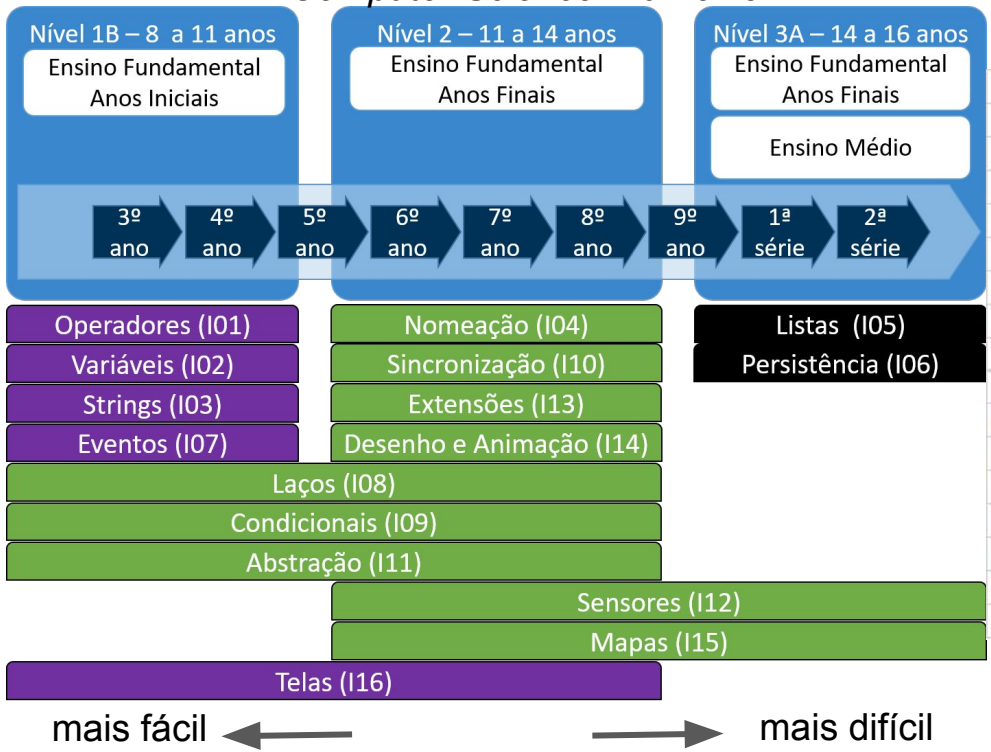
- Correlação entre os itens:
  - Itens de **algoritmos e programação** apresentam alta correlação
  - Itens de **integração entre elementos da VPL** não apresentam validade convergente
  - Apenas o item Mapas apresenta baixa correlação item-total
- Análise fatorial:
  - 2 subdimensões como resultado, confirmando as 2 subdimensões definidas
  - Item Mapas apresenta baixo fator de carregamento para 1 fator

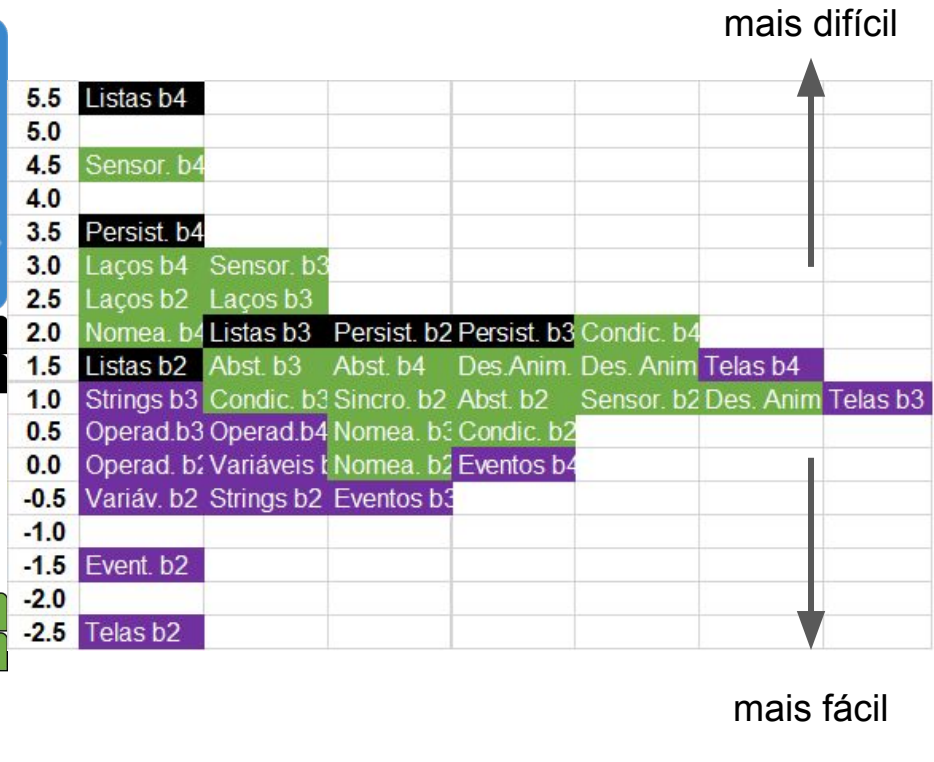


Resultados da avaliação do CodeMaster v2.0

Análise da **dificuldade** dos itens usando o **Modelo de Resposta Gradual** da Teoria da Resposta ao Item

K-12 Computer Science Framework





Conclusão

- Suporte **automatizado** para avaliação de programas no contexto da Educação Básica alinhado ao guia de currículo CSTA e à BNCC
  - Modelo de avaliação **válido** e **confiável**
  - Ferramenta **disponível on-line** (http://apps.computacaonaescola.ufsc.br:8080/)
- Publicações

ALVES, N. da C., GRESSE VON WANGENHEIM, C., HAUCK, J. C. R., BORGATTO, A. F. A Large-scale Evaluation of a Rubric for the Automatic Assessment of Algorithms and Programming Concepts. In: Proc. of the **51st ACM Technical Symposium on Computer Science Education** (SIGCSE), Portland/USA, 2020. (Qualis A1)

ALVES, N. d. C.; GRESSE VON WANGENHEIM,C.; HAUCK, J. C. R. Approaches to Assess Computational Thinking Competences Based on Code Analysis in K-12 Education: A Systematic Mapping Study. **Informatics in Education**, 18(1), 2019. (Qualis B1)

ALVES, N. da C., GRESSE VON WANGENHEIM, C., HAUCK, J. C. R., BORGATTO, A. F., ANDRADE, D. F. Uma Análise do Sequenciamento Pedagógico no Ensino de Computação na Educação Básica. Anais do VIII **Congresso Brasileiro de Informática na Educação**. Brasília/DF, 2019. (Qualis B1)

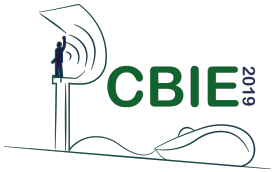
ALVES, N. da C., GRESSE VON WANGENHEIM, C. , HAUCK, J. C. R., BORGATTO, A. F., ANDRADE, D. F. Uma Análise das Diretrizes para Ensino de Pensamento Computacional propostas pela SBC na Educação Básica. Anais da **X Reunião da ABAVE**. São Paulo/SP, 2019. (resumo - cartaz)

ALVES, N. da C., GRESSE VON WANGENHEIM, C. , HAUCK, J. C. R., BORGATTO, A. F., ANDRADE, D. F. CodeMaster: Um Modelo de Avaliação do Pensamento Computacional na Educação Básica através da Análise de Código de Linguagem de Programação Visual. Anais da **X Reunião da ABAVE**. São Paulo/SP, 2019. (resumo)

- Trabalhos futuros
  - Avaliar com uma **base de dados mais recentes**
  - **Avaliar outras competências** como o design visual
  - Criar uma **escala com interpretação pedagógica**

Obrigada! Perguntas?

Um Modelo de Avaliação do Pensamento Computacional na Educação Básica através da Análise de Código de Linguagem de Programação Visual



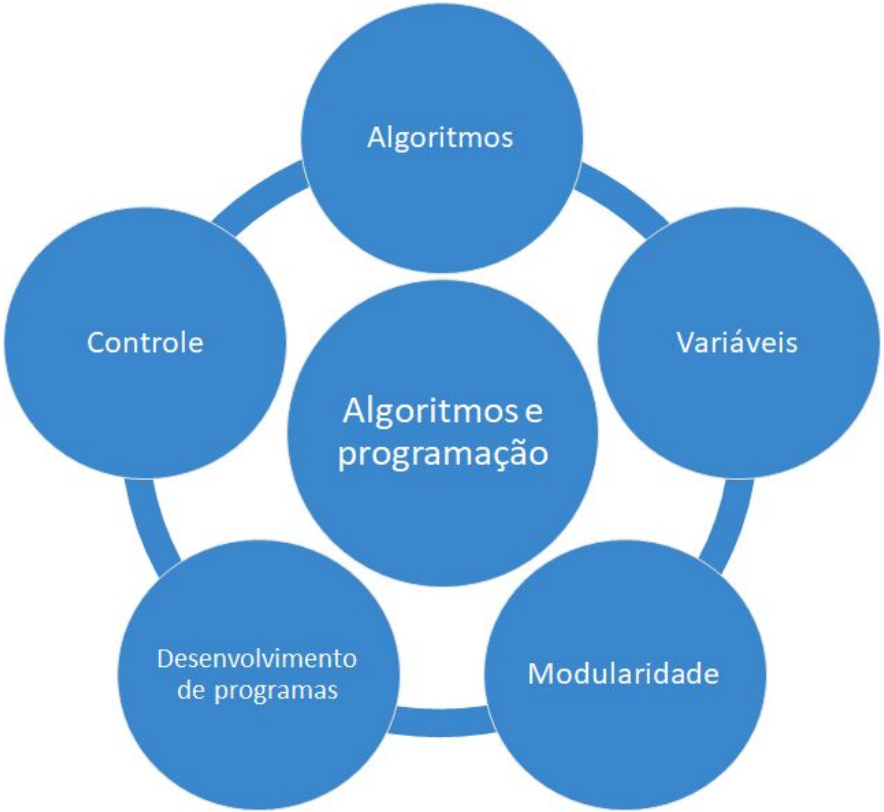
Nathalia da Cruz Alves

nathalia.alves@posgrad.ufsc.br

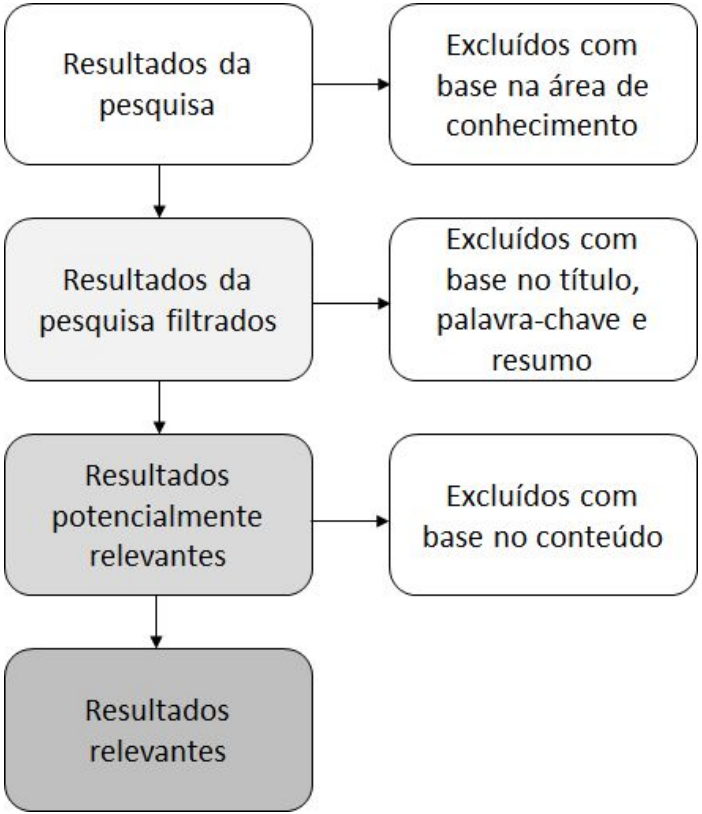
Slides extras

K-12 Computer Science Framework - CSTA

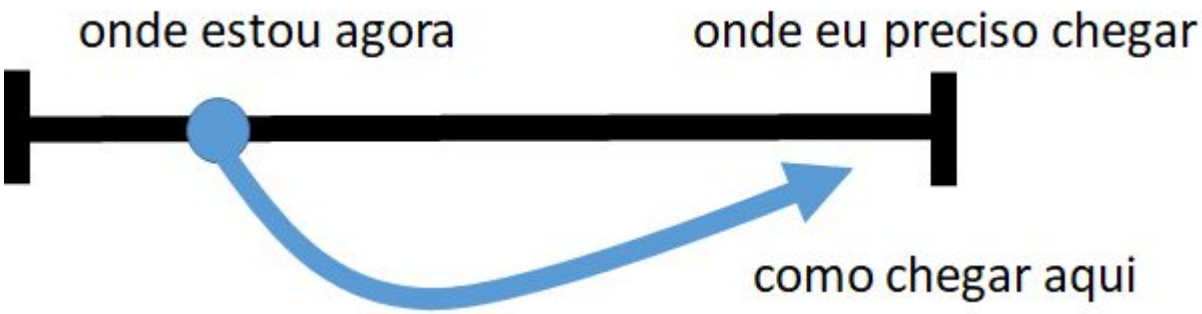
Algoritmos e programação



Levantamento do estado da arte



Feedback



CodeMaster

http://apps.computacaonaescola.ufsc.br:8080/rubrica_appinventor.jsp

Criteria	Level of Performance			
	0	1	2	3
Screens	Single screen with visual components that do not programmatically change state.	Single screen with visualcomponents that do programmatically change state.	Two screens with visual components and one screen with visual components that do programmatically change state.	Two or more screens with visual components and two or more screens with visual components that do programmatically change state.
User Interface	Uses one visual component without arrangement.	Uses two or more visualcomponents without arrangement.	Uses five or more visual components with one type of arrangement.	Uses five or more visual components with two or more types of arrangement.
Naming: Components, Variables, Procedures	Few or no names were changed from their defaults.	10 to 25% of the names were changed from their defaults.	26 to 75% of the names were changed from their defaults.	More than 75% of the names were changed from their defaults.
Events	No use of any type of event handlers.	Uses one type of event handlers.	Uses two types of event handlers.	Uses more than two types of event handlers.
Procedural Abstraction	No use of procedures.	There is exactly one procedure, and it is called.	More than one procedure is used.	There are procedures for code organization and reuse (with more procedure calls than procedures).
Loops	No use of loops.	Uses simple loops ("while").	Uses "for each" loops with simple variables.	Uses "for each" loops with list items.
Conditional	No use of conditionals.	Uses "if".	Uses one "if then else".	Uses more than one "if then else".
Operators	No use of any operators blocks.	Uses one type of operator blocks.	Uses two types of operator blocks.	Uses more than two types of operator blocks.
Lists	No use of lists.	Uses one single-dimensional list.	Use more than one single-dimensional list.	Uses lists of tuples.
Data persistence	Data are only stored in variables or UI component properties, and do not persist when app is closed.	Data is stored in files (File or FusionTables).	Uses local databases (TinyDB).	Uses web databases (TinyWebDB or Firebase).
Sensors	No use of sensors.	Uses one type of sensor.	Uses two types of sensors.	Uses more than two types of sensors.
Media	No use of media components.	Uses one type of media components.	Uses two types of media components.	Uses more than two types of media components.
Social	No use of social components.	Uses one type of social components.	Uses two types of social components.	Uses more than two types of social components.
Connectivity	No use of connectivity components.	Uses activity starter.	Uses bluetooth connection.	Uses low level web connection.
Drawing and Animation	No use of drawing and animation components.	Uses canvas component.	Uses ball component.	Uses image sprite component.