Análise do Nível de Dificuldade dos Conceitos de Design de Interface de Usuário usando a Teoria de Resposta ao Item

Nathalia da Cruz Alves[1], Igor Silva Solecki[1], Christiane Gresse von Wangenheim[1], Adriano Borgatto[1], Jean Carlo Rossa Hauck[1], Miriam Nathalie Fortuna Ferreira[2]

[1]Departamento de Informática e Estatística
[2]Departamento de Expressão Gráfica
Universidade Federal de Santa Catarina (UFSC) – Florianópolis – SC – Brasil

{nathalia.alves,igor.solecki}@posgrad.ufsc.br,{c.wangenheim,
adriano.borgatto,jean.hauck}@ufsc.br,nathalie.fortuna@prosgrad.ufsc.br

Abstract. *Computing education can be introduced in K-12 through the development of apps with App Inventor, involving not only algorithms & programming but also user interface (UI) design concepts and practices. However, the sequencing of effective instructional units needs to be designed based on quantitative difficulty estimates. Therefore, this article presents the results of a large-scale analysis of the difficulty of UI design concepts of 1870 App Inventor projects using Item Response Theory. The results indicate concepts more easy (typography) and difficult ones (images and color), partly due to a lack of support of the App Inventor environment more aligned with visual design guides.*

Resumo. *O ensino de computação pode ser introduzido na Educação Básica por meio do desenvolvimento de aplicativos com App Inventor, envolvendo não apenas algoritmos e programação, mas também conceitos e práticas de design visual. No entanto, o sequenciamento de unidades instrucionais eficazes precisa ser projetado com base em estimativas quantitativas de dificuldade. Assim, este artigo apresenta os resultados de uma análise em larga escala do nível de dificuldade dos conceitos de design de UI de 1870 projetos App Inventor usando Teoria da Resposta ao Item. Os resultados indicam conceitos mais fáceis (tipografia) e difíceis (imagens e cores), resultantes da falta de um suporte mais alinhado aos guias de design visual no App Inventor.*

1. Introdução

O ensino da computação na Educação Básica é tipicamente focado em algoritmos e programação [Grover e Pea 2013; CSTA 2016]. Uma das alternativas de ensino é o desenvolvimento de aplicativos móveis utilizando o App Inventor (http://ai2.appinventor.mit.edu), um ambiente de programação baseada em blocos que permite projetar a interface de usuário e programar a parte funcional do aplicativo. Assim, além da corretude da funcionalidade, a usabilidade e estética visual são características que também podem influenciar no sucesso de um aplicativo móvel [Kumar et al. 2018], e, portanto, devem ser abordadas no ensino de computação. O design visual se organiza a partir de meta-princípios (consistência, hierarquia e personalidade), considerando os princípios de contraste, uniformidade, coerência e *layout*, e envolve o uso de elementos como cor, tipografia e imagens [Schlatter e

DOI: 10.5753/cbie.sbie.2020.1563

Levinson 2013]. O uso equilibrado desses elementos pode resultar em uma interface bonita e agradável afetando também a usabilidade [Rogers et al. 2011].

Assim, o design de *User Interface* (UI) é essencial para o desenvolvimento de software e faz parte do corpo de conhecimento de computação visando maximizar a usabilidade e a experiência do usuário [ACM 2013]. As competências relacionadas ao design de UI são importantes não apenas para os profissionais de Tecnologia da Informação, mas para qualquer pessoa, pois estão relacionadas às habilidades do século XXI [AIGA 2013]. Inclusive a Base Nacional Comum Curricular (BNCC) propõe o exercício da curiosidade intelectual e o uso de diferentes linguagens (visual e digital) [MEC 2018], como também diretrizes curriculares referentes ao ensino de computação indicam a sua importância na Educação Básica [CSTA 2016]. Alguns países já começaram a incorporar o ensino de design para estudantes da Educação Básica, como China e Coréia do Sul. Também já existem vários esforços de ensino de design [Ferreira et al. 2018] de forma criativa (http://www.cityxproject.com) ou o laboratório K-12 da Stanford *dschool* (https://dschool.stanford.edu/programs/k12-lab-network) no contexto do movimento *maker*, assim como unidades instrucionais especificamente destinadas a ensinar conceitos básicos de design de UI [Agapie e Davidson 2018], inclusive para apps criados com App Inventor [Ferreira et al. 2019a; Ferreira et al. 2020].

A inserção do ensino de conceitos de design de UI na Educação Básica proporciona diversos benefícios, incluindo a aprendizagem de métodos de pesquisa, visualização e apresentação de informações, incentivo à análise crítica, colaboração e treinamento [AIGA 2013]. Aprender design pode tanto incentivar os alunos a serem imaginativos, quanto ensiná-los como aproveitar essa inventividade e colocá-la em prática [AIGA 2013]. Desta forma, a aprendizagem das competências de design fornece soluções e formas de engajamento com o mundo, que permitem aos indivíduos atuarem como agentes de mudança e criação do século XXI [Christensen et al. 2016]. Outro benefício da incorporação de conteúdos relacionados ao design de UI no ensino de computação é a ampliação da percepção de que a computação é mais do que apenas programar e pode, assim, aumentar o interesse nessa área e motivar os alunos para seguir uma carreira na área de TI [Robinson et al. 2015].

Aprender design de UI, no entanto, é um processo complexo e por essa razão os iniciantes têm diversas dificuldades, as quais são evidenciadas pela falta de estética visual e pelo baixo grau de conformidade com guias de estilo nos aplicativos criados pelos alunos [Solecki et al. 2020a]. Desta forma, para que unidades instrucionais sejam efetivas, é necessário que sejam cuidadosamente projetadas considerando não apenas o conteúdo, mas também o seu sequenciamento. Tendo em vista que a ordem e organização de atividades de ensino afetam a forma na qual a informação é processada e assimilada [Patten et al. 1986], é importante sequenciar o conteúdo de forma que seja mais facilmente compreendido pelo aluno, melhorando a transferência de conhecimento e auxiliando os alunos a atingirem objetivos [Morrison et al. 2010]. Existem diversas formas de sequenciar conteúdos, por exemplo, adotando uma estratégia do mais simples para o mais complexo de acordo com os tipos de objetos de conhecimento.

No entanto, encontrar uma sequência ideal de objetos de conhecimento não é uma tarefa fácil. Sendo assim, é importante investigar as dificuldades dos alunos em relação aos objetos de conhecimento de design de UI. Existem diversos trabalhos que

examinam as dificuldades dos alunos no contexto de desenvolvimento de aplicativos com App Inventor, focando na abstração de procedimentos [Li et al. 2017], eventos [Turbak et al. 2014], efetividade [Park e Shin 2019] e adequação [Papadakis et al. 2017] do App Inventor como ambiente de ensino. Outros trabalhos também analisam a progressão da aprendizagem no ensino de computação na Educação Básica [Xie e Abelson 2016], contudo, sem focar no ensino de design de UI.

Portanto, o objetivo deste estudo é analisar a dificuldade observada no design de UI em projetos desenvolvidos com App Inventor. Considerando essas habilidades como um construto psicométrico latente, a Teoria da Resposta ao Item (TRI) [De Ayala 2009] é utilizada para obter estimativas quantitativas de dificuldade para cada objeto de conhecimento [Alves et al. 2019]. A TRI se refere a uma família de modelos matemáticos que tentam explicar a relação entre traços latentes (características ou atributos não observáveis, como o conhecimento de conceitos de design de UI) e suas manifestações (como o desempenho observado, p.ex., o uso de cores em projetos do App Inventor). A dificuldade dos itens de design de UI é analisada com base nos itens da rubrica CodeMaster [Solecki et al. 2020b; Gresse et al. 2018] extraindo-os automaticamente de projetos criados com App Inventor. Os resultados fornecem informações sobre a dificuldade da aplicação dos conceitos, bem como a distribuição estatística desses conceitos nos projetos App Inventor. Os resultados deste estudo podem ser utilizados por designers instrucionais a fim de orientar o sequenciamento de objetos de conhecimento de design de UI na Educação Básica. Podem também ser usados para evoluir sistematicamente o suporte tecnológico fornecido pelo ambiente de programação App Inventor, a fim de facilitar o aprendizado de design de UI.

2. Metodologia

Adotando a abordagem *Goal Question Metric* [Basili et al. 1994], o objetivo deste estudo é definido como: analisar a dificuldade dos conceitos de design de UI em projetos criados com App Inventor utilizando a rubrica CodeMaster [Solecki et al., 2020b]. Para a análise, é utilizada uma amostra de 1870 projetos dentre todos os projetos disponíveis na Galeria do App Inventor (http://ai2.appinventor.mit.edu). Na composição da amostra foram incluídos projetos que atendem pré-requisitos para uma amostra equilibrada, incluindo 250 aplicativos aleatórios para cada um dos 2 ou 3 níveis de desempenho de cada item da rubrica CodeMaster e pelo menos 750 projetos com pontuação para cada um dos item, sendo que alguns projetos satisfazem vários requisitos ao mesmo tempo, resultando numa amostra de 1870 projetos. Os projetos foram avaliados automaticamente usando a ferramenta CodeMaster para os conceitos de design de UI, extraindo-os do código-fonte por meio de uma análise estática do código.

A rubrica CodeMaster UI Design - App Inventor [Solecki et al. 2020b] foi construída com base na teoria e guias de estilo e acessibilidade proeminentes, incluindo o *Material Design* (https://material.io/design) e o guia W3C para acessibilidade móvel [W3C 2015], com o objetivo de avaliar a conformidade do design visual. A rubrica é automatizada por meio de uma ferramenta web e foi amplamente avaliada evidenciando a sua confiabilidade (alfa de Cronbach = 0,84) e validade [Solecki et al. 2020b].

Tabela 1. Rubrica CodeMaster UI Design – App Inventor

Item	0pt	1pts	2pts
Layout			
L1. Todos os componentes alvos de toque têm largura e altura maior ou igual a 48 pixels?	Não		Sim
L2. Todos os botões têm o mesmo formato?	Não		Sim
L3. Botões agrupados na interface sempre têm o mesmo tamanho?	Não		Sim
L4. Qual é o número mínimo e o número máximo de elementos nas telas?	min < 2 ou max ≥ 20	min ≥ 2 e max ∈ [10, 19]	min ≥ 2 e max ≤ 9
Tipografia			
T1. Todos os componentes usam família da fonte sem serifa?	Não		Sim
T2. Todos os botões com texto têm tamanho da fonte igual a 14?	Não		Sim
T3. Todos os componentes (exceto botões) têm um dos tamanhos de fonte recomendados pelo *Material Design*?	Não		Sim
T4. Há texto em itálico?	Sim		Não
Escrita			
E1. Todos os botões têm texto todo em letras maiúsculas?	Não		Sim
E2. Todas as sentenças começam com letra maiúscula ou dígito?	Não		Sim
E3. Todos os componentes têm texto diferente do padrão (p. ex., "Texto para Botão1")?	Não		Sim
E4. Existem legendas que terminam com ":" (dois pontos)?	Sim		Não
E5. Existem sentenças que terminam com "." (ponto)?	Sim		Não
E6. O texto de botão mais longo tem quantos caracteres?	15 ou mais	De 8 a 14	7 ou menos
Cores			
C1. Quantas cores são usadas no aplicativo (além de preto, branco e cinza, incluindo cores de imagens)?	4 ou mais, ou nenhuma	3	1 ou 2
C2. Qual é o nível WCAG do aplicativo em relação ao contraste do texto?	Insuficiente	Nível AA	Nível AAA
C3. Usam-se apenas cores da paleta do *Material Design*?	Não		Sim
C4. As tonalidades de cores usadas são harmônicas entre si (variantes, complementares, análogas ou triádicas)?	Não		Sim
Imagens			
I1. Todos os ícones usados são do catálogo de ícones do Material Design?	Não		Sim
I2. Existem imagens pixelizadas?	Sim		Não
I3. Existem imagens distorcidas?	Sim		Não

Para analisar a dificuldade, é utilizado o Modelo de Resposta Gradual (MRG) da TRI [Samejima 1969]. A escolha desse modelo se deu devido às características dos itens que possuem duas ou mais categorias de respostas ordenadas. O MRG permite analisar as propriedades do item, incluindo a dificuldade. Isso é feito estimando a correspondência entre uma característica latente não observada (conhecimento sobre o design de UI) e evidências observáveis (projetos App Inventor avaliados).

O MRG propõe um modelo probabilístico para estimação de parâmetros que não depende de um conjunto específico de itens e é usado para determinar a probabilidade de alguém receber uma pontuação específica (ou superior), dado o nível do traço latente subjacente. Para cada item, são estimados: o parâmetro *a*, que representa o parâmetro de inclinação do item (comum a todas as categorias de itens), os parâmetros *b*, indicando as dificuldades dos níveis de desempenho dos itens, com *b2* representando a dificuldade de obter pontuação 1 em qualquer item e *b3* representando a dificuldade de obter pontuação 2 em qualquer item, bem como os respectivos erros-padrão (EP) de estimação.

Para usar o MRG para o modelo unidimensional, primeiro é necessário garantir que a rubrica (Tabela 1) é unidimensional. Nesse sentido, foi realizada uma análise gráfica (*scree plot*) por meio de análise paralela, com resultados indicando que a rubrica CodeMaster possui possui um fator preponderante [Solecki 2020c], isto é, unidimensionalidade, pré-requisito para usar o MRG unidimensional [Samejima 1969].

3. Resultados

3.1 Estimação dos parâmetros e posicionamento na escala

Na estimação de parâmetros, valores do parâmetro de inclinação (a) maiores do que 0,7 são considerados bons [Bortolotti et al. 2012]. Os valores dos parâmetros de dificuldade (*b2* e *b3*) variam de -∞ a +∞, porém, na prática, são esperados valores dentro do intervalo [-4, 4] [Rabelo 2013]. Como resultado, utilizando a avaliação de 1870 aplicativos da galeria do App Inventor realizada pelo CodeMaster UI Design - App Inventor, a maioria dos parâmetros foi estimada com valores considerados bons, apresentando parâmetro de inclinação (*a*) maior que 0,7 e parâmetros de dificuldade (*b2* e *b3*) dentro do intervalo [-4, 4] (Tabela 2).

Tabela 2. Parâmetros da TRI estimados para os itens da rubrica CodeMaster UI Design – App Inventor [Solecki 2020c]

Subdimensão	Item	a	EP(a)	b2	EP(b2)	b3	EP(b3)
Layout (L)	L1. tamanhoDeAlvosDeToque	**1,238**	0,080	-0,251	0,051		
	L2. formatoDosBotoes	**1,708**	0,125	-1,168	0,062		
	L3. tamanhoConsistenteDeBotoes	**1,449**	0,093	0,254	0,045		
	L4. densidadeDaIU	**1,401**	0,079	-0,797	0,055	-0,127	0,045
Tipografia (T)	T1. familiaFonte	**1,290**	0,094	-1,558	0,092		
	T2. tamanhoFonteBotoes	**1,214**	0,086	-0,346	0,056		
	T3. tamanhoDaFonte	**1,172**	0,081	-0,115	0,054		
	T4. evitarItalico	**1,276**	0,090	-1,399	0,084		
Escrita (E)	E1. caixaAltaBotao	0,461	0,078	3,195	0,533		
	E2. capitalizacaoSentenca	**0,780**	0,064	-0,899	0,092		
	E3. textoPadrao	**0,710**	0,077	-2,617	0,255		
	E4. doisPontosLegenda	**0,869**	0,085	-1,699	0,136		
	E5. pontoFinalSentenca	**1,718**	0,115	-1,249	0,064		
	E6. tamanhoMaximoTextoBotao	**1,086**	0,072	-0,472	0,062	0,897	0,080
Cores (C)	C1. sistemaDeCores	**1,575**	0,105	0,584	0,047	1,197	0,066
	C2. contrasteEntreTextoFundo	**1,556**	0,087	-0,465	0,047	0,062	0,043
	C3. coresMaterialDesign (MD)	**1,251**	0,122	1,881	0,136		
	C4. coresHarmonicas	**1,102**	0,088	-0,650	0,067		
Imagens (I)	I1. iconesMD	0,338	0,225	10,796	7,286		
	I2. pixelizacao	**0,985**	0,085	-0,038	0,069		
	I3. distorção	**2,017**	0,166	0,528	0,053		

Alguns itens apresentaram um parâmetro de inclinação baixo. Valores abaixo 0,7 no parâmetro de inclinação, indicam que o item tem pouco poder de discriminação, não diferenciando devidamente indivíduos com diferentes níveis de habilidade [Bortolotti et al. 2012]. Esse problema pode ser em consequência de os itens terem sido definidos de forma inadequada ou de a amostra não representar bem os diferentes níveis de habilidade. O item "I1. iconesMD", por exemplo, apresenta o valor mais baixo de

parâmetro de inclinação (a = 0,338), bem como valores do erro-padrão maiores do que os demais itens. Presume-se que os valores baixos de parâmetro de inclinação se devem ao conjunto de dados, que podem não representar bem o aprendizado desses conceitos, sendo possivelmente necessária uma amostra que represente melhor o aprendizado. Em relação aos parâmetros de dificuldade (*b2* e *b3*), todos os itens apresentam valores entre -4,0 e 4,0, com exceção também do item "I1. iconesMD", que apresentou um valor alto (*b2* = 10,796). Esse resultado pode ser explicado pela forma como o item foi definido, além de a amostra não representar bem o aprendizado do item.

3.2. Posicionamento na escala

Com base nos parâmetros de dificuldade estimados, os itens são posicionados numa escala (0,1), isto é, com média = 0 e desvio-padrão = 1 (Figura 1). A escala de habilidade é uma escala arbitrária na qual o que importa são as relações de ordem existentes entre seus pontos e não necessariamente sua magnitude.

Todos os itens considerados bem calibrados, com parâmetro de inclinação (*a*) acima de 0,7, são dispostos nos pontos da escala de acordo com os parâmetros de dificuldade (*b2* e *b3*) estimados (Tabela 2). Exemplificando, o "T3. tamanhoDaFonte" possui parâmetro *a* = 1,172 e *b2* = -0,115, portanto, foi posicionado no ponto 0 da escala (o ponto mais próximo de valor maior do que *b2*). Já o "E3. textoPadrao" com parâmetro *a* = 0,710 e *b2* = -2,617, foi posicionado no ponto -2,5 da escala (Figura 1).

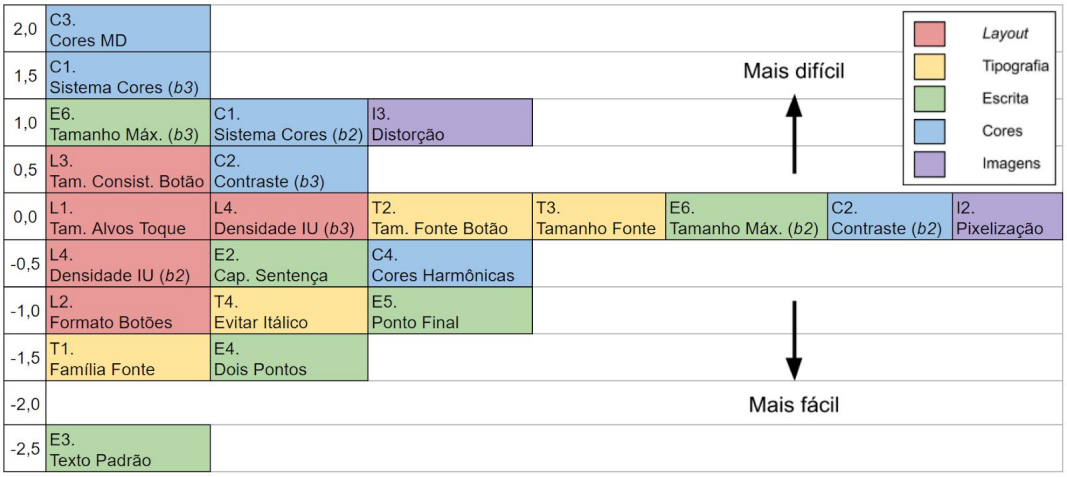

Figura 1. Posicionamento dos itens de design de UI na escala (0,1).

Cada um dos parâmetros *b2* e *b3* representa a habilidade necessária para que um indivíduo tenha probabilidade acima de 50% de satisfazer o respectivo item da rubrica. Assim, com base na escala (0,1), i.e., média 0 e desvio-padrão 1, um indivíduo com habilidade 2,0 (2 desvios-padrão acima da habilidade média) tem probabilidade maior do que 50% de satisfazer os itens da rubrica posicionados abaixo do ponto 2,0. Desta forma, a posição dos itens na escala sugere um sequenciamento baseado na dificuldade para o conteúdo de uma unidade instrucional para o ensino de design de UI.

Analisando os itens por categoria ou subdimensões (Figura 1), observa-se de forma geral que todas as categorias e subdimensão possuem itens espalhados desde níveis baixos (fáceis) até níveis altos (difíceis) da escala. Mesmo tendo uma leve indicação de que itens da subdimensão E (Escrita), por exemplo, parecem mais fáceis

do que itens referentes à subdimensão C (Cores). Principalmente a escolha de cores mais agradáveis, por exemplo, o item "C3. coresMaterialDesign", cuja definição é relacionada a cores propostas pelo *Material Design*, contém o mais alto parâmetro b estimado (b2 = 1,881) entre todos os itens posicionados na escala (Figura 1). Assim, esse item é posicionado no valor mais alto da escala. Uma das razões dessa dificuldade pode ser a sugestão de diversas cores saturadas, não agradáveis, no menu de cores do App Inventor, levando o aluno a escolher uma dessas cores ao invés de selecionar uma cor mais agradável de acordo com o *Material Design* (Figura 2). Da mesma forma, o item "E3. textoPadrao", cuja definição é relacionada à alteração do texto padrão de um botão, contém o menor parâmetro b estimado (b2 = -2,617). Isso pode ser explicado pelo fato de que, na tela de desenvolvimento da UI no App inventor, tipicamente após inserir um botão na interface, imediatamente altera-se o nome (texto padrão) para representar a ação pretendida do botão (e.g. confirmar, cancelar, voltar, etc.) (Figura 2).

Figura 2. Item mais fácil (E3. textoPadrao) e mais difícil (C3. coresMaterialDesign).

A dificuldade referente à subdimensão L (*Layout*) está mais próxima da média (Figura 1), indicando uma maior dificuldade em relação a consistência do design de elementos de UI, como o tamanho dos botões nas telas (L3. tamanhoConsistenteDeBotoes). Já a subdimensão T (Tipografia) é a única que apresenta todos os itens posicionados no ponto 0 (média) ou abaixo na escala (Figura 1). Isso pode ser explicado pelo fato de que a fonte padrão do App Inventor é a Helvetica, sem serifa, não italizada, e com tamanho 14. Sendo assim, caso o aluno não mude nenhum valor padrão relacionado à fonte, seu aplicativo estará automaticamente em conformidade com os itens definidos para a subdimensão T (Tipografia).

Figura 3. Ação relacionada a distorção de imagem ("I3. distorção").

Por outro lado, os dois itens da subdimensão I (Imagem) foram posicionados no ponto 0 ou acima na escala (Figura 1), especialmente o item "I3. distorção" cujo posicionamento ficou em 0,5. Tendo em vista que, por padrão, imagens adicionadas ao

App Inventor não são distorcidas, alguns alunos propositalmente distorcem as imagens para que preencham uma tela, p.ex., aumentando a altura da imagem (Figura 3).

4. Discussão

Geralmente, as unidades instrucionais voltadas ao ensino de computação com App Inventor não abordam o design de UI de forma sistemática e alinhada a princípios e diretrizes de design visual [Gomes e Melo 2013]. No entanto, a falta de qualidade do design visual da maioria dos aplicativos analisados revela a necessidade de que esses princípios sejam abordados [Solecki et al. 2020a]. Além disso, resultados de estudos pilotos integrando explicitamente conceitos de design visual no ensino de computação [Ferreira et al. 2019] apontam que integrar esses conceitos no ensino de computação pode ter um impacto positivo tanto na qualidade do design de UI de apps criados, quanto na motivação, experiência e aprendizado dos alunos. Porém, atualmente ainda não há uma clara definição de como deve ser esse sequenciamento, visto que as unidades instrucionais existentes não abordam o design visual de forma detalhada [Ferreira et al., 2019a]. Assim, os resultados da presente análise podem propor uma sequência de conteúdos mais fáceis (Tipografia), dentro da média (*Layout*) aos mais difíceis (Imagens) levando o aluno ao longo da progressão a aprender os principais conceitos básicos de design visual de UIs.

De forma geral, o App Inventor tem se mostrado eficaz para o ensino de criação de apps. Porém, vários estudos apontam que mesmo existindo aplicativos desenvolvidos com App Inventor com bom design visual, há também potencial para melhoria do ambiente para guiar o design visual mais alinhado às diretrizes de guias de estilo [Solecki et al., 2020a]. Essa falta de suporte em alinhamento com os guias pode também ter impactado o posicionamento de alguns itens como mais difíceis. Por exemplo, substituindo as cores do menu do App Inventor por cores propostas no *Material Design*, poderia impactar a dificuldade do item referente à seleção de cores agradáveis deixando-o mais fácil. A redução da quantidade das cores utilizadas pode também ser estimulada por um menu selecionando a cor primária e secundária sem demais alternativas. Desta forma, se todos os elementos de design de UI recomendados estiverem prontamente disponíveis no App Inventor, o aprendiz será direcionado a utilizar princípios e conceitos de design visual de maneira apropriada, criando assim aplicativos com uma melhor estética visual.

Ameaças à validade. Um risco ameaçando os resultados deste estudo está relacionado aos projetos da Galeria do App Inventor provenientes de vários contextos, sem informações sobre os criadores dos apps. No entanto, como o App Inventor é tipicamente usado por iniciantes no contexto educacional, esse conjunto de apps é considerado aceitável. Outra ameaça está relacionada ao uso de projetos de apenas um ambiente de programação. Nesse sentido, esse risco é minimizado usando uma amostra de tamanho satisfatório (quase 2 mil apps) permitindo a geração de resultados significativos. Para minimizar os riscos relacionados à validade, foi utilizada a rubrica CodeMaster que demonstra confiabilidade e validade [Solecki et al., 2020b]. A avaliação dos projetos do App Inventor foi automatizada pela ferramenta CodeMaster, reduzindo ainda mais riscos de confiabilidade que podem ocorrer numa avaliação manual. Com relação à metodologia da pesquisa, foi adotada uma metodologia

sistemática utilizando a abordagem GQM [Basili et al. 1994], e selecionadas técnicas estatísticas apropriadas para a análise.

5. Conclusão

Como resultado desse estudo analisando a dificuldade de objetos de aprendizagem relacionados ao design de UI de apps criados com App Inventor, foi identificado que itens relacionados a escrita, *layout* e tipografia são mais fáceis. Itens relacionados a imagens foram identificados com dificuldade mediana, e itens relacionados a cor foram identificados como os mais difíceis. Essas dificuldades observadas em relação a esses elementos de design de UI podem também evidenciar uma falta de suporte do ambiente App Inventor mais alinhado a diretrizes de design visual para facilitar o processo de aprendizagem de design de UI. Os resultados deste estudo podem ser usados tanto para discutir como melhorar sistematicamente o sequenciamento de conteúdo no ensino de design de UI no contexto do ensino de computação, quanto no desenvolvimento de diretrizes de currículo por meio da adoção de técnicas de *scaffolding*. Os resultados também indicam oportunidades de melhoria do ambiente App Inventor para apoiar de forma mais efetiva o ensino de design de UI.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001 e do Conselho Nacional de Desenvolvimento Científico e Tecnológico – Brasil (CNPq).

Referências

ACM/IEEE. (2013) Computer Science Curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science. Nova York, NY, EUA,

Agapie, E. & Davidson, A. (2018). Human-centered design charrettes for K-12 outreach. ACM Interactions, 25(6), 74-77.

AIGA. (2019). The Professional Association for Design. Disponível em https://www.aiga.org/

Alves, N. et al. (2019). Uma análise do sequenciamento pedagógico no ensino de computação na educação básica. Anais do Simpósio Brasileiro de Informática na Educação, Brasília/DF.

Basili, V. R., Caldiera, G., Rombach, H. D. (1994). The goal question metric approach. In Encyclopedia of Software Engineering, John Wiley & Sons.

Bortolotti et al. (2012). Avaliação do nível de satisfação de alunos de uma instituição de ensino superior: uma aplicação da Teoria da Resposta ao Item. Gestão & Produção, 19(2), 287-302.

Christensen, K. S. et al. (2016). Towards a formal assessment of design literacy: Analyzing K-12 students' stance towards inquiry. Design Studies, 46, 125-151.

CSTA. (2016). K–12 Computer Science Framework. Disponível em https://k12cs.org/

De Ayala, R. J. (2009). The theory and practice of item response theory. Guilford Press.

Ferreira, M. N. F. et al. (2019a). Learning user interface design and the development of mobile applications in middle school. ACM Interactions, 26(4), 66-69.

Ferreira, M. N. F. et al. (2019b). Ensinando design de interface de usuário na educação básica: um mapeamento sistemático do estado da arte e prática. Anais do Congresso Brasileiro de Informática na Educação. Brasília/DF.

Ferreira, M. N. F. et al. (2020). Ensinando design de interface de usuário de aplicativos móveis no ensino fundamental. Revista Brasileira de Informática na Educação, 28.

Gomes, T. C. & de Melo, J. C. (2013). App inventor for Android: Uma nova possibilidade para o ensino de lógica de programação. Anais dos Workshops do Congresso Brasileiro de Informática na Educação, Campinas, Brasil.

Gresse von Wangenheim, C. G. et al. (2018). CodeMaster – Automatic assessment and grading of App Inventor and Snap! programs. Informatics in Education, 17(1), 117-150.

Grover, S. & Pea, R. (2013a). Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42(1), 38-43.

Kumar, D. S., Purani, K. & Viswanathan, S. A. (2018). Influences of 'appscape' on mobile app adoption and m-loyalty. Journal of Retailing and Consumer Services, 45, 132-141.

Li, I., Turbak, F., Mustafaraj, E. (2017). Calls of the wild: Exploring procedural abstraction in App Inventor. Proc. of the IEEE Blocks and Beyond Workshop. Raleigh, NC, USA.

MEC. (2018) Base Nacional Comum Curricular. Disponível em: http://basenacionalcomum.mec.gov.br/images/BNCC_EI_EF_110518_versaofinal_site.pdf.

Morrison, G. R. et al. (2010). Designing effective instruction, 6th ed., John Wiley & Sons.

Papadakis, S. et al. (2017). The appropriateness of Scratch and App Inventor as educational environments for teaching introductory programming in primary and secondary education. Int. Journal of Web-Based Learning and Teaching Technologies, 12(4), 58-77.

Park, Y., Shin, Y. (2019). Comparing the effectiveness of Scratch and App Inventor with regard to learning computational thinking concepts. Electronics, 8, 1269.

Patten, J., Chao, C. I., Reigeluth, C. M. (1986). A review of strategies for sequencing and synthesizing instruction. Review of Educational Research, 56(4), 437-471.

Rabelo, M. (2013). Avaliação Educacional: fundamentos, metodologia e aplicações no contexto brasileiro. Rio de Janeiro: SBM - Sociedade Brasileira de Matemática.

Robinson, A., Pérez-Quinones, M. A. & Scales, G. (2015). Understanding the attitudes of African American middle school girls toward computer science. Proc. of Research in Equity and Sustained Participation in Engineering, Charlotte, NC, EUA.

Rogers, Y., Sharp, H. & Preece, J. (2011). Interaction design: beyond human-computer interaction (3rd ed.). Chichester: Wiley.

Samejima, F. A. (1969). Estimation of latent ability using a response pattern of graded scores. Psychometric Monograph, 34(4), 2-17.

Schlatter, T. & Levinson, D. (2013). Visual usability: principles and practices for designing digital applications. San Francisco: Morgan Kaufmann.

Solecki, I. et al. (2020a). Estado da prática do design visual de aplicativos móveis desenvolvidos com App Inventor. Revista Brasileira de Informática na Educação, 28.

Solecki, I. et al. (2020b). Automated assessment of the visual design of Android apps developed with App Inventor. Proc. of the 51st ACM Technical Symposium on Computer Science Education, Portland, USA.

Solecki, I. (2020c). Uma abordagem para a avaliação do design visual de aplicativos móveis criados com linguagens de programação baseadas em blocos. Dissertação (Programa de Pós-Graduação em Ciência da Computação) – Universidade Federal de Santa Catarina.

Turbak, F. et al. (2014). Events first programming in App Inventor. Journal of Computing Sciences in Colleges, 29(6), 81-89.

Xie, B., & Abelson, H. (2016). Skill progression in MIT App Inventor. Proc. of the IEEE Symposium on Visual Languages and Human-Centric Computing. Cambridge, Inglaterra.