Um modelo colaborativo para a aprendizagem do pensamento computacional aliado à autorregulação

Rozelma Soares de França[1], Patrícia Cabral de Azevedo Restelli Tedesco[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Recife – PE – Brasil

{rsf2, pcart}@cin.ufpe.br

Abstract. *The recent literature on computer science education in high school and assessment of learning emphasises the need to develop students' self-regulatory and problem-solving skills. Students must be trained so they can to set their own goals, monitor and evaluate their own learning. They must also be able to problems solving with the application of computational thinking and to carry out such activities in collaboration with their peers. In this light, this paper presents a model that aims at developing such skills, in the context of computational thinking learning. The model was built based on an exploratory study and its phases are presented in detail. We also discuss its strengths with respect to other related works found in the literature.*

Resumo. *A literatura recente sobre ensino de ciência da computação no ensino médio e avaliação da aprendizagem enfatiza a necessidade de se desenvolver as habilidades de autorregulação e resolução de problemas dos estudantes. Os aprendizes devem ser formados para que possam definir seus próprios objetivos, monitorar e avaliar a própria aprendizagem. Eles devem também ser capazes de resolver problemas com a aplicação do pensamento computacional e realizar tais atividades em colaboração com seus pares. Assim, este trabalho apresenta um modelo para o desenvolvimento de tais habilidades no contexto da aprendizagem do pensamento computacional. O modelo foi construído a partir de um estudo exploratório e suas fases são descritas, assim como seus diferenciais em relação a outros trabalhos correlatos encontrados na literatura.*

1. Introdução

A Computação permeia todas as atividades humanas, das artes às tecnologias, e hoje não se pode imaginar um cidadão ignorante do ponto de vista da Ciência da Computação (NUNES, 2008). O desenvolvimento de habilidades computacionais na educação básica é necessário por promover a capacidade de resolução de problemas, apoiar e relacionar-se com outras ciências, promover múltiplos caminhos profissionais futuros, dentre outros aspectos (SEEHORN *et al*., 2011). Assim, diversas iniciativas têm sido realizadas em escolas brasileiras e do exterior visando a disseminação do pensamento computacional, habilidade que baseia-se em fundamentos da Ciência da Computação.

Por outro lado, a formação em Computação apresenta diversos desafios, a exemplo das dificuldades enfrentadas por estudantes na aplicação de conceitos básicos de Programação já compreendidos (LAHTINEN *et al*., 2005). Além disso, outros problemas podem ser desencadeados pela inabilidade dos estudantes de estabelecer metas de estudo, em monitorar e refletir sobre sua própria aprendizagem, processos estes, associados à aprendizagem autorregulada (*Self-Regulated Learning* – SRL), considerada uma das competências-chave para iniciar e manter o aprendizado ao longo da vida (EU Council, 2002).

Neste contexto, como destacado no documento produzido no *APEC'2008 Education Reform Symposium in Xi'na, China*, os sistemas educacionais existentes devem ser modificados visando a integração das competências requeridas dos estudantes no século XXI, a saber: i) aprender ao longo da vida; ii) resolver problemas; iii) autogerenciar a aprendizagem e iv) trabalhar em equipe. Espera-se, com isto, que os estudantes aprendam a participar de forma apropriada em uma sociedade cada vez mais diversificada, utilizar as novas tecnologias e lidar com as rápidas mudanças nos locais de trabalho, algo que, indubitavelmente, está ancorado nos quatro pilares da educação apresentados pela Comissão Internacional sobre Educação para o Século XXI à UNESCO (DELORS, 1996).

Diante do exposto, esta pesquisa visa investigar de que forma competências requeridas dos aprendizes na atualidade podem ser exploradas no ensino médio no processo de desenvolvimento do pensamento computacional através da resolução e avaliação de problemas de programação com suporte computacional. Objetiva-se, com isto, contribuir para a educação em Computação, especificamente, com a promoção de uma habilidade necessária a todos, o pensamento computacional. Além disso, almeja-se corroborar com a formação dos estudantes em competências como a aprendizagem ao longo da vida, o gerenciamento da aprendizagem e a colaboração pela incorporação, ao modelo proposto, de estratégias autorregulatórias, como a autoavaliação, e uma visão de avaliação da aprendizagem centrada no aprendiz.

Assim, no restante do trabalho, serão detalhados tais aspectos da seguinte forma: nas Seções 2 e 3 serão apresentados o arcabouço teórico da pesquisa e trabalhos correlatos; na Seção 4 será descrita a metodologia empregada na pesquisa; na Seção 5 será apresentado um modelo colaborativo para o desenvolvimento do pensamento computacional aliado à autorregulação da aprendizagem; por fim, na Seção 6 a proposta será discutida, comparando-a com estudos relacionados e as considerações finais e trabalhos futuros serão apresentados.

2. Referencial teórico

2.1. Pensamento computacional

O pensamento computacional baseia-se em conceitos fundamentais da Ciência da Computação (WING, 2006). Ele é uma espécie de pensamento analítico e compartilha com a Matemática a resolução de problemas, com a Engenharia a concepção e avaliação de um sistema grande e complexo que opera dentro dos limites do mundo real e com a Ciência a compreensão sobre computabilidade, inteligência, a mente e o comportamento humano (WING, 2008). Ele tem influenciado a pesquisa em vários campos, a exemplo da Biologia com o algoritmo que acelera o sequenciamento do genoma humano (*ibid*.).

Segundo Wing (2006), a resolução de problemas com a aplicação do pensamento computacional requer a capacidade de pensar em vários níveis de abstração e não o simples uso de técnicas de programação. A autora ainda esclarece que o pensamento computacional é a maneira na qual os seres humanos pensam, e não os computadores, e que a partir dele são geradas ideais e não artefatos. Ele é uma habilidade fundamental para todos e não apenas para os cientistas da computação (*ibid*.). No estudo de algoritmos, conceitos como abstração, modularização e recursão podem ser aplicados às outras ciências, aumentando a capacidade de resolver problemas (NUNES, 2011). Assim, faz-se necessário o seu ensino na educação básica, uma realidade nos países desenvolvidos.

Analisando-se o currículo de computação para a educação básica definido pela *ACM Computer Science Teachers Association* (SEEHORN *et al*., 2011), em seu nível 2, na categoria *Computing Practice and Programming*, percebe-se uma estreita relação entre o seu objetivo 5 e aqueles preconizados pelos cursos introdutórios de Programação oferecidos em cursos técnicos e de graduação em Computação. No currículo mencionado, estão previstos conceitos como estruturas

condicionais e de repetição, expressões, variáveis e funções. Através do seu ensino objetiva-se envolver os estudantes no uso do pensamento computacional como um recurso para a resolução de problemas. Nesse sentido, deverá haver uma conexão entre esses conceitos e métodos de programação e os interesses dos aprendizes, com estímulo à percepção dos estudantes como solucionadores de problemas proativos, criadores, inovadores, capazes de mudar o mundo (SEEHORN *et al*., 2011). Por outro lado, ao analisar-se as disciplinas introdutórias de Programação, percebe-se um histórico de baixo rendimento e retenção dos estudantes. Assim, optou-se por redesenhar os processos de resolução e avaliação de problemas da disciplina pela integração de estratégias que serão descritas adiante.

2.2. Aprendizagem autorregulada (SRL)

Perspectivas contemporâneas de aprendizagem reconhecem que os estudantes controlam ativamente a sua própria aprendizagem e os seus resultados (HADWIN *et al*., 2011). Nesse contexto, a aprendizagem autorregulada refere-se a um planejamento intencional dos aprendizes, atrelado ao monitoramento de componentes cognitivos, comportamentais e motivacionais para a conclusão de um objetivo acadêmico (*ibid.*).

O processo de SRL envolve diversas fases compostas por processos que podem ajudar os estudantes em sua aprendizagem. Zimmerman (2002) propõe um modelo compreendido por um ciclo de três fases. A primeira delas é a *premeditação* que ocorre antes dos esforços para aprender e diz respeito ao estabelecimento de objetivos pedagógicos e a definição de um plano estratégico. Na segunda fase, também chamada de *controle volitivo*, é iniciada a realização da tarefa para atingir os objetivos previamente definidos. A última fase, a *autorreflexão*, ocorre depois dos esforços para a aprendizagem e refere-se à necessidade do estudante de avaliar o processo e seus resultados comparando-o com os objetivos delineados na primeira fase.

Nesse contexto, a autoavaliação é vista como uma atividade primordial por possibilitar que o estudante identifique a causa dos seus próprios erros e acertos. Todavia, se ele não consegue diferenciar com precisão o que sabe do que não sabe, dificilmente irá conseguir se envolver em atividades metacognitivas avançadas, tais como avaliar a sua aprendizagem de forma realista, ou planejar um efetivo controle da aprendizagem (TOBIAS; EVERSON, 2002). Desse modo, tornar-se necessário formar os aprendizes para que adquiram conhecimento sobre sua própria aprendizagem, bem como desenvolvam habilidades para gerenciá-la e regulá-la. Tal atividade pode ocorrer de forma independente, cooperativa ou colaborativa, conduzindo a mudanças no conhecimento, crenças e estratégias que os estudantes poderão transpor para novos contextos (HADWIN *et al*., 2011).

2.3. Avaliação como condição para aprendizagem

É fundamental encarar a avaliação e a aprendizagem como intimamente relacionadas, assumindo a avaliação como parte integrante do processo de aprendizagem (BORRALHO; FIALGO, 2012). Nesse sentido, Earl (2003) apresenta três abordagens do conceito de avaliação: avaliação *da* aprendizagem, *para* aprendizagem e *como* aprendizagem. De um modo geral, a avaliação *da* aprendizagem predomina nas escolas. Ela tem um caráter somativo e é destinada a certificar a aprendizagem dos estudantes no final do semestre, por exemplo. A avaliação *para* aprendizagem, no entanto, muda a ênfase da avaliação somativa para a formativa, investe no uso de instrumentos de avaliação diversificados e os resultados poderão ser utilizados pelos professores para fornecer *feedback* aos estudantes e ajudá-los a avançar na aprendizagem.

Na avaliação *como* aprendizagem, o estudante desempenha um papel mais ativo em suas atividades através de um processo crítico constante. Ela ocorre quando o aprendiz monitora a sua própria aprendizagem e usa o *feedback* deste monitoramento para fazer ajustes e adaptações na forma de aprender. Ela exige que o professor ajude o estudante a desenvolver, praticar e tornar-se

confortável com a reflexão e com uma análise crítica da sua própria aprendizagem. Neste cenário, a autoavaliação e a avaliação por pares podem assumir um importante papel no processo avaliativo (EARL; KATZ, 2006).

Segundo Boud (1995), a autoavaliação é uma habilidade necessária para a aprendizagem ao longo da vida. Além disso, ela tem um papel importante na efetivação da aprendizagem. Outros benefícios dessas duas abordagens de avaliação incluem: i) aumento da compreensão do próprio estilo de aprendizagem; ii) desenvolvimento de habilidades de avaliação crítica construtiva através do fornecimento de comentários a trabalhos de outros estudantes; iii) aumento da capacidade de dar e receber *feedback*; iv) trabalho colaborativo em prol da própria aprendizagem e dos pares.

3. Trabalhos relacionados

Na literatura há diversos estudos que versam sobre o desenvolvimento do pensamento computacional na Educação Básica. Scaico *et al*. (2012) relatam a experiência de uma olimpíada que objetivou apresentar a estudantes do ensino médio as principais estruturas de uma linguagem de programação e praticar técnicas utilizadas na construção de algoritmos, como é o caso do uso da abstração, depuração de erros, testes e melhoria de algoritmos simples. Mais recentemente, Barcelos e Silveira (2013) investigaram de que forma as competências relacionadas à Matemática são mobilizadas por estudantes do ensino médio no processo de desenvolvimento do pensamento computacional através da construção de jogos digitais. Outros trabalhos também debruçam-se sobre a apropriação do pensamento computacional na educação básica, como os descritos por de Souza *et al*. (2014) e França *et al*. (2014), e seus resultados trazem novas perspectivas para pesquisadores da Ciência da Computação e de outras áreas.

No que se refere à pesquisa sobre autorregulação, também há evidências de seus efeitos em diversas áreas, incluindo disciplinas de Computação. Bergin *et al*. (2005), realizaram um estudo que visou investigar a relação entre SRL e o desempenho acadêmico em um curso introdutório de Programação. Os resultados apontam que os estudantes que tiveram bom desempenho utilizaram mais estratégias metacognitivas e de gestão de recursos do que os estudantes com menor performance. Alhazbi *et al*. (2010), treinaram estudantes de Programação visando promover estratégias de SRL e os resultados indicam que as estratégias adotadas melhoraram a SRL dos estudantes na disciplina e possibilitaram um melhor desempenho acadêmico, se comparado àqueles que não participaram do treinamento. As evidências encontradas por Parham *et al*. (2010) também apóiam a metacognição como tendo um papel significativo na resolução de problemas de Ciência da Computação.

A autoavaliação da aprendizagem por iniciantes em Programação também tem sido alvo de investigações. Alaoutinen e Smolander (2010) elaboraram um instrumento de autoavaliação baseado na taxonomia de Bloom e os resultados sugerem que os estudantes conseguem mensurar o próprio conhecimento a partir dos níveis da taxonomia e percebem o instrumento como um recurso que pode ajudar na aprendizagem. Na pesquisa de Ngai *et al*. (2009), os exercícios de Programação foram projetados de modo a possibilitar a prática de autoavaliação. Os resultados mostram que há uma forte correlação entre a autoavaliação dos estudantes e avaliação realizada pelo professor, o que sugere que os estudantes são capazes de compreender os critérios de classificação e de avaliar a própria aprendizagem de modo preciso.

Sirotheau *et al*. (2011) propõem a incorporação, a um ambiente de aprendizagem de programação, de recursos que possibilitam a prática de *feedback* entre os estudantes como forma de estimular o desenvolvimento da autoavaliação. Nessa direção, a avaliação por pares tambem é investigada por Hamer *et al*. (2009), que analisaram a qualidade dos *feedbacks* fornecidos por estudantes em cursos introdutórios de Programação e seus resultados revelam que há uma

significativa correlação entre as avaliações dos tutores e as dos estudantes. Já Sitthiworachart e Joy (2004) discutem o uso de uma ferramenta web e evidenciam a necessidade de se ter critérios claros que ajudem os estudantes no processo de avaliação por pares.

4. Metodologia

Segundo Gil (2002), as pesquisas podem ser classificadas com base em seus objetivos e procedimentos técnicos utilizados. Nesse sentido, quanto aos objetivos, a presente pesquisa caracteriza-se como um estudo exploratório, por proporcionar maior familiaridade com o problema, tornando-o mais explícito, a partir de um significativo e relevante referencial teórico. Quanto aos procedimentos técnicos, ela é definida como pesquisa bibliográfica, pois foi desenvolvida com base em materiais já publicados, constituído principalmente de livros, teses e artigos científicos.

Assim como ocorre em outras modalidades, a pesquisa bibliográfia desenvolve-se ao longo de uma série de etapas. Desse modo, a partir da definição do tema e formulação do problema, um levantamento bibliográfico foi realizado. Procedeu-se com buscas automáticas nas bases de dados *ACM Digital Library*, *IEEE Xplore, SCOPUS* e *Elsevier (Science Direct)* e buscas manuais nos anais do Simpósio Brasileiro de Informática na Educação, Workshop de Informática na Escola, Workshop sobre Educação em Computação e Revista Brasileira de Informática na Educação. Além disso, livros, teses e dissertações foram selecionados.

Para inclusão de um estudo, ele deveria tratar de pelo menos um dos temas de interesse da pesquisa: pensamento computacional, programação para iniciantes, autorregulação da aprendizagem, metacognição, autoavaliação e avaliação por pares. Esta busca foi executada em diversas etapas e a cada iteração procedia-se com a leitura exploratória do material objetivando ter uma visão global dos estudos e sua utilidade para a pesquisa. Excluídos os estudos irrelevantes para este trabalho, procedeu-se com a leitura completa do material relacionando-o com o problema abordado nesta pesquisa. Atrelado a esta etapa, realizou-se fichamento e organização lógica do trabalho. Identificados os problemas na aprendizagem de programação para iniciantes e os princípios norteadores da autoavaliação e avaliação por pares, seguiu-se com a definição do modelo.

5. penC: um modelo para o desenvolvimento do pensamento computacional

5.1. Objetivo

Tendo em vista a importância de promover e estimular o pensamento computacional na educação básica e a necessidade de integração, aos sistemas educacionais, de competências como a aprendizagem ao longo da vida, a resolução de problemas, o autogereciamento da aprendizagem e o trabalho em equipe, um modelo foi definido neste trabalho, intitulado penC (pronuncía-se ‘pense’). Ele tem a intenção de criar condições adequadas para que estudantes do ensino médio desenvolvam habilidades e competências como as mencionadas, pensando sobre si mesmos (consciência metacognitiva) como solucionadores de problemas e refletindo sobre suas experiências contínuas de aprendizagem.

O penC foi concebido para ser integrado a ambientes de resolução de problemas de programação. Desse modo, não deve ser usado isoladamente. Em vez disso, deve ser acoplado a algum ambiente computacional, atuando como assistente no processo de aprendizagem do pensamento computacional. Com a adoção contínua do modelo espera-se que os estudantes sejam capazes de:

- Melhorar a precisão da avaliação do próprio conhecimento;
- Monitorar o processo de desenvolvimento do pensamento computacional;
- Avaliar crítica e construtivamente a solução dos pares, evidenciando os pontos fortes, diagnosticando equívocos e sugerindo melhorias ao solucionador do problema avaliado;

- Aceitar críticas à solução proposta e corrigir erros identificados;
- Trabalhar colaborativamente produzindo uma comunidade de aprendizagem no processo de avaliação por pares;
- Refletir sobre o próprio conhecimento a partir da revisão das soluções de seus pares aumentando a consciência da qualidade do próprio trabalho;
- Refletir e melhorar o próprio processo de resolução de problemas de programação a partir da autoavaliação do conhecimento e *feedback* provido pelos pares.

O modelo aborda principalmente as seguintes dificuldades normalmente enfrentadas por estudantes em contextos de aprendizagem (HARTMAN, 2001; TOBIAS, EVERSON, 2002; SEEHORN *et al*., 2011; WANG *et al*., 2012): i) avaliar o próprio conhecimento com precisão; ii) determinar a dificuldade de uma tarefa; iii) monitorar a compreensão do problema de forma eficaz; iv) resolver problemas com a aplicação do pensamento computacional; e v) perceber o erro como forma de aprender. Destes problemas, alguns têm natureza metacognitiva e outros são específicos do domínio de computação.

A consciência metacognitiva permite ao estudante planejar, sequenciar e monitorar a própria aprendizagem de modo a melhorar diretamente o seu desempenho acadêmico. Todavia, segundo Gama (2004), a metacognição é necessária, mas não suficiente para o sucesso acadêmico. O ponto mais importante é que por meio da prática da autorregulação os estudantes possam desenvolver o controle sobre a própria aprendizagem. Os professores podem aumentar a consciência e o controle sobre a aprendizagem dos estudantes, ensinando-lhes a refletir sobre como eles pensam, aprendem, lembram e realizam tarefas acadêmicas. Isto pode ocorrer antes, durante e após a execução de uma tarefa de aprendizagem.

5.2. Fases

O penC é constituído de quatro fases, conforme Figura 1.

Figura 1. Fases do modelo penC

(1) Pré-reflexão: esta fase antecede a resolução de um problema e constituída de duas atividades. A primeira delas é a reflexão sobre o estado atual do conhecimento tem como objetivo provocar a reflexão sobre o progresso do conhecimento do estudante em habilidades que devem ser desenvolvidas numa disciplina de Introdução à Programação. Nesse sentido, os resultados de aprendizagem pretendidos pelo Currículo de Computação da ACM e IEEE Computer Society (2013) foram considerados na elaboração do componente gráfico que auxiliará o estudante na

percepção do estado atual do próprio conhecimento (Figura 2a). Atrelada a este objetivo, está a pretensão de preparar o estudante para os próximos passos que ele deverá percorrer em seu processo de aprendizagem.

A segunda atividade é a autoavaliação da compreensão e dificuldade do problema que destina-se a fazer com que o estudante reflita sobre a compreensão do problema a ser resolvido, reconhecendo as metas e identificando os dados do problema, e sobre sua confiança para resolvê-lo corretamente. Perguntas como "Você identificou o objetivo deste problema?"; "Quão difícil você acha que é resolver este problema?" e "Você acha que consegue resolver este problema?" são apresentadas e devem ser respondidas pelo estudante antes de começar a tentar resolver o problema. Não há resposta certa ou errada e as informações prestadas serão utilizadas na última fase deste modelo.

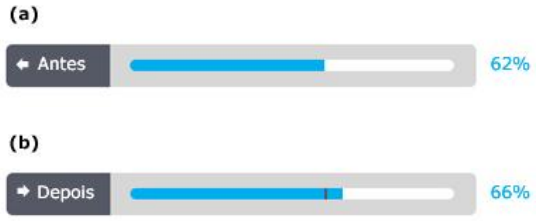

Figura 2. Progresso do estudante numa habilidade de Programação: (a) antes; (b) depois da resolução de um problema

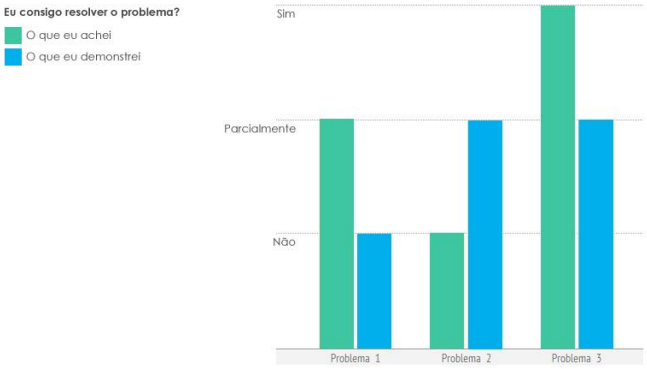

Figura 3. Comparação entre o desempenho esperado e o obtido

(2) ***Resolução:*** Nesta fase o estudante irá resolver o problema de programação proposto seguindo padrões de codificação (e.g. nomeclatura de variáveis). Ainda, ele deverá submeter sua solução antes do término do prazo definido pelo professor.

(3) ***Avaliação por pares:*** Nesta fase as soluções dadas pelos estudantes serão avaliadas por seus pares. Ela inicia-se com a definição dos avaliadores e atribuição dos códigos a cada um deles. Cada código é revisto por pelo menos 3 estudantes e antes de iniciar a avaliação cada um deles deve informar se compreendeu os critérios de análise. Esta atividade é importante, pois um entendimento impreciso dos critérios poderá interferir na forma de avaliação e, consequentemente, repercurtir na aprendizagem dos estudantes que receberão os comentários. Outro aspecto a este respeito é que o professor será notificado sobre aqueles avaliadores que não entenderam os critérios e poderá acompanhá-los no processo de avaliação das soluções.

A ficha de critérios para avaliação de códigos prevê uma análise quantitativa e qualitativa. Assim, para cada um dos critérios, o avaliador deverá assinalar um número numa escala entre 1 e 5, onde 5 representa a maior pontuação. Ele ainda deverá fornecer comentários que auxiliem os estudantes criadores das soluções a refletirem sobre seus erros e acertos. Nesse sentido, *scaffoldings* foram projetados para auxiliarem os avaliadores a elaborarem comentários que evidenciem os pontos fortes da solução, que apontem os erros e sugiram melhorias, ajudando, assim, os criadores das soluções a corrigirem seus erros, quando houver, ou a repetirem boas práticas de programação em suas atividades enquanto solucionadores de problemas da área.

Ao final dessa etapa, cada avaliador poderá enviar a sua análise ao solucionador do problema e comparar a avaliação realizada com a dos demais avaliadores que analisaram a mesma solução. Durante todo o processo, o anonimato de autores e avaliadores é mantido. Os criadores das soluções que tiveram seus códigos avaliados poderão optar por compartilhar com a turma os

códigos e avaliações recebidas, abrindo espaço para discussão sobre os erros cometidos e comentários atribuídos à solução.

(4) Pós-reflexão: Esta fase visa envolver o estudante na avaliação e reflexão da experiência de resolução de problemas. É nesta etapa que ocorre o monitoramento do conhecimento do aprendiz na aprendizagem do pensamento computacional. Dois aspectos da capacidade de monitoramento são inferidos: precisão no monitoramento do conhecimento (*Knowledge Monitoring Accuracy* – KMA) definido por Tobias e Everson (2002) e o viés no monitoramento do conhecimento (*Knowledge Monitoring Bias* – KMB) definido por Gama (2004).

O KMA refere-se à habilidade do estudante em predizer como irá realizar uma tarefa de aprendizagem. Esta métrica reflete a consciência do conhecimento que o estudante possui e é calculada a partir da previsão de resolver um problema corretamente, ocorrida na Fase 1 deste modelo, e o desempenho do estudante na resolução do problema, verificado na Fase 4. De modo complementar, o KMB é calculado a partir do KMA e identifica o tipo de desvio do aprendiz possibilitando saber se ele é pessimista, aleatório ou otimista.

O mecanismo de inferência é aplicado a cada vez que um problema é resolvido e como resultado o modelo do estudante é atualizado. As informações desse modelo são então utilizadas para apoiar a reflexão do aprendiz sobre sua aprendizagem. Os resultados inferidos são apresentados de forma gráfica e textual, assim como Gama (2004) realizou em seu trabalho. Inicialmente um gráfico em colunas é exibido mostrando a previsão de desempenho do estudante junto com o desempenho que ele obteve nos problemas (Figura 3). Abaixo do gráfico, explicações textuais são expostas para provocar interpretações adequadas à ilustração. O estudante deve observar para as tendências para verificar se há melhorias no monitoramento do próprio conhecimento, interagir com os mecanismos da interface interpretando os resultados num processo de autoavaliação. Na tela seguida, os resultados do KMA e KMB do estudante são apresentados, referente ao histório de problemas solucionados. Esta atividade objetiva fomentar a precisão do monitoramento do conhecimento e também é realizada a partir de recursos gráfico e textual. Além disso, com os *feedbacks* fornecidos a partir dos resultados, objetiva-se motivar os estudantes em seu processo de desenvolvimento do pensamento computacional, principalmente quando ele avaliar imprecisamente o próprio conhecimento.

É nesta fase, também, que o estudante poderá visualizar detalhadamente os comentários realizados por seus pares na avaliação de sua solução e explicitar, a partir da análise dos resultados, o que errou e o porquê. Ele também deverá avaliar a qualidade dos comentários recebidos de seus colegas. Além disto, o estudante poderá visualizar um componente gráfico que exibe o desempenho nas habilidades a serem desenvolvidas na disciplina de Programação. Tal recurso é similar ao exposto na Fase 1, porém ele traz uma marcação, informando o estado do conhecimento do estudante antes de iniciar o problema atual. Por meio dela, será possível ao estudante perceber visualmente se avançou ou retrocedeu em cada habilidade após a resolução do problema corrente (Figura 2b).

6. Discussões e considerações finais

Tendo em vista a necessidade de disseminar o pensamento computacional na educação básica, esta pesquisa apresentou um modelo para o desenvolvimento dessa habilidade. Atrelado a isto, integrou-se estratégias associadas à autorregulação e avaliação como forma de mitigar problemas na aprendizagem de conceitos computacionais.

Ao comparar esta pesquisa com outros trabalhos dispostos na literatura, percebe-se alguns diferenciais. No trabalho de Ngai *et al*. (2009), por exemplo, os autores propõem um mecanismo de avaliação por pares que apoia a autoavaliação dos estudantes numa disciplina de Algoritmos e

Programação. Atrelado ao *feedback* colaborativo, há também a avaliação automática da solução proposta pelos estudantes. Todavia, ao analisar-se o *design* da ficha de critérios de avaliação disposta na ferramenta, não percebe-se mecanismos que auxiliem os aprendizes na elaboração de comentários que apoiarão a reflexão dos autores dos códigos.

Neste contexto, é importante destacar a inexperiência dos estudantes em participar ativamente do processo de avaliação, fazendo-se necessário, naturalmente, fornecer-lhes suporte nesta atividade. Assim, no modelo proposto buscou-se, por meio de *scaffoldings*, ajudar os avaliadores nessa tarefa de modo que elaborem comentários sugestivos, corretivos e que evidenciem os pontos fortes das soluções ajudando, assim, os programadores iniciantes a refletirem sobre os artefatos criados.

Além disso, a pesquisa de Long e Aleven (2013) evidencia a necessidade de se ter instruções explícitas que auxiliem os estudantes a autoavaliarem o conhecimento. Este aspecto também não foi percebido no trabalho de Ngai *et al*. (2009) e na presente pesquisa buscou-se suportar tal atividade antes e depois do processo de resolução de problemas. A avaliação da qualidade dos comentários recebidos pelos solucionadores de problemas e a percepção do erro como forma de aprender também surgem como contribuições desta pesquisa no contexto da aprendizagem do pensamento computacional.

Espera-se, através da adoção de estratégias autorregulatórias associadas à atividade de avaliação por pares no processo de desenvolvimento do pensamento computacional, contribuir com a formação dos estudantes do ensino médio em habilidades requeridas no século XXI. Em especial, à formação para a aprendizagem ao longo da vida, ao trabalho colaborativo e a capacidade de resolução de problemas com a aplicação do pensamento computacional. Como trabalhos futuros, pretende-se validar o penC verificando sua eficácia na promoção das habilidades almejadas.

Agradecimentos

Rozelma Soares de França é bolsista de mestrado pela CAPES.

Referências

ACM & IEEE Computer Society (2013). *Computer Science Curricula 2013 - Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*.

Alaoutinen, S., & Smolander, K. (2010). “Student self-assessment in a programming course using bloom's revised taxonomy”. In *Proceedings of the 15th ITiCSE*, 155-159.

Alhazbi, S., & Hassan, M. (2010). “Fostering Self-Regulated learning in Introductory Computer Programming Course”.

APEC’2008 Education Reform Symposium in Xi'na, China. (2008). *21st Century Competencies*. Disponível em: < http://bit.ly/1jLooj4 >. Acesso em: 01 de abr. 2014.

Barcelos, T. S. & Silveira, I. F. (2013) “Relações entre o Pensamento Computacional e a Matemática através da construção de Jogos Digitais”. In *Proceedings of XII SBGames*, São Paulo.

Bergin, S., Reilly, R., & Traynor, D. (2005). “Examining the role of self-regulated learning on introductory programming performance”. In *Proceedings of the 1st ICER*, 81-86.

Borralho, A., Fialho, I., & Cid, M. (2012). “Aprendizagem no ensino superior: relações com a prática docente”. *Ensino Superior - Inovações e qualidade na docência*. Porto, Portugal.

Boud, D. (2013). *Enhancing learning through self-assessment*. Routledge.

de Souza, C. S., Salgado, L. C., Leitão, C. F., & Serra, M. M. (2014). “Cultural appropriation of computational thinking acquisition research: seeding fields of diversity”. In *Proceedings of the 19th ITiCSE*, 117-122.

Delors, J. (1996). *Educação: um tesouro a descobrir. Relatório para a UNESCO da Comissão Internacional sobre Educação para o Século XXI*. São Paulo: Cortez.

E.U. Council, E. U. (2002). *Council resolution of 27 June 2002 on lifelong learning. Official Journal of the European Communities*, 9.

Earl, L. M. (2012). *Assessment as learning: Using classroom assessment to maximize student learning*. Corwin Press.

Earl, Lorna, & Katz, Steven (2006). *Rethinking classroom assessment with purpose in mind: assessment for learning, assessment as learning and assessment of learning*. Western Northern Canadian Protocol Assessment Document.

França, R. S. ; Ferreira, V. F. S.; Almeida, L. C. F. ; Amaral, H. J. C. (2014) "A disseminação do pensamento computacional na educação básica: lições aprendidas com experiências de licenciandos em computação". In *Anais do XXII Workshop sobre Educação em Computação (WEI - CSBC).*

Gama, C. A. (2004). *Integrating metacognition instruction in interactive learning environments.* Tese de Doutorado. University of Sussex.

Gil, A. C. (2002). *Como elaborar projetos de pesquisa*. 4. ed. - São Paulo: Atlas.

Hadwin, A. F., Järvelä, S., & Miller, M. (2011). "Self-regulated, co-regulated, and socially shared regulation of learning". *Handbook of self-regulation of learning and performance*, 65-84.

Hamer, J., Purchase, H. C., Denny, P., & Luxton-Reilly, A. (2009). "Quality of peer assessment in CS1". In *Proceedings of the 5th ICER*, 27-36.

Hartman, H. J. (2001). "Developing students' metacognitive knowledge and skills". In *Metacognition in learning and instruction*, Springer Netherlands. 33-68.

Lahtinen, E.; Ala-Mutka, K.; Järvinen, H.-M. (2005). "A Study of the Difficulties of Novice Programmers". In: *Proceedings do 10th ITiCSE,*. 14-18.

Long, Y., & Aleven, V. (2013). "Active Learners: Redesigning an Intelligent Tutoring System to Support Self-regulated Learning". In *Scaling up Learning for Sustained Impact* , 490-495. Springer Berlin Heidelberg.

Ngai, G., Lau, W. W., Chan, S. C., & Leong, H. V. (2009). "On the implementation of self-assessment in an introductory programming course". *ACM SIGCSE Bulletin, 41*(4), 85-89.

Nunes, D. J. (2008). "Licenciatura em Computação". Jornal da Ciência, 30 de Maio.

Nunes, D. J. (2011). "Ciência da Computação na Educação Básica". Jornal da Ciência. 09 de setembro.

Parham, J., Gugerty, L., & Stevenson, D. E. (2010). "Empirical evidence for the existence and uses of metacognition in computer science problem solving". In *Proceedings of the 41st SIGCSE*, 416-420.

Scaico, P. D., de Lima, A. A., Azevedo, S., Belo da Silva, J. B., Raposo, E. H., Alencar, Y., ... & Scaico, A. (2013). "Ensino de Programação no Ensino Médio: Uma Abordagem Orientada ao Design com a linguagem Scratch".*Revista Brasileira de Informática na Educação*, *21*(02), 92.

Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cunniff, D., ... & Verno, A. (2011). *CSTA K--12 Computer Science Standards*: Revised 2011.

Sirotheau, S., de Brito, S. R., da Silva, A. D. S., Eliasquevici, M. K., Favero, E. L., & Tavares, O. D. L. (2011). "Aprendizagem de iniciantes em algoritmos e programação: foco nas competências de autoavaliação". In *Anais do XXII Simbósio Brasileiro de Informática na Educação.*

Sitthiworachart, J., & Joy, M. (2004). "Effective peer assessment for learning computer programming". In *Proceedings of the 9th SIGCSE*, 122-126.

Tobias, S., & Everson, H. T. (2002). *Knowing what you know and what you don't: Further research on metacognitive knowledge monitoring*.

Wang, Y., Li, H., Feng, Y., Jiang, Y., & Liu, Y. (2012). "Assessment of programming language learning based on peer code review model: Implementation and experience report". *Computers & Education, 59*(2), 412-422.

Wing, J. M. (2006). "Computational thinking". *Communications of the ACM, 49*(3), 33-35.

Wing, J. M. (2008). "Computational thinking and thinking about computing". *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717-3725.

Zimmerman, B. J. (2002). "Becoming a self-regulated learner: An overview". *Theory into practice*, *41*(2), 64-70.