Um Framework de Estratégias de Autorregulação Contextualizado em Programação Introdutória

Deller James Ferreira[1], Dirson Santos de Campos[1], Anderson Cavalcante Gonçalves[1]
Instituto de Informática – Universidade Federal de Goiás (UFG)
Av. Esperança, s/n - Chácaras de Recreio Samambaia
74690-900 – Goiânia – GO – Brazil
{deller,dirson_campos}@ufg.br, andersontwoand@gmail.com

Abstract. *A aprendizagem autorregulada é definida como o grau em que os alunos são participantes ativos em sua aprendizagem quanto aos aspectos motivacional, comportamental, metacognitivo e cognitivo. Pesquisas recentes mostram, que a maioria dos novatos tem habilidades de autorregulação pobres, que estão associadas a resultados ruins em programação. Deste modo, neste trabalho é aplicado um método exploratório da literatura para contribuir com uma taxonomia de estratégias regulatórias, a fim de apoiar os professores de programação introdutória, na criação de cenários de aprendizagem para promover a autorregulação dos alunos.*

Resumo. *Self-regulated learning is defined as the degree to which students are active participants in their learning in terms of motivational, behavioral, metacognitive, and cognitive aspects. Recent research shows that most beginners have poor self-regulation skills, which are associated with poor programming results. Thus, in this work, an exploratory method of the literature is applied to contribute with a taxonomy of regulatory strategies, in order to support teachers of introductory programming, in the creation of learning scenarios to promote self-regulation of students.*

1. Introdução

Os cursos introdutórios de programação apresentam muitos desafios para os alunos, já que um grande número de alunos entra em programas de ciência da computação com pouca ou nenhuma experiência prévia na disciplina. Os alunos são obrigados a dominar uma ampla gama de habilidades que eles continuam a lutar para dominar, tanto em termos de desenvolvimento de habilidades de programação, quanto em termos de consciência e domínio do processo de desenvolvimento de software [Falkner et al. 2014]. Embora existam várias iniciativas para melhorar o ensino e aprendizagem da programação introdutória, o panorama ainda é negativo [Mourão et al. 2019].

Para melhorar o aprendizado dos alunos nos cursos de programação, é importante mantê-los motivados, atentos ao seu progresso e engajados cognitivamente e socialmente. Portanto, para ter sucesso, os alunos muitas vezes precisam se tornar pensadores autoconscientes e sistemáticos de forma independente, enquanto desenvolvem e refinam estratégias para compreender e manipular novos conceitos abstratos em uma linguagem que provavelmente nunca viram antes [Loksa 2020].

Os professores podem ajudar os alunos nos processos de regulação, identificando e articulando estratégias de aprendizagem autorreguladas bem-sucedidas para contextos de programação específicos. Portanto, no ensino de ciência da computação, as abordagens de aprendizagem não priorizam as habilidades alinhadas à autorregulação [Pedrosa et al. 2019]. Os alunos que tentam aprender a programar nem sempre recebem treinamento ou suporte explícito para desenvolver as habilidades mentais necessárias para a programação [Loksa 2020].

Revisões recentes da literatura sugerem que a aprendizagem regulada é tópico de grande interesse na pesquisa em educação em computação, mas existem poucas teorias, modelos ou instrumentos específicos ao contexto de programação [Prather et al. 2020; Szabo et al. 2019; Malmi et al 2019]. As estratégias de autorregulação para programação ainda não são bem compreendidas por pesquisadores ou educadores de ciência da computação, tornando difícil desenvolver com sucesso métodos para promover a aprendizagem de autorregulação.

Assim, é necessária a elaboração de um framework, que oriente os professores a facilitar as competências de autorregulação e a realçar a sua importância, com o intuito de apoiá-los no desenvolvimento de atividades de programação que deem destaque e apoiem o desenvolvimento de competências relacionadas ao ensino de programação.

2. Revisão da Literatura

A aprendizagem autorregulada é um tópico importante na educação, que envolve a regulação da motivação, o envolvimento, a cognição e a metacognição do aluno. A aprendizagem autorregulada é definida como o grau em que os alunos são participantes ativos em sua própria aprendizagem acadêmica com respeito aos aspectos motivacional, comportamental, metacognitivo e cognitivo [Pintrich 2000]. Bergin (2005) investigou a relação entre a aprendizagem autorregulada e o desempenho introdutório da programação e mostrou que a aprendizagem autorregulada é um preditor útil para o desempenho da programação. A autorregulação inclui habilidades como monitorar os próprios processos, refletir se o processo é bem-sucedido, monitorar a compreensão de conceitos importantes e identificar estratégias alternativas para resolver problemas [Loksa 2020].

Por exemplo, mesmo se alguém já estiver dominando o básico de uma linguagem de programação, fortes habilidades de autorregulação podem ajudá-lo a reconhecer que não tem um bom entendimento de como um loop for é executado em Python. Isso pode fazer com que eles aumentem seu entendimento antes de continuar a escrever ou revisar seu código. Ou, quando alguém está lutando para diagnosticar uma falha em seu programa, as habilidades de autorregulação podem ajudá-lo a reconhecer que está se debatendo e buscar orientação especializada sobre como diagnosticar o problema de forma mais produtiva. Pesquisas em programação mostram consistentemente que as habilidades de autorregulação estão fortemente associadas ao sucesso na resolução de problemas computacionais [Falkner et al. 2014].

Trabalhos recentes mostram, no entanto, que a maioria dos novatos tem habilidades de autorregulação pobres, que estão associadas a resultados ruins na programação [Hauswirth e Adamoli 2017]. Um exemplo de estratégia de autorregulação cognitiva para a solução de problemas é a interpretação do problema. Quando os alunos interpretam o problema de maneira imprecisa, eles provavelmente usarão estratégias

ineficazes ou não conseguirão resolver o problema. É relatado em estudos que os alunos muitas vezes são incapazes de identificar e articular o objetivo do problema, os requisitos/restrições e o resultado esperado. Em outras palavras, os alunos carecem de habilidades de autorregulação, especialmente relacionadas à compreensão de tarefas.

Segundo Schoeffel (2019), a motivação é o estímulo para o desejo de aprender algo ou de participar e ter sucesso no processo de aprendizagem. Em relação às estratégias de autorregulação motivacionais, Keller (2017) propôs quatro categorias que estão diretamente ligadas a ela: atenção, relevância, confiança e satisfação. A falta de regulação da motivação pode causar uma forte discrepância entre o potencial e o desempenho na aprendizagem. Isso explica por que alunos altamente qualificados podem apresentar desempenho ruim, enquanto alunos com menos potencial podem figurar entre os melhores.

Entre as estratégias de autorregulação comportamentais, como gerenciamento de esforço, gerenciamento de tempo e busca de ajuda provaram estar positivamente correlacionados com os resultados acadêmicos [Daradoumis 2021]. Estratégias de gerenciamento de esforço ajudam os alunos a concentrar sua atenção na tarefa que estão realizando e a usar seu esforço para alcançá-lo de forma eficaz. Durante esse processo, os alunos adquirem competências que os capacitam a lidar com o fracasso, persistir e superar as dificuldades. Para este fim, essas estratégias fomentam a motivação e o compromisso para cumprir seus objetivos, mesmo quando há problemas ou distrações. As estratégias de gerenciamento do tempo permitem que os alunos adquiram habilidades relacionadas ao estabelecimento de metas e prioridades, planejamento, automonitoramento, resolução de conflitos, negociação, atribuição de tarefas, negociação e resolução de problemas. Os alunos obtêm sucesso no gerenciamento do tempo se puderem maximizar o uso do tempo para facilitar o desempenho acadêmico, o equilíbrio e a satisfação. As estratégias de busca de ajuda envolvem processos de busca de ajuda de outras pessoas, por exemplo, professor, colegas ou outras fontes que facilitam a realização dos objetivos desejados em um ambiente de aprendizagem. Essas estratégias estão associadas ao envolvimento do aluno e podem ajudar os alunos não apenas a atender às suas necessidades imediatas de aprendizagem, mas também a melhorar seu desempenho, adquirindo conhecimentos e habilidades e aliviando dificuldades, o que, em última análise, melhora a compreensão, o desempenho e a subsequente independência.

Em relação às estratégias metacognitivas, a aprendizagem reflexiva ajuda os alunos a se tornarem mais conscientes do processo de aprendizagem e suas dificuldades [Chang 2019]. Quando os alunos fazem uma autorreflexão eficaz, eles analisam como aprenderam, como entenderam os objetivos do processo de aprendizagem e o que é necessário para criar condições para o sucesso. Também estimula o pensamento crítico dos alunos sobre suas habilidades e reflete sobre estratégias de melhoria do processo de aprendizagem, conscientizando-os das vantagens da aprendizagem no futuro e ajudando-os a desenvolver habilidades transversais [Chang 2019]. A interação entre compromisso, o autocontrole, a autonomia e a autodisciplina dos alunos permitem que eles regulem suas próprias ações para atingir seus objetivos de aprendizagem.

Por outro lado, a aprendizagem reflexiva fornece feedback aos professores, permitindo-lhes reajustar suas experiências e ferramentas pedagógicas. A utilização do diário reflexivo é uma técnica que reforça e estimula a reflexão sobre a componente

teórica e prática do trabalho. Nesse contexto, para ter sucesso, os alunos precisam do conhecimento disciplinar exigido, bem como desenvolver estratégias de autorregulação [Falkner et al. 2014]. O desenvolvimento de estratégias de autorregulação é vital para ajudar os alunos a alcançarem o sucesso. Um aluno autorregulado definirá seus objetivos, organizará seus recursos e, em seguida, administrará seu tempo de maneira eficaz, sem esse nível fundamental de metacognição, ele não pode direcionar seu conhecimento de maneira útil e construtiva.

Com relação ao ineditismo da proposta da presente pesquisa, o framework proposto aqui apresenta uma abrangência única na literatura, ainda não encontrada em outros trabalhos, integralizando 4 (quatro) fases, que englobam 4 (quatro) áreas para regulação. Loksa e Ko (2021) propuseram um referencial teórico para a autorregulação no contexto da programação, mas contempla apenas 6 (seis) fases: planejamento, monitoramento do processo, monitoramento da compreensão, reflexão sobre a cognição e autoexplicação. A abordagem de Caruso et al. (2011) se restringe a estratégias de planejamento, encorajando programadores novatos a desenvolver planos ao praticar exercícios de planejamento durante a aprendizagem de programação introdutória. Bellhäuser et al. (2016) desenvolveram uma plataforma de treinamento baseada na web visando a autorregulação, porém limitada ao estabelecimento de metas, planejamento, motivação e estratégias de autorreflexão dos alunos.

3. Metodologia

O método utilizado, nesta pesquisa, na construção do framework de autorregulação dos alunos foi o método proposto por Gibbons e Bunderson (2005). Assim, este estudo adota um método exploratório da literatura na elaboração do marco regulatório do processo. Gibbons e Bunderson (2005) identificam os modelos exploratórios como um meio de definir e categorizar agrupamentos e relacionamentos, fornecendo uma base para explicar e compreender os processos envolvidos nas abordagens de aprendizagem. Eles identificaram três características importantes a serem consideradas na produção do conhecimento: explorar, explicar e projetar.

A pesquisa exploratória busca definir e categorizar, identificando “o que existe e quais são os possíveis agrupamentos e relações entre o que existe” (Gibbons & Bunderson, 2005, p. 927). Com esse tipo de pesquisa de "história natural", Charles Darwin documentou as semelhanças e diferenças entre tentilhões e outros animais selvagens nas Ilhas Galápagos antes de desenvolver uma teoria cientificamente testável. Esta pesquisa identifica os padrões que se tornam a base para questões na pesquisa científica (explicar) ou a base para o desenvolvimento de artefatos e processos (design), mesmo que os mecanismos causais subjacentes possam não ser totalmente compreendidos. A pesquisa explicativa é muitas vezes rotulada como pesquisa científica; procura “explicar o porquê e explicar como”, especificamente através da “investigação experimental da causa”. A pesquisa em design descreve a estruturação intencional de artefatos e planos de intervenção para aumentar a probabilidade de resultados específicos. Durante esta etapa busca-se responder também se os resultados foram alcançados e descritas intervenções para validar os resultados.

Uma avaliação por especialistas foi utilizada para a avaliação formativa do framework desenvolvido esta pesquisa. 7 (sete) especialistas foram recrutados, e os critérios de seleção utilizados foram se o especialista é doutor e possuir no mínimo 5

(cinco) anos de experiência no ensino de programação introdutória. Os sete especialistas responderam a um questionário baseado no referencial de avaliação proposto por Kimmons et al. (2020). As respostas ao questionário foram 100% positivas para todos os especialistas, ou seja, todos especialistas responderam “SIM” para todas as questões. Kimmons et al. (2020) argumentaram que os critérios para a valorização de um modelo pedagógico não são puramente arbitrários, mas sim baseados em sistemas de valores estruturados que representam as crenças, necessidades, desejos e intenções dos professores em um determinado contexto. Kimmons et al. (2020) propuseram seis critérios para determinar a qualidade de uma estrutura educacional: clareza, compatibilidade, produtividade, papel, escopo e foco no aluno. A Tabela 2 descreve o questionário, no qual as duas respostas possíveis eram "sim" ou "não".

Tabela 2. Instrumento para avaliação do framework proposto nesta pesquisa.

Critério	*Questões*
Clareza	O framework é suficientemente simples, claro e fácil de entender, sem complexidades ocultas?
Compatibilidade	O Framework apoia as práticas educacionais existentes consideradas valiosas para os professores na programação introdutória?
Produtividade	Os processos de autorregulação presentes no framework estão ligados ao engajamento do aluno gerando esforços produtivos em programação introdutória?
Papel	O framework melhora a autorregulação do aluno durante seu aprendizado em programação introdutória?
Escopo	O framework é parcimonioso o suficiente para ignorar aspectos que não são úteis para os professores, mas abrangente o suficiente para orientar as estratégias do professor para a autorregulação durante a aprendizagem de programação introdutória?
Foco no estudante	O framework dialoga com as habilidades a serem desenvolvidas pelos alunos de programação introdutória?

4. Resultados

A teoria de autoregulação que embasa esta pesquisa é a teoria desenvolvida por Pintrich (2000). O framework de autorregulação de Pintrich é o mais completo encontrado na literatura. De acordo com Pintrich (2000), as estratégias cognitivas são uma combinação de estratégias básicas e complexas para reter informações e incluem estratégias como ensaio (por exemplo, aprendizado mecânico), elaboração (por exemplo, conectar conhecimento prévio com novo material), organização (por exemplo, esboçar e escrever) e pensamento crítico (por exemplo, síntese e avaliação). As estratégias metacognitivas são aquelas que ajudam a regular e controlar a cognição para atingir um objetivo, e incluem estratégias como estabelecimento de metas, planejamento, automonitoramento e autorregulação. As estratégias de gerenciamento de recursos, por outro lado, ajudam os alunos a controlar recursos externos e incluem gerenciamento de tempo e ambiente (por exemplo, planejamento), regulação do esforço (por exemplo, manter o foco) e aprendizado entre colegas (por exemplo, usar colegas para entender de forma colaborativa) e buscar ajuda (por exemplo, perguntar para ajuda).

O modelo de Pintrich (2000) envolve o controle metacognitivo dos indivíduos sobre seus estados cognitivos, afetivos, motivacionais e comportamentais ao planejar, monitorar, avaliar e adaptar a aprendizagem, porém é genérico, o que demanda mais esforço por parte de um professor de programação introdutória ao utilizá-lo. Nesta pesquisa trazemos estratégias no contexto de programação para enriquecer e exemplificar o framework de Pintrich (2000) sobre a autorregulação na programação (Tabela 3).

Após uma extensa pesquisa bibliográfica, estratégias regulatórias foram selecionadas para integrar o framework. Na instanciação do framework no contexto de

programação, foram incluídos exemplos mais concretos de processos regulatórios importantes, relacionados ao aprendizado introdutório de programação. De acordo com Arab (2021), seguir estratégias de programação explícitas pode aumentar o sucesso na programação, reduzindo erros de compreensão, melhorando o sucesso da codificação e reduzindo o tempo de depuração.

O framework proposto neste trabalho integra, categoriza e justifica processos/estratégias regulatórias fragmentados de diferentes estudos na literatura científica, constituindo um recurso educacional abrangente para que os professores utilizem estratégias de engajamento do aluno em atividades regulatórias durante o aprendizado introdutório de programação e provê dimensões que servem como suporte para o desenvolvimento de atividades.

Tabela 3. Contextualização do Framework de Pintrich, contendo processos regulatórios e sua instanciação na aprendizagem de programação.

Fases	**Áreas para Regulação**			
	Cognitivo	Motivação/Afeto	Comportamento	Contexto
Fase 1. Previsão, planejamento e ativação **Planejamento antes da Codificação**	Definindo meta Avaliando conhecimento prévio sobre conteúdo Ativando metacognição **Detectando itens de problemas importantes antes da codificação.** **Desenhando a solução antes de codificar.** **Usando diagramas para explicar o design da solução e suas conexões com o código.** **Praticando e entendendo os fundamentos antes de começar a programar**.	Predispondo-se para alcançar o objetivo Julgando autoeficácia Percebendo da dificuldade da tarefa Percebendo o valor da tarefa Ativando o interesse **Vendo Obstáculos de uma Perspectiva Positiva na Programação**	Planejando tempo e esforço Planejando auto-monitoramento do comportamento. **Usando Kanban para planejamento de tarefas de programação.** **Utilizando Kanban para autoavaliação de conteúdos teóricos e práticos de programação.**	Percebendo a tarefa Percebendo o contexto **Estabelecendo estratégias para executar e monitorar tarefas de programação.**
Fase 2. Monitoramento **Monitoramento durante Codificação e Teste**	Monitorando cognição e meta-cognição **Entendendo padrões de programação.** **Experienciando padrões de programação.** **Monitorando a resolução de problemas em programação.**	Monitorando da motivação e afeto **Monitorando a motivação em programação.**	Monitorando do esforço, uso do tempo, necessidade de ajuda Observando o próprio comportamento **Usando Kanban para monitoramento de tarefas de programação.**	Monitorando a mudança de tarefas e condições de contexto **Monitorando o contexto de programação individual.**
Fase 3. Controle **Codificação e Teste**	Selecionando e adaptando estratégias cognitivas para aprender, pensar **Adaptando padrões de programação.** **Combinando padrões de programação.**	Selecionando e adaptando estratégias para gerenciar a motivação e o afeto. **Reduzindo a Ansiedade em programação.**	Aumentado/diminuindo o esforço Persistindo/desistindo Comportamento de busca de ajuda **Usando Kanban para gerenciamento de tarefas em programação**	Alterando ou renegociando tarefas Alterando ou abandonando o contexto **Atuando no contexto individual de programação**.
Fase 4. Reação e Reflexão **Reflexões sobre a Codificação do Programa**	Julgamentos cognitivos Pensamento crítico e metacognição **Aprendendo com os erros e acertos na programação.**	Reações afetivas Objetivos intrínsecos e extrínsecos, valor da tarefa, crenças de controle, autoeficácia e teste de ansiedade. **Refletindo sobre a motivação do aluno na programação.**	Comportamento de escolha Regulação do esforço Busca de ajuda Tempo/ambiente de estudo **Usando Kanban para refletir sobre tarefas de programação.** **Refletindo sobre os prós e contras do Kanban para o progresso individual na programação.**	Avaliação das tarefas Avaliação do contexto Aprendizado entre pares, tempo/ambiente de estudo **Refletindo sobre prós e contras na programação.** **Refletindo sobre o contexto de programação individual.**

O framework de processos/estratégias regulatórias é extenso e sua descrição completa não cabe neste artigo, desse modo elencamos algumas estratégias para uma explanação mais detalhada. A descrição do framework, na íntegra, pode ser acessada no link: https://dellerjames.github.io/.

Na fase de previsão, planejamento e ativação, a dimensão cognitiva envolve o estabelecimento de metas, a ativação do conhecimento prévio do conteúdo e a ativação do conhecimento metacognitivo. Como exemplo de processo regulatório contextualizado na programação, é importante que os alunos projetem a solução antes da codificação e também possam usar diagramas para explicar o desenho da solução e suas conexões com o código. De acordo com Falkner et al. (2014), os alunos devem entender e priorizar a estratégia de desenvolvimento do design da solução antes da codificação. A falta de compreensão de que o design é um processo indica que os alunos lutam para formar estratégias consistentes para auxiliar seu processo de programação.

Os alunos devem ser capazes de articular a importância de concluir um projeto antes de iniciar a implementação do código, representado como uma troca positiva entre gastar tempo inicialmente na fase de projeto para economizar tempo posteriormente na implementação do código e nas fases de teste. Uma estratégia que pode ser utilizada pelos alunos é usar descrições algorítmicas para representar o projeto que deve delinear as principais partes do problema: a incógnita, os dados, a condição. O desconhecido é o que é exigido, o que o aluno deve encontrar. Dados são as informações e variáveis fornecidas. O aluno deve perceber em que condições o desconhecido está ligado aos dados.

Na fase de monitoramento, com relação ao monitoramento cognitivo, um dos principais objetivos da aprendizagem autorregulada no contexto da programação de computadores é determinar como os alunos se auto monitoram e avaliam suas abordagens de planejamento e ajustam sua abordagem de programação com base nas informações recebidas com base no feedback obtido dos produtos criados (ou seja, completar vários elementos do espaço de tarefas e verificar bugs que surgem do exercício de codificação). O auto monitoramento fornece uma janela de conscientização sobre o funcionamento de uma pessoa. A autoconsciência pode ajudar a fazer as adaptações necessárias. O envolvimento com comportamentos metacognitivos, como reflexão e monitoramento da compreensão, está correlacionado com o sucesso nos cursos de programação. As estratégias de programação constituem um componente central da experiência em programação [Arab 2021]. Exemplos de questionamentos para o auto-monitoramento são: Os objetivos foram alcançados facilmente? Os objetivos foram alcançados com dificuldades? Os objetivos não foram alcançados? Há dificuldades em conteúdos teóricos de programação? Há dificuldades na prática de programação? Tenho dificuldade para entender os problemas? Tenho dificuldade para traçar objetivos para alcançar uma solução computacional para o problema? Tenho dificuldade para transformar os objetivos de solução em um programa? Tenho dificuldade para testar o programa? A maioria de meus programas contém muitos erros? Meus programas não funcionam para a maioria dos casos?

No que diz respeito à autorregulação da cognição na fase de controle, dois processos regulatórios relacionados a esta fase de controle, que contextualizados na programação, seriam as fases de codificação e teste, são a combinação de padrões de programação e a adaptação de padrões de programação. Pesquisas sobre teorias

cognitivas de aprendizagem de programação mostraram evidências de que programadores experientes armazenam e recuperam experiências anteriores de resolução de problemas que podem ser aplicadas a um novo problema e podem ser adaptadas para resolvê-lo [Johnson e Soloway 1984]. Por outro lado, um programador iniciante não tem experiência real, mas apenas as estruturas primitivas da linguagem de programação. Inspirada por essas ideias, uma estratégia para ensinar programação é apresentar pequenos trechos de código de programação para uma determinada finalidade, ao invés de deixar o aluno programar do zero. Esses trechos de código de programação para uma finalidade de programação específica para iniciantes são chamados de padrões de programação ou padrões de programação elementares [Wallingford, 2001].

Os padrões de programação podem ser vistos em diferentes contextos. Ver uma ideia em diferentes contextos e também ver as ideias em um quadro maior é uma forma de superar as barreiras conceituais. Considerar ideias em novos contextos é uma forma de obter insights sobre outros usos e significados possíveis. Esse tipo de atividade também está relacionado à flexibilidade do pensamento divergente. Guilford (1950) argumenta que a flexibilidade é um componente fundamental do processo criativo. Os alunos ampliam seus conhecimentos buscando relações com ideias fora de um determinado contexto, considerando novos escopos, limitações e restrições. Além disso, a correspondência de padrões está relacionada ao pensamento criativo, mais especificamente à habilidade de elaboração. A correspondência de padrões envolve a construção de um sistema mais complexo de ideias. A principal premissa nesse tipo de atividade é que arranjos inesperados e novos desencadeiam novas interpretações e ideias.

Na fase de premeditação, planejamento e ativação, a dimensão motivação/afeto abrange a adoção de orientação para metas, percepções de dificuldade da tarefa, julgamentos de autoeficácia e ativação de interesse e valor da tarefa. O desenvolvimento de boas habilidades de programação normalmente exige que os alunos pratiquem muito, o que não pode ser sustentado a menos que estejam adequadamente motivados (Law, 2010). A motivação pode afetar o aprendizado da programação de computadores. Nossos pensamentos têm implicações que afetam nossas emoções, motivação e realizações potenciais. Criar pensamentos positivos é uma daquelas habilidades que podem ser adquiridas quando intencionalmente perseguidas e consideradas em um nível de pensamento diferente daquele que produz pensamentos negativos. A conversa interna compassiva é como qualquer outra habilidade, devendo ser aprendida. Os alunos precisam ser ensinados a desenvolver consciência e estratégias para repensar sua fala interior, a fim de implementar com sucesso a capacidade de falar interiormente de maneira positiva. Como exemplos de discurso interior positivo na programação temos: “Quando um exercício de programação parecer difícil, estarei motivado a fazer melhor.”, “Um exercício de programação desafiador me motiva a trabalhar mais.”, “Estou interessado na tarefa proposta em aula, pois esta prática é importante para o meu desenvolvimento em programação.” e “Eu posso programar!”.

Na fase de reflexão na dimensão cognitiva, os alunos refletem sobre suas soluções e estratégias cognitivas de programação. Na programação introdutória, muito pode ser ganho dedicando um tempo para refletir e olhar para o que você fez, o que funcionou e o que não funcionou. Isso permitirá que você preveja qual estratégia usar para resolver problemas futuros. Mesmo alunos razoavelmente bons, quando têm a

solução para o problema e escreveram o argumento perfeitamente, fecham seus livros e procuram outra coisa. Ao fazê-lo, perdem uma fase importante e instrutiva do trabalho. Olhando para trás para a solução concluída, reconsiderando e reexaminando o resultado e o caminho que levou a ele, eles foram capazes de consolidar seus conhecimentos e desenvolver suas habilidades de resolução de problemas. Um bom professor deve compreender e impressionar seus alunos com a visão de que nenhum problema está completamente esgotado. Sempre há algo a fazer; com suficiente estudo e penetração, podemos melhorar qualquer solução e, em qualquer caso, sempre podemos melhorar nossa compreensão da solução.

Ainda com respeito à reflexão cognitiva, uma estratégia que pode ser usada para aprender com os erros e acertos é ir ao professor para descobrir qual foi o erro, porque a estratégia usada estava incorreta ou como melhorar a solução. No caso de alunos que "nem conseguiam errar", ou seja, que não sabiam por onde começar, o que fazer para superar suas dificuldades. O seguinte é um exemplo de uma lista de tarefas que os alunos podem seguir:

1. Solucionar individualmente os problemas computacionais exigidos pelo professor.

2. Relate suas limitações, erros e acertos ao professor.

3. Dialogar com o professor e refletir sobre as observações feitas pelo professor.

4. Reflita também sobre os diálogos que ocorreram entre o professor e os demais alunos.

5. Reflita individualmente sobre as perguntas: O professor o ajudou a melhorar ainda mais seu código? O professor o ajudou a terminar seu código incompleto ou corrigi-lo? O professor o ajudou a superar limitações? O que você aprendeu com seus erros? O que você aprendeu com os erros dos outros? O que você aprendeu com seus sucessos? O que você aprendeu com o sucesso de outras pessoas?

Com relação à dimensão afetiva na fase de controle, alunos em cursos introdutórios de programação geralmente têm uma percepção ruim de suas habilidades e ansiedade sobre os desafios que esperam enfrentar ao aprender a codificar. Estratégias de diálogo interno para gerenciar o afeto negativo e a ansiedade foram observadas por pesquisadores de ansiedade. Os alunos podem usar a estratégia de autoafirmação pela qual minimizam uma tarefa para proteger sua autoestima, especialmente se tiverem se saído mal na tarefa. Para manter o equilíbrio sem perder o compromisso, os alunos também podem usar a estratégia para ver e expressar o humor nas contradições e ambiguidades de seu processo de aprendizagem. Exemplos de autoafirmações que podem ser usadas pelos alunos são: Não devo me preocupar com notas ruins agora, devo pensar que se me esforçar mais, obterei melhores resultados no futuro! Eu não deveria pensar agora no programa que não recebi, mas focar no próximo problema! Nem sempre podemos escolher a música, mas podemos escolher a forma de dançar! Se eu trabalhar mais, vou conseguir. Eu não preciso acertar todas as vezes para me sentir bem! As coisas vão melhorar! Posso fazer um código de programa!

Quanto à dimensão comportamental, uma estratégia útil para a autorregulação do comportamento é elaborar um plano para buscar ajuda por precaução, por exemplo, enviar uma mensagem de texto para um amigo para saber se ele estará na biblioteca durante o estudo, à tarde, ou, por exemplo, enviar uma mensagem de texto para seu amigo no momento em que surgir a necessidade. O aluno também deve prever o acompanhamento do plano, onde cada plano, ainda não implementado, é monitorado

para ver se vai acalmar suficientemente sua ansiedade. Caso contrário, uma adaptação pode ser construída. Kanban pode ser usado como uma ferramenta simplificada e ágil para começar a estimular a autoaprendizagem em cursos introdutórios de programação por meio de planos de ação.

No que diz respeito ao tempo/ambiente de estudo os alunos podem avaliar e adaptar o tempo de aprendizagem. Como dito anteriormente, o método Kanban pode ser usado durante o aprendizado de programação para gerenciar esforços e objetivos. Alguns exemplos de processos reflexivos e questionamentos envolvidos no método Kanban são: Verifique se novas tarefas devem estar na coluna A FAZER. Verifique se os prazos das tarefas devem ser flexíveis. O Kanban me ajuda a focar no controle da quantidade de trabalho em andamento em um determinado momento? O Kanban me ajuda a entregar pequenas porções (subtarefas) da entrega maior (tarefa)? O Kanban melhora minha produtividade fazendo pequenas e contínuas mudanças no meu fluxo de trabalho? O Kanban pode me ajudar a maximizar meu trabalho produtivo e visualizar melhor o trabalho concluído, em andamento e incompleto? Se todos os itens foram atendidos, marque como APRENDIDO. Verifique, com base em suas respostas, se há necessidade de retornar à teoria, praticar mais ou recorrer ao professor. Atualize o Kanban após estudos extras.

5. Conclusões

Tomando como ponto de partida o framework de processos regulatórios de Pintrich, esta pesquisa se desenvolveu por meio de estudos de literatura, a partir de uma variedade de fontes de dados, para explorar os aspectos emocionais, cognitivos, comportamentais e contextuais da aprendizagem de programação introdutória. O framework elaborado nesta pesquisa foi avaliado, de modo qualitativo, positivamente, por 7 (sete) especialistas com relação aos aspectos clareza, compatibilidade, produtividade, papel, escopo e foco no aluno.

O framework proposto nesta pesquisa pode oferecer aos professores a oportunidade de organizar ambientes educacionais para facilitar e avaliar os alunos para obter experiências e aprender diferentes tipos de habilidades regulatórias de aprendizagem. Foi apresentado um framework que destaca as melhores práticas regulatórias usadas com o objetivo de melhorar a autorregulação durante o aprendizado introdutório de programação. O framework estendido aqui apresentado tem implicações para a prática, sendo especialmente benéfico para facilitar e avaliar a autorregulação, dos alunos. Deste modo, este estudo possui o potencial de fornecer informações valiosas para educadores sobre design instrucional e seleção de processos regulatórios apropriados para moldar e avaliar as atividades de aprendizado para facilitar a regulação dos alunos durante o aprendizado introdutório de programação.

No que se refere às limitações do estudo, acredita-se, mesmo estabelecendo critérios e parâmetros para análise dos dados, a avaliação subjetiva por um grupo pequeno com 7 (sete) especialistas, pode gerar vieses de interpretação de dados. Uma outra limitação é que o método exploratório usado para a detecção de estratégias regulatórias carece de maiores formalismos e sistematização. Como trabalho future, pretende-se refazer todo o processo, porém utilizando-se revisões sistemáticas da literatura, com as referências atualizadas, juntamente com análise de conteúdo da literatura.

Referências

Arab, M., Liang, J., Yoo, Y., Ko, A. J., and LaToza, T. D. (2021). HowToo: A Platform for Sharing, Finding, and Using Programming Strategies. IEEE. ISBN 978-1-6654-4592-4.

Bellhäuser, H., Lösch, T., Winter, C., and Schmitz, B. (2016). Applying a web-based training to foster self-regulated learning — Effects of an intervention for large numbers of participants. The Internet and Higher Education, 31, 87-100. https://doi.org/10.1016/j.iheduc.2016.07.002.

Bergin, S., Reilly, R., and Traynor, D. (2005). Examining the role of self-regulated learning on introductory programming performance. In Proceedings of the first international workshop on Computing education research (pp. 81-86). ACM.

Caruso, T., Hill, N., Van DeGrift, T., & Simon, B. (2011). Experience report: Getting novice programmers to THINK about improving their software development process. Paper presented at the SIGCSE'11 - Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, 493-498. doi:10.1145/1953163.1953307

Chang, B. (2019). Reflection in learning. Online Learning, 23(1), 95-110. doi:10.24059/olj.v23i1.1447

Daradoumis, T., Marquès Puig, J. M., Arguedas, M., et al. (2021). A distributed systems laboratory that helps students accomplish their assignments through self-regulation of behavior. Education Tech Research Dev, 69(4), 1077-1099. doi:10.1007/s11423-021-09975-6

Falkner, K., Vivian, R., and Falkner, N. J. G. (2014). Identifying computer science self-regulated learning strategies. In Proceedings of the 2014 conference on Innovation and technology in computer science education (pp. 291-296). ACM. doi:10.1145/2591708.2591715

Gibbons, A. S., and Bunderson, C. V. (2005). Explore, explain, design. In K. K. Leondard (Ed.), Encyclopedia of Social Measurement (pp. 927-938). New York, NY: Elsevier.

Guilford, J. P. (1950). Creativity. American Psychologist, 5, pp. 444–454.

Hauswirth, M., and Adamoli, A. (2017). Metacognitive calibration when learning to program. In Proceedings of the 17th Koli Calling International Conference on Computing Education Research (pp. 50-59). ACM.

Johnson, W. L. and Soloway, E. (1984). Proust: Knowledge-based program understanding. In Proceedings of the 7th international conference on Software engineering, Florida, United States, pages 369 – 380.

Keller, J. M. (2017). Motivation, learning, and technology: Applying the ARCS-V motivation model. Participatory Educational Research, 3(2), 1-13. doi:10.17275/per.16.06.3.2

Kimmons, R., Graham, C. R., and West, R. E. (2020). The PICRAT model for technology integration in teacher preparation. Contemporary Issues in Technology and Teacher Education, 20(1). doi:10.7916/cite.v20i1.4486

Law, K. M. Y., Lee, V. C. S., and Yu, Y. T. (2010). Learning motivation in e-learning facilitated computer programming courses. Computers and Education, 55(1), 218-228. doi:10.1016/j.compedu.2010.01.007

Loksa, D. (2020). Explicitly training metacognition and self-regulation for computer programming [Doctoral dissertation, University of Washington].

Loksa, D., and Ko, A. J. (2016). The role of self-regulation in programming problem-solving process and success. In Proceedings of the 2016 ACM Conference on International Computing Education Research (ICER '16) (pp. 83-91). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/2960310.2960334.

Malmi, L., Sheard, J., Kinnunen, P., Simon, and Sinclair, J. (2019). Computing education theories: What are they and how are they used? In Proceedings of the 2019 ACM Conference on International Computing Education Research (pp. 187-197). ACM. doi:10.1145/3291279.3339409

Mourão, A.B. and Netto, J.F.M. 2019. SIMROAA Multi-Agent Recommendation System for Recommending Accessible Learning Objects. In Frontiers in Education Conference (FIE) 2019 IEEE, pp. 1-9.

Pedrosa, D., Cravino, J., Morgado, L., and Barreira, C. 2019. Co-regulated Learning in Computer Programming: Students Co-reflection About Learning Strategies Adopted During an Assignment.

Pintrich, P. 2000. The Role of Goal Orientation in Self-Regulated Learning. PhD thesis, The University of Michigan, Ann Arbor, Michigan.

Law, K.M.Y., Lee, V.C.S., and Yu, Y.T. 2010. Learning motivation in e-learning facilitated computer programming courses. Computers and Education 55, 1 (2010), 218-228. https://doi.org/10.1016/j.compedu.2010.01.007.

Prather, J., Becker, B.A., Craig, M., Denny, P., Loksa, D., and Margulieux, L. 2020. What Do We Think We Think We Are Doing? Metacognition and Self-Regulation in Programming. In Proceedings of the 2020 ACM Conference on International Computing Education Research (ICER '20). Association for Computing Machinery, New York, NY, USA, 2–13. https://doi.org/10.1145/3372782.3406263.

Schoeffel, P. 2019. A Method to Predict At-Risk Students in Introductory Computing Courses Based on Motivation. PhD thesis, Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina.

Szabo, C., Falkner, N., Petersen, A., Bort, H., Cunningham, K., Donaldson, P., Hellas, A., Robinson, J., and Sheard, J. 2019. Review and Use of Learning Theories within Computer Science Education Research: Primer for Researchers and Practitioners. In Proc. of the WG Reports on Innovation and Technology in Comp Sci Education (Aberdeen, Scotland Uk) (ITiCSE-WGR '19). ACM, NY, USA, 89–109. https://doi.org/10.1145/3344429.3372504.

Wallingford, E. (2001). The Elementary Patterns Home Page. https://www.cs.uni.edu/~wallingf/patterns/elementary/