Anais dos Trabalhos de Conclusão de Curso
Pós-Graduação em Computação Aplicada à Educação
Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
Versão Preliminar

Análise do efeito de dicas especializadas de tutoria no ensino de introdução à programação

Maurício Vieira Dias Júnior[1], Seiji Isotani[2], Luiz Antonio Lima Rodrigues[3]

Resumo

Uma das dificuldades de ensinar programação para iniciantes condiz com a simplicidade das dicas emitidas pelos compiladores. A medição do efeito destas mensagens representam uma lacuna recorrente na área. Este presente estudo, tem como objetivo avaliar as Dicas Especializadas de Tutoria (DET) e as Dicas Padrões do Compilador (DPC), quanto ao desempenho discente na resolução de questões-problemas em Sistemas Tutores Inteligentes (STI). Foi realizado um experimento com discentes (n=42) do 1º ano do ensino técnico de informática, respondendo as questões "SomaDosQuadrados"e "Média3NúmerosInteiros". Como resultados obteve-se que não houve diferença estatisticamente significante a depender do tipo de dica usado, quanto as métricas observadas. Conclui-se na necessidade de mais estudos, a partir da investigação de outras métricas.

1. Introdução

Em uma recente revisão sistemática [Luxton-Reilly et al. 2018] constatou-se que a literatura sobre o ensino de introdução à programação (EIP) vem crescendo gradativamente, há mais de uma década em diversas frentes, seja pela perspectiva de aprimoramento discente, docente, curricular e/ou avaliação no processo de ensino-aprendizagem. Com base nos resultados coletados pelos autores, entre tendências/evidências com o foco docente, desde o ano de 2010 há lacunas notáveis no quesito de avaliação de ferramentas (compiladores/interpretadores) na área de depuração e erros de código-fonte.

Embora haja muitas publicações referentes ao desenvolvimento de ferramentas para o ensino de programação, não há muita atenção ao aspecto de avaliação destas ferramentas, especificamente, no tocante a eficiência das mensagens de erro aprimoradas (para além das mensagens padrão) [Luxton-Reilly et al. 2018], sendo raros os estudos empíricos que mostram evidências acerca dos efeitos das mensagens de erro para os discentes [Denny, Luxton-Reilly e Carpenter 2014].

Mesmos nestes raros estudos, que tem o intuito de comprovar se há algum efeito (positivo/negativo/neutro), não encontra-se convergência de resultados, conforme é relatado

[1]Pós-Graduando(a) em Computação Aplicada à Educação, USP, mauriciodias.junior@usp.br
[2]Orientador, USP, sisotani@icmc.usp.br
[3]Coorientador, USP, lalrodrigues@usp.br

Cite as: Junior, M. & Isotani, S. & Rodrigues, L. (2020). Análise do efeito de dicas especializadas de tutoria no ensino de introdução à programação. Anais dos Trabalhos de Conclusão de Curso. Pós-Graduação em Computação Aplicada à Educação Instituto de Ciências Matemáticas e de Computação. Universidade de São Paulo.

indícios de efeito neutro [Denny, Luxton-Reilly e Carpenter 2014] e positivo [Becker et al. 2016]. Estes mesmos estudos sugerem que devem ser realizadas outras pesquisas, contendo mais detalhes e aprofundamento, buscando evidenciar efeitos diretos e indiretos de mensagens de erro aprimoradas, portanto, constituindo-se em uma lacuna atual a ser amplamente investigada.

Diante deste cenário, identifica-se a relevância do tema para o aprimoramento das práticas pedagógicas docente no EIP, que tradicionalmente é altamente complexo [Robins, Rountree e Rountree 2003], sendo difícil por diversos fatores (assimilação da sintaxe, conhecimento matemático, ambientes de programação, instrução dos docentes entre outros) [Qian e Lehman 2017], os discentes se assustam facilmente, por não terem certeza de suas habilidades [Bruce 2018] e sofrem com a falta de instrução pessoal [Lahtinen, Ala-Mutka e Järvinen 2005], resultando, na maioria das vezes, em fracasso no processo de ensino-aprendizagem [Khouri, Santos e Barbosa 2020].

Em paralelo, observa-se também o potencial que os sistemas tutores inteligentes (STI) são capazes de agregar nestas ferramentas, como por exemplo, no fornecimento de mensagens, que podem ser com certo conteúdo especializado de tutoria em detrimento de mensagens padrões do compilador, sendo estas puramente técnicas.

O STI é um sistema projetado para interagir com discentes e docentes apresentando comportamentos inteligentes para adaptação e personalização da aprendizagem, sendo possível, a partir de seus recursos, oferecer auxílio passo-a-passo para proporcionar dicas de um determinado fragmento de conhecimento específico do modelo de domínio, denominado componente de conhecimento (CC), sendo viável trabalhar a sua competência para a resolução de uma tarefa por etapa [VanLehn 2006] [VanLehn 2011].

Torna-se relevante portanto, para o EIP, estudos com relato de experiências sobre o uso de ferramentas para este fim, conforme sugerido por [Luxton-Reilly et al. 2018], aprimorando-se na investigação de seus efeitos, complementado em [Qian e Lehman 2017], que juntos busquem ampliar as evidências sobre o impacto, identificando o efeito positivo/negativo/neutro destas ferramentas com dicas de tutoria, tudo isso em prol da melhoria da aprendizagem.

Aliando-se ao cenário apresentado, conforme identifica lacuna no tocante ao conhecimento sobre o impacto das dicas de tutoria em comparação com as dicas padrões, instiga-se para este presente estudo, a seguinte questão de pesquisa **[QP]**: *há diferença no desempenho de tarefas entre os tipos*[4] *de dicas usando STI para o EIP?*

Como forma de contribuição para a área, o presente estudo busca essa resposta tendo como objetivo principal, baseando-se no framework Goal/Question/Metric (GQM) [Basili e Rombach 1988]: **[OP]** ***analisar*** *os tipos de dicas* ***para fins de*** *comparação* ***com relação*** *ao desempenho discente na resolução de questões-problemas de algoritmos computacionais* ***do ponto de vista*** *docente* ***no contexto*** *dos discentes em EIP*.

Como objetivos específicos tem-se:

[OE1] *planejar e realizar experimento com discentes em STIs com os tipos de*

[4]Neste estudo foram adotadas as seguintes denominações para representar dois tipos de dicas: Dicas Especializadas de Tutoria (DET) e Dicas Padrões do Compilador (DPC))

dicas;

[OE2] *avaliar a proporção de etapas com solicitações de dica, mas sem respostas incorretas*;

[OE3] *avaliar proporção de etapas com respostas incorretas e solicitações de dicas*; e

[OE4] *avaliar o tempo médio/real gasto por problema*.

2. Fundamentação Teórica

Esta seção abordará os temas que embasam o presente estudo, fazendo um paralelo com a atual literatura, como os desafios presentes enfrentados pelo discente e docente no EIP, uma visão sobre o comportamento de um componente de STI denominado *inner loop* e abordando uma definição sobre os tipos de dicas utilizados.

2.1. Desafios discente e docente no EIP

Atualmente o EIP traz consigo níveis significativos de ansiedade e frustração para os discentes [Luxton-Reilly et al. 2018], ao mesmo tempo que constata-se níveis de engajamento muito baixos em comparação com outros componentes curriculares [Morgan et al. 2017]. Estudos anteriores revelam que são diversas as dificuldades experimentadas, seja pela falta de familiaridade com a sintaxe, linguagem natural, conhecimento matemático, modelos mentais imprecisos, falta de estratégias, ambientes de programação, instrução inadequada e conhecimento dos docentes [Qian e Lehman 2017].

Estes discentes geralmente relatam muitos déficits de conhecimento e estratégias para a resolução de problemas, sendo um processo extremamente complexo e difícil em obter êxito no planejamento de uma simples tarefa de programação [Robins, Rountree e Rountree 2003] e que muitas vezes não dispõem de recursos suficientes, a exemplo de ferramentas adequadas, penalizando a instrução [Lahtinen, Ala-Mutka e Järvinen 2005]. Como consequências destas dificuldades, essas questões estão sempre em ampla e constante discussão, pela considerável proporção nas taxas de evasão resultantes da falta de êxito nos cursos que oferecem EIP, o enfrentamento destes obstáculos tem sido o aumento e disseminação de ferramentas, investigando seus efeitos [Qian e Lehman 2017] [Bruce 2018] [Morgan et al. 2017].

Mesmo assim, essas ferramentas denominadas compiladores/interpretadores, utili-zadas pelos discentes iniciantes, continuam a apresentar, desde a década de 70, obstáculos na construção de tarefas, devido a forma em que as mensagens sobre o erro cometido são fornecidas, seja pela qualidade e/ou quantidade de mensagens, que quando colocadas de forma inadequadas e/ou limitadas representam o único feedback do discente, tornando difícil o seu progresso, por se tratar de um usuário ainda com pouca experiência, onde se poderia ter uma melhor orientação, mesmo apenas com a ferramenta, para conseguir corrigir o erro apresentado [Becker et al. 2016].

Faz-se portanto, necessária, por parte dos docentes, a construção de conhecimentos com materiais e abordagens bem elaboradas, tendo o discente como o principal ator de sua aprendizagem [Lahtinen, Ala-Mutka e Järvinen 2005], a exemplo dos recursos contidos nos STIs. Todas essas ações convergem no desafio do melhoramento de forma mais ampla

entre os atores: docente - com o investimento pedagógico de ferramentas para auxiliá-lo em sua prática; e discente - com o retorno de seu envolvimento na ferramenta para a obtenção do êxito em sua aprendizagem.

2.2. Feedback com Dicas no *Inner Loop* do STI

A instituição britânica *Education Endowment Foundation* (EEF) que desenvolve trabalhos independentes em prol de melhorias do desempenho educacional, elencou vertentes de ensino adotadas internacionalmente em discentes entre 5 a 16 anos, baseando-se em revisões e meta-análises. Pode-se constatar que a vertente feedback, que trata da informação dada sobre o desempenho do discente com relação aos objetivos ou resultados de aprendizagem, tem sido de maior impacto e baixo custo, demonstrando efeitos bastantes altos para a aprendizagem, sendo efetivo em todas as faixas etárias, possibilitando o avanço em até oito meses na aprendizagem em comparação com outras vertentes analisadas [EEF 2018].

Apropriando-se desta vertente, no contexto deste presente estudo, o recurso de dicas sobre uma determinada resolução de tarefas é uma forma de fornecer feedback, onde ambos representam o redirecionamento ou reorientação do discente para atingir uma meta pedagógica, alinhando o esforço e a atividade com um resultado. Questões sobre estratégias pedagógicas com o uso de tecnologias em provimento de tornar um sistema de computador tipo *coaching* robusto, amigável e inteligente possível para o discente complementar seus estudos fora da sala de aula [Burton e Brown 1979], são estudados há bastante tempo.

Atualmente, os STIs vem sendo desenvolvidos, inclusive com foco no EIP [Crow, Luxton-Reilly e Wuensche 2018], modelados [Lane e Vanlehn 2004] [Koedinger et al. 2013] e comparados em experimentos [VanLehn et al. 2005] [Aleven, V. and McLaughlin, E. A. and Glenn, R. A. and & Koedinger e R 2017], buscando evidenciar o aumento da eficácia no ensino por proporcionar granularidades de interação menores, diferenciando-se de outros sistemas aos quais são baseados apenas em respostas [VanLehn 2011].

O feedback com dicas para a próxima etapa/passo, de forma granulada, podem con-figurar em adaptações relevantes em um STI passando a representar diferentes estratégias para diferentes discentes [Aleven, V. and McLaughlin, E. A. and Glenn, R. A. and & Koedinger e R 2017].

Esse tipo de granulação de interação é baseado em etapas e sub-etapas, que correspondem respectivamente a *outer loop* e *inner loop* [VanLehn 2011]. Basicamente, em um STI, determinado CC é executado dinamicamente uma vez para cada tarefa tendo um comportamento de iteração (*outer loop*) e executado uma vez para cada etapa/passo do discente na solução de uma tarefa (*inner loop*) [VanLehn 2006].

Evidências refletem positivamente melhorias na aprendizagem com o uso de estratégias e erros do discente adaptadas com o *inner loop* [Aleven, V. and McLaughlin, E. A. and Glenn, R. A. and & Koedinger e R 2017]. Cabe portanto ao *inner loop* a funcionalidade de promover os denominados feedbacks: mínimo/imediato (genérico) e específicos com dicas em cada etapa/passo, avaliando o discente de quão competente é nos CC, extraindo evidências sobre o processo de aprendizagem, a fim de atualizar a evolução no modelo discente implementado no modelo pedagógico [VanLehn 2006]. Em

uma análise teórica e uma revisão da literatura empírica [Aleven et al. 2016] foi constatado que a ajuda sob demanda (dicas solicitadas pelo discente) é considerada útil e recomendada nas implementações dos STIs.

Consolida-se portanto nesta seção, a clarificação da importância do feedback/dica para o desempenho educacional, observando que esta vertente pode ser enriquecida com os recursos internos dos STIs, utilizados neste estudo, a partir da resolução dos CC, tratados com os tipos de erros de programação, principalmente com o dispositivo *inner loop*.

2.3. Dicas Padrões do Compilador (DPC) x Dicas Especializadas de Tutoria (DET)

Conforme já observado, as DPC representam um obstáculo para a realização das tarefas propostas pelos docentes, tornando a dica em nenhuma ou limitada ajuda ao discente para que possa sanar a sua falha. Ainda não há evidências comprovadas de quando e onde essas dicas deverão ser mostradas, tornando esta tarefa dos desenvolvedores/docentes mais uma arte do que uma ciência, porém, estudos vem sendo desenvolvidos tendo em vista a forma de tornar eficaz as dicas, seja no tocante de como serão produzidas, categorizadas e/ou apresentadas [VanLehn et al. 2005].

Um destes estudos [Suzuki et al. 2017] identificou cinco tipos de dicas docente para o EIP, baseando-se a partir de postagens de fórum entre docentes e discentes e uma entrevista semiestruturada com especialista, ampliando assim, relevantes perspectivas para geração de dicas de programação automáticas e pedagogicamente úteis, porém, ainda sem respaldos empíricos mais amplos, conforme afirmam seus autores.

Embora não seja especificamente para o EIP, trabalhando o sequenciamento das dicas, em outro estudo [D'Antoni et al. 2015], foi evidenciado que fornecer dicas é considerado útil, sendo possível aumentar o envolvimento do discente, diminuindo o tempo de finalização da tarefa em até 35%, sendo também constatado que os discentes que recebem feedback tem menor probabilidade de desistir da tarefa prática, identificando claramente os seus erros.

Resultados de outro estudo [Head et al. 2017] demonstram que exemplos de docentes e correções anteriores de erros discentes, que proporcionem correções sintetizadas, poderão ser úteis também para gerar feedback.

Uma prática padrão, também dos estudos com STIs, baseando-se na psicologia da memória humana, é mostrar dicas em sequência com o intuito de ampliar o aprendizado, do mais alto até o mais baixo nível. Este sequenciamento de dicas segue geralmente: 1) uma dica de apontar o erro (explicitando a localização do erro no código-fonte); 2) uma de ensino (declarar um conceito relevante); e 3) uma de baixo nível (ao qual é dada a resposta) [VanLehn et al. 2005].

O conjunto de dicas passo-a-passo, inicialmente da mais genérica, passando pelas específicas, até chegar na resposta, considera os passos anteriores, sendo possível ao discente checar todo o processo que ele percorreu, dando ao discente possibilidades de pensar o problema de forma conceitual e agindo de forma procedimental [VanLehn et al. 2005].

Constata-se na literatura [Aleven et al. 2006] também uma taxonomia comporta-mental de erro na busca de ajuda por dicas, a partir do julgamento dos discentes, definindo

categorias como *Help Abuse*, *Help Avoidance*, *Try-step Abuse*, *Miscellaneous Bugs* e suas sub-categorias.

Diante destes subsídios referentes ao contexto de produção, categorização e apresentação das dicas, fica considerado neste presente estudo, as DET como mensagens que atendem alguma perspectiva, mesmo que não comprovadas cientificamente, de es-pecialistas no EIP para cada CC abordados no código-fonte de uma questão-problema, em detrimento das DPC que são, na maioria dos compiladores tradicionais, apenas uma mensagem de erro que não conseguem ser convertidos em ajuda eficiente para o discente.

3. Trabalhos Relacionados

No estudo empírico mais recente encontrado [Denny, Prather e Becker 2020], foi realizada e avaliada uma formalização na abordagem de construção de mensagens, que até então era difusa, com uma coleção mais recente de diretrizes publicadas [Becker et al. 2019]. O experimento (n=718) controlado foi realizado utilizando uma tarefa de depuração dividindo os participantes aleatoriamente em 2 grupos. Um programa na linguagem C de 20 linhas foi criado, contendo quatro linhas com erros de sintaxe, que ao ser enviada pelo grupo de controle apareceria as mensagens padrão do compilador, já no grupo experimental apareceria mensagens aprimoradas propostas no estudo, sendo registrados a hora, o código real e uma classificação do tipo do erro gerado. Como resultado os discentes que se depararam com as mensagens aprimoradas efetuaram menos envios em uma quantidade menor de tempo para terminar uma tarefa em detrimento com mensagens padrão de um compilador, além de expressarem a utilidade das novas mensagens em detrimento com as mensagens padrão do compilador.

Diferente dos estudos anteriores, que trabalhavam com o código de uma linguagem de programação específica, [Marwan, Williams e Price 2019] exploraram o impacto das dicas no EIP com um ambiente de programação tipo blocos. No experimento (n=201) controlado cada participante resolveu 2 tarefas de programação (15 minutos cada com no máximo 7 dicas), divididos em três grupos (63 - sem dica, 79 - com dicas de código com explicações textuais e 59 - com dicas de código com instruções de autoexplicação). Embora não tenha sido em um contexto de sala de aula, entre outros resultados, os autores relataram que as dicas de código com explicações textuais ampliam o desempenho discente melhorando o seu aprendizado em tarefas similares futuras.

Baseando-se nos estudos de [Denny, Luxton-Reilly e Carpenter 2014] e [Becker et al. 2016] em [Pettit, Homer e Gee 2017] um experimento (n=?) foi realizado com histórico de submissões de tarefas de programação de 8 semestres com duas versões (com e sem dicas melhoradas) da mesma ferramenta, esperava-se que houvesse diminuição na quantidade de vezes de submissão de tarefas pelo discente que continham dicas melhoradas, porém o encontrado foi que não há nenhum benefício mensurável entre ambas as versões da ferramenta, embora haja

Já em [Becker et al. 2016] o experimento (n>200) realizado contou com mais de 50.000 mensagens de erro. O grupo experimental que recebeu apenas uma mensagem aprimorada por compilação (com o intuito de não confundir os discentes em comparação com as mensagens padrão do compilador já trabalhado), exibiu menos variância com um perfil de erro mais homogêneo em comparação com o grupo controle, que tinha as

mensagens de erro padrão do compilador. Também os discentes relataram que a experiência com as mensagens aprimoradas foram menos frustrantes, que tornam o aprendizado de programação mais fácil e que recomendariam para outro discentes. Os resultados do estudo mostraram redução na frequência de erros: gerais, por discente e repetidos por mensagens do compilador. Foram identificadas 8 mensagens de erro mais frequentes que se melhoradas podem ter efeitos estatisticamente relevantes.

Em [Antonucci et al. 2015] o sistema gera dicas incrementais diretamente no código-fonte em um ambiente *Massive Open Online Courses* (MOOC) de forma flexível (manual ou automática), sendo possível, quando necessário, intervenção docente. As dicas criadas antecipadamente com base no código-fonte, podem ser solicitadas pelo discente a medida que for necessária. No experimento (n=38) não-controlado realizado, foi observado que os discentes que tiveram submissões corretas sem o uso de dicas foram 99,5% e com dicas 63%, sendo possivelmente explicado, segundo os autores, que estes discentes precisariam naturalmente de mais dicas, pois tiveram 37% de submissões incorretas com o uso de dicas. Embora os dados quantitativos tenham sido desfavoráveis no tocante a relevância no uso das dicas, os resultados qualitativos obtido pelo questionário com 38 discentes, sugerem que o sistema de dicas foi útil (73%) na orientação dos discentes que necessitavam de auxílio para o desenvolvimento da tarefa, fornecendo dicas com granuridades satisfatórias (80%).

No estudo de [Denny, Luxton-Reilly e Carpenter 2014] foi realizado um experimento (n=83) onde os participantes fizeram 10 exercícios sobre expressões, condicionais e métodos das classes java.lang.String e java.lang.Math. Segundo os autores, os resultados encontrados foram surpreendentes quanto a eficácia destas mensagens no grupo experimental, pois demonstraram que não houve efeito/benefício significativo. O módulo criado gera o feedback instantâneo sobre o erro de sintaxe de forma mais descritiva/ilustrativa, contendo dois fragmentos lado a lado (um mostrando o erro de sintaxe e o outro mostrando uma versão corrigida do código destacando as diferenças de sintaxe). Foram identificados 53 tipos de erros de sintaxe, sendo estes classificados em 9 categorias.

Na tabela 3.1 há um comparativo sintetizado dos seis trabalhos acima relacionados, mostrando as suas principais limitações perante este presente estudo. De uma maneira geral, estes estudos não contemplaram em suas análises o processo de verificação por etapas, conforme abordado e contemplado por este presente estudo, sendo apoiado pelo recurso inerente dos STIs.

Observa-se que nos trabalhos empíricos relacionados, houve distinção no tocante a linguagem de programação utilizada, [Becker et al. 2016] e [Denny, Luxton-Reilly e Carpenter 2014] trabalharam com Java, já [Antonucci et al. 2015] utilizou Eiffel, [Marwan, Williams e Price 2019] fez uso de programação em blocos (tipo scratch), [Denny, Prather e Becker 2020] usou C e [Pettit, Homer e Gee 2017] utilizou C++. Neste estudo foi realizado em pseudocódigo, sendo este mais próximo da linguagem natural dos participantes.

Verifica-se que a linguagem de programação por si só já se traduz em uma barreira quando o discente não é nativo do idioma, como a grande maioria são desenvolvidas em inglês, não é tão simples a sua imediata compreensão por discentes de EIP iniciantes que não dominam esta língua, algo que este presente estudo buscou solucionar. Embora todos esses trabalhos apresentaram-se como propostas para o EIP, apenas [Marwan, Williams

Tabela 3.1. Comparativo dos trabalhos relacionados

Autor(es) (ano)	Ferramenta	O que foi analisado?	Efeito (Abordagem(ns))	Principais Limitações
Denny et al. (2020)	Ferramenta online (C)	- Estimativa do tempo gasto por discente para interpretar tipo de mensagem e resolvê-los; - Se os discentes leram as mensagens de erro; - Quão úteis foram as mensagens de erro para ajudar os discentes a identificar e corrigir os erros.	Positivo (Quali-Quanti)	Código-fonte verificado por completo, sem ser por etapas.
Marwan et al. (2019)	iSnap (Blocos)	- Classificação da utilidade geral do ambiente em uma escala de 1 a 10; - Rastreamento de códigos completos; - Perspectivas dos discentes sobre a utilidade do iSnap com e sem receber dicas; - Aprendizagem, conforme medido pelo desempenho em tarefas futuras sem dicas.	Positivo (Quali-Quanti)	Verificação de códigos completos, sem ser por etapas.
Pettit et al. (2017)	*Front-end* C ++ para *GNU Compiler Collection* (GCC)	- Probabilidade de erros de compilação sucessivos; - Ocorrência de erros do compilador nos semestres; - O progresso do discente para a conclusão; - Nível de detalhe nas mensagens aprimoradas.	Neutro (Quali-Quanti)	Apenas utilizou os códigos completos das submissões, sem ser por etapas.
Becker et al. (2016)	Decaf (Java)	- Número de erros gerados em todas as mensagens padrões do compilador e não apenas o número de submissões não compiladas; - Medição dos erros não apenas ao completar os exercícios, mas todas atividades de programação do discente; - Frustração discente quanto as mensagens de erro padrão do compilador; - Nível de barreira que os erros do compilador quanto ao progresso das tarefas.	Positivo (Quali-Quanti)	Não verifica solicitação de dicas por etapas, além de não abordar a questão de tempo gasto em desenvolvimento para tarefas com e sem dicas.
Antonucci et al. (2015)	AutoTeach (Eiffel)	- Se o sistema de dicas era muito difícil ou fácil de usar; - Se os discentes que experimentaram o sistema de dicas acharam útil ou não; - Se o nível de granularidade das dicas era adequado.	Positivo (Quali)	Não utiliza a submissão do código-fonte por etapa, sem análise quanti, sem comparativo de tempo de resolução de questão com dica e sem dica
Denny et al. (2014)	Módulo no CodeWrite (Java)	- Número de submissões consecutivas não compiladas feitas durante a tentativa de uma determinada tarefa; - Número total de submissões não compiladas em todos as tarefas; - Número de tentativas necessárias para resolver os tipos de erros mais comuns.	Neutro (Quanti)	Submissões foram analisadas por completo, sem ser por etapas e sem controle de proporção de dicas solicitadas

e Price 2019] continha algo mais lúdico que poderia mitigar essa dificuldade, sendo nos demais apresentadas as implementações dos código-fontes originalmente.

Outro detalhe foi que o formato de apresentação das dicas aprimoradas no compi-lador também divergiu. [Denny, Prather e Becker 2020], [Becker et al. 2016] e [Denny, Luxton-Reilly e Carpenter 2014] apresentaram em locais similares as saídas dos com-piladores padrões, diferentemente de [Antonucci et al. 2015] que mostrou no próprio código-fonte e [Marwan, Williams e Price 2019] e [Pettit, Homer e Gee 2017] que apre-sentaram as dicas em uma janela a parte. Com o presente estudo a estrutura das dicas seguiram o formato padrão de um STI, que embora as dicas se posicionem em um local separado, se faz necessária a sua ativação por parte do usuário.

4. Metodologia

Diante do **[OP]** definido que foi **analisar** as DET e as DPC **para fins de** comparação **com relação** ao desempenho discente na resolução de questões-problemas de algoritmos computacionais **do ponto de vista** docente **no contexto** dos discentes em EIP, o desenvol-vimento do presente estudo, foi conduzido de acordo com as etapas do processo de um experimento [Wohlin et al. 2012]: **(a)** escopo - já descrito na seção de introdução com o GQM, **(b)** planejamento e **(c)** operacionalização que será exposto nesta seção e **(d)** análise e interpretação que serão mostrados e discutidos nas seções seguintes.

No que se refere a etapa de planejamento **(b)**, conforme sumarizada por [Wohlin et al. 2012], "como"o experimento foi desenvolvido, foram planejados os seguintes pontos:

- **Seleção do contexto:** *off-line*, sendo realizado em um ambiente preparado para o experimento, com discentes executando problemas reais e específicos sobre o assunto de estruturas sequenciais de algoritmos computacionais, envolvendo as seguintes duas questões-problemas: "SomaDosQuadrados"e "Média3NúmerosInteiros";
- **Seleção das variáveis independente e dependente:** tem-se a variável independente (mensagens de dica) com os seguintes tratamentos: com DET e com DPC. A variável dependente é o desempenho discente na resolução de tarefas;
- **Formulação das hipóteses:** foram feitas comparações das hipóteses com base nas métricas para DET e DPC. Foram denominadas as métricas **M1** (proporção de etapas com solicitações de dica, mas sem respostas incorretas), **M2** (proporção de etapas com respostas incorretas e solicitações de dicas) e **M3** (tempo médio/real gasto por problema (em segundos)), que qualificam o desempenho para a resolução das questões-problemas de algoritmos computacionais):

$$M1 \quad \begin{matrix} H_{0.1} : M_{1.DET} = M_{1.DPC} \\ H_{1.1} : M_{1.DET} > M_{1.DPC} \end{matrix}$$

Onde $\mathbf{H}_{0.1}$ a métrica M1 para DET e DPC deverão ser iguais (não há diferença significativa entre a proporção de etapas com solicitações de dica, mas sem respostas incorretas) e $\mathbf{H}_{1.1}$ a métrica M1 para DET deverá ser maior que a DPC (há diferença).

$$M2 \quad \begin{matrix} H_{0.2} : M_{2.DET} = M_{2.DPC} \\ H_{1.2} : M_{2.DET} < M_{2.DPC} \end{matrix}$$

Onde $\mathbf{H}_{0.2}$ a métrica M2 para DET e DPC deverão ser iguais (não há diferença significativa entre a proporção de etapas com respostas incorretas e solicitações de dicas) e $\mathbf{H}_{1.2}$ a métrica M2 para DET deverá ser menor que a DPC (há diferença).

$$M3 \quad \begin{matrix} H_{0.3} : \mathrm{M}_{3.DET} = \mathrm{M}_{3.DPC} \\ H_{1.3} : \mathrm{M}_{3.DET} < \mathrm{M}_{3.DPC} \end{matrix}$$

Onde $\mathbf{H}_{0.3}$ a métrica M3 para DET e DPC deverão ser iguais (não há diferença significativa entre o tempo médio/real (em segundos) gasto por problema) e $\mathbf{H}_{1.3}$ a métrica M3 para DET deverá ser menor que a métrica DPC (há diferença).

Sendo esperado para M1, M2 e M3 comparações de nível de significância (α) de 0,05 (erro tipo I), para aceitar ou rejeitar as hipóteses;

- **Seleção dos sujeitos:** discentes do ensino técnico do curso de informática do Instituto Federal de Alagoas (IFAL), sendo amostragem por conveniência;

- **Design do experimento:** foi aplicado o *within-subject design*, que separou em dois grupos de forma aleatória, usando o *counterbalancing* para mitigar problemas do design, sendo balanceada com mesmo número de repetições e sem blocagem, pois todos os sujeitos eram discentes com mesma experiência. Foi planejado um fator (mensagem de dica) com 2 tratamentos (com DET e com DPC), cada sujeito usa ambos os 2 tratamentos para melhorar a precisão, utilizando-se de ambos para o mesmo objeto (*crossover*) denominado *paired comparison design*;

- **Instrumentação:** a partir dos trabalhos relacionados, vislumbrou-se para o objeto, a criação de STIs, tanto para simular DPC quanto DET, padronizando e enfatizando a janela de mensagens de dicas para ambos os casos. Um STI executa um procedimento micro para cada CC através do *inner loop*, diferente de outras tecnologias como *Computer-Aided Instruction* (CAI) que não possuem este mecanismo [VanLehn 2006] [VanLehn 2011].

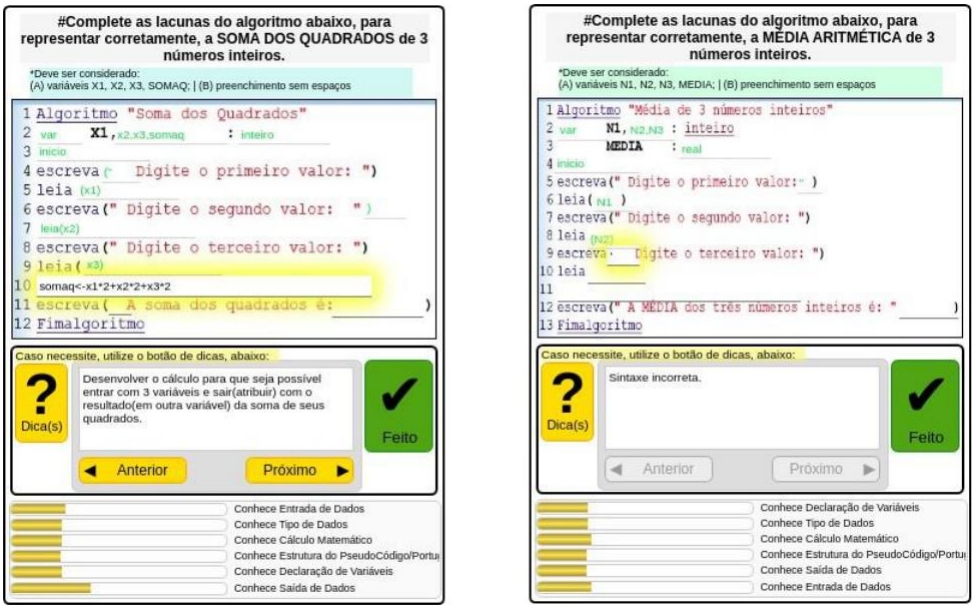
  Para o desenvolvimento dos STIs foi utilizado o *Cognitive Tutor Authoring Tools* (CTAT[5]) que trata-se de uma ferramenta de autoria para não-programadores, ao qual atende os modelos de um STI (discente, pedagógico e domínio), apoiando na produção de tutores do tipo cognitivo e também rastreadores de exemplos/padrões [Aleven et al. 2009], sendo este último adotado neste estudo.

  Há evidências empíricas positivas no uso do CTAT, como exemplo de seus benefícios tem-se: eficiência, tempo de autoria, qualidade dos artefatos, usabilidade entre outros [Dermeval et al. 2018]. No tocante aos STIs do tipo rastreadores de exemplos/padrões, estes avaliam o comportamento discente fazendo uma comparação flexível com exemplos de comportamentos corretos e incorretos para solucionar um determinado problema, fornecendo orientações passo a passo, reconhecendo e interpretando estratégias que o discente está utilizando [Aleven et al. 2009].

  A partir do **[OE1]**, para buscar resposta da questão de pesquisa **[QP]**, foram desenvolvidos[6] 2 STIs do tipo rastreador de exemplos/padrões, com o intuito de cada

[5] http://ctat.pact.cs.cmu.edu/

[6] Seguindo tutorial https://github.com/sisotani/CTAT/wiki/Tutores-rastreadores-de-padrões



(a) STI com DET da questão SomaDosQua-drados

(b) STI com DPC da questão Média3NúmerosInteiros

Figura 4.1. STIs desenvolvidos para o experimento

discente responder sobre questões-problemas denominadas: "SomaDosQuadrados"e "Média3NúmerosInteiros"em pseudocódigos (portugol), nas quais abordaram os conteúdos atuais apresentados em sala de aula (Figura 4.1).

Embora as questões sejam ligeiramente diferentes, atendem as mesmas exigências de habilidade para cada um dos 6 CC: 1) Estrutura do Pseudocódigo/Portugol; 2)Saída de Dados; 3) Entrada de Dados; 4) Tipo de Dados; 5) Declaração de Variáveis; e 6) Cálculo Matemático;

Como recurso para a criação do formato de apresentação das questões para o experimento foi adotada uma perspectiva utilizada no ensino de línguas estrangeiras abordado por [Milková 2015], do tipo "*gapped text*"ou também chamado "*open cloze*", exibindo um código-fonte com espaços/lacunas em branco para serem obrigatoriamente preenchidos com palavra ou texto que falta, sendo exigido do discente um pensamento mais completo para a escolha que preencherá a lacuna de forma correta.

O STI com DET contém 3 dicas para cada lacuna referente a um CC (Figura 4.1a), já os STIs que simula na íntegra as DPC emitidas pelo software Visualg[7] (Figura 4.1b) que é o recurso atual utilizado pelos discentes, contém apenas uma única mensagem (quando tem). O Visualg é um ambiente para programação bastante prático, fácil e didático [Leite et al. 2013], com versões para os sistemas operacionais Windows[8], Linux[9] e Android[10]. Trabalha com pseudocódigo e vem sendo estudado em diversas

[7] https://visualg3.com.br/
[8] https://sourceforge.net/projects/visualg30/
[9] https://snapcraft.io/visualg
[10] http://shorturl.at/juBK3

pesquisas [Viana e Portela 2019] [Borba 2018] [Souza, Silveira e Parreira 2018] [Bilabila 2017] [Souza et al. 2013] [Souza e França 2013] [Souza 2009], que evidenciam a sua importância para o EIP no contexto brasileiro, porém tornando-se similar as ferramentas tradicionais no tocante a problemática aqui relatada, quanto a utilização das DPC.

Na construção das mensagens de dicas foram considerados itens contidos no guia de referência [Becker et al. 2019] para geração/construção de relevantes mensagens de erro de compilador, sendo: curtas, positivas e legível. Estes itens são rechaçados por [Denny, Prather e Becker 2020], seguindo também a prática padrão que envolve o aprendizado incremental com a visualização sequencial das dicas [VanLehn et al. 2005]

A seguir, exemplos de dicas utilizadas nos STIs criados são apresentados, para representar um mesmo erro.

Com DPC:
- Sintaxe incorreta na declaração de variáveis.

Com DET (de forma gradual):
- Incógnitas que representam espaços de memória do computa-dor, para armazenar dados.
- Variáveis necessárias ao funcionamento de todo o código.
- Neste ponto, devem ser adicionadas as outras variáveis 'N2,N3' necessárias ao desenvolvimento pleno do código.

Após o desenvolvimento do planejamento foi efetivada a seguinte **(c)** operacionali-zação com o experimento controlado:

- **Preparação:** discentes do IFAL do 1º ano do ensino técnico do curso de informática, que estavam no assunto referente ao abordado neste estudo, foram convidados a participar do experimento, durante a aula, sob o incentivo de 0,5 ponto extra no bimestre. Sendo confirmados, já imediatamente se tornaram participantes sendo relatado previamente o objetivo e mostrando os recursos possíveis nos STIs a serem utilizados. O projeto foi submetido ao comitê de ética, tendo sua aprovação sob o número do processo: 37474720.0.0000.5013[11];
- **Execução:** foi utilizado o LMS TutorShop[12], **para** hospedar de forma online e gratuita os STIs desenvolvidos. Trata-se de um sistema em que o discente implanta STIs criados no CTAT para gerenciar turmas e atividades discentes de forma gratuita e personalisável para fins de pesquisa, gerando dados (logs) de progressão de uso, cada questão-problema teve 15 minutos de duração (sendo permitido 5 minutos de

[11] http://shorturl.at/afBCE

[12] https://school.tutorshop.web.cmu.edu/

tolerância). Para o experimento os participantes foram aleatoriamente divididos em dois grupos. Os sujeitos foram, um dia antes, encorajados a realizar o experimento no computador/notebook, porém quem não dispusesse poderia utilizar celular ou tablet. Se encontraram com o pesquisador, que também é docente dos mesmos, em data e hora de aula em um aplicativo de comunicação por vídeo. Ambos os grupos acessaram o ambiente TutorShop, cada um com um login específico criado pelo pesquisador para a resolução das tarefas onde cada discente respondeu as 2 questões-problemas[13] propostas. Dos 68 discentes convidados, 44 aceitaram, porém apenas 42 efetivaram a participação nas duas questões-problemas propostas, tendo sido divididos aleatoriamente entre dois grupos: A (n=25) e B (n=17). Os dados (logs) utilizados para a avaliação encontram-se em http://shorturl.at/iopNT.

5. Avaliação

Na execução do experimento, através do ambiente TutorShop, foram armazenados dados de uso dos STIs implementados (DET e DPC). Esses dados contemplam desempenho discente perante a resolução das questões-problemas, entre estes as métricas M1, M2 e M3 objetos deste estudo.

No primeiro momento, para conhecer os dados coletados, foi utilizado estatística descritiva, onde nota-se a distribuição dos dados de forma gráfica, verificando as suas variações [Wohlin et al. 2012]. Observa-se nos gráficos (Figura 5.2) que embora haja diversos *outliers* em $M_{1.DPC}$ praticamente o valor da mediana é igual a $M_{1.DET}$. Quanto as outras métricas, tanto as variações dos dados, quanto a mediana tem valores muito próximos para a métrica M2 ($M_{2.DET}$ e $M_{2.DPC}$) e M3 ($M_{3.DET}$ e $M_{3.DPC}$).

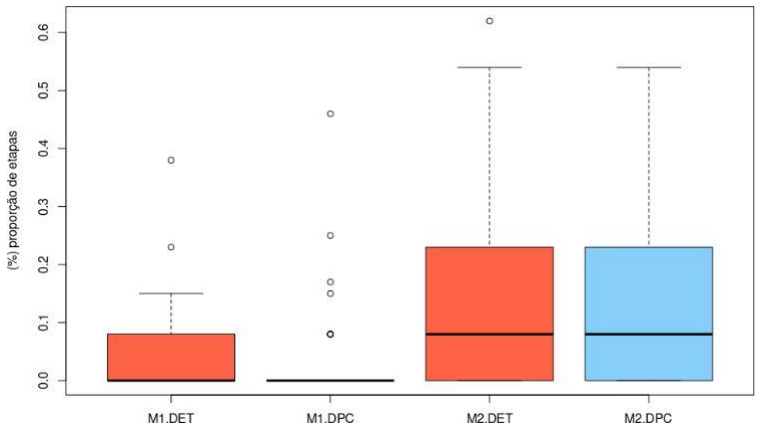
Para constatar formalmente a distribuição dos dados foi utilizado o teste de norma-lidade Shapiro-Wilk, que tem como objetivo fornecer uma estatística capaz de avaliar se uma amostra tem seus dados em distribuição normal, sendo possível utilizar em amostras de qualquer tamanho. Ao executar o teste, obteve-se como resultado um *p-value* abaixo de 0.05 para as três métricas (M1, M2 e M3), sendo portanto constatado que os dados não são normais, sendo necessária a utilização dos testes não-paramétricos.

Diante da especificidade do design do experimento *paired comparison*, foi utilizado o teste não-paramétrico Wilcoxon pareado, sendo este uma alternativa não-paramétrica, tendo como premissas a possibilidade de determinar qual das medidas de um pareamento é a maior e que seja possível ranquear as diferenças, sendo portanto baseado em classificação [Wohlin et al. 2012]. Ao ser executado o referido teste foi constatado os seguintes resultados estatísticos (baseado na mediana) para as métricas apresentadas:

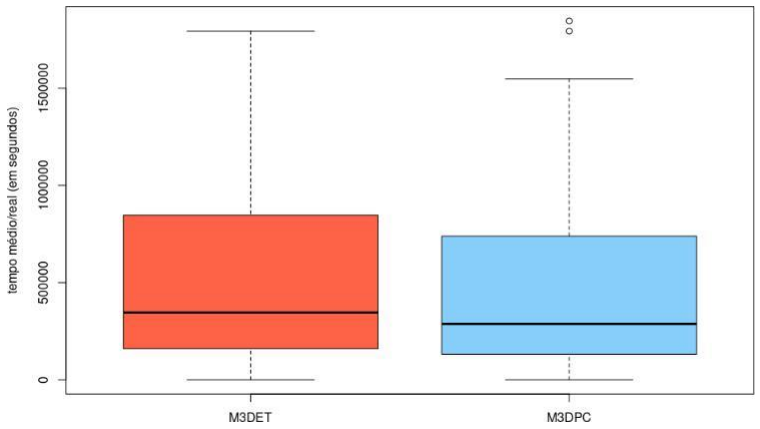
- **M1:** resultando em V = 49.5, *p-value* = 0.5882.
- **M2:** resultando em V = 304, *p-value* = 0.5477.
- **M3:** resultando em V = 481, *p-value* = 0.3598.

Com base nos resultados dos testes, diante das evidências estatísticas com significância do *p-value* > 0,05, nas comparações das métricas (M1, M2 e M3) entre as hipóteses

[13]Para acessar entre em https://school.tutorshop.web.cmu.edu/ com login: testesti e senha: testesti



(a) Comparações entre as proporções de etapas de M1 e M2



(b) Comparação entre os tempo médio/real gasto M3

Figura 5.2. Boxplots das métricas M1, M2 e M3 com DET e DPC

alternativas ($H_{1.1}$, $H_{1.2}$ e $H_{1.3}$) e as hipóteses nulas ($H_{0.1}$, $H_{0.2}$ e $H_{0.3}$), sugere-se que não se tem evidência para rejeitar as hipóteses nulas das três métricas definidas.

6. Discussão do resultado

O resultado deste estudo se junta a outros relacionados de efeito neutro, sendo este puramente de abordagem quantitativa. As hipóteses nulas ($H_{0.1}$, $H_{0.2}$ e $H_{0.3}$) não carregaram evidências estatísticas para suas respectivas rejeições.

Portanto, diante deste resultado obtido de forma quantitativa, a resposta para a **[QP]** deste estudo é que não há diferença no desempenho de tarefas entre os tipos de dicas usando STI para o EIP. A seguir, são identificados alguns contributos deste estudo entre outras considerações aprendidas, assim como uma subseção com suas limitações, ameaças

e validade.

Diferentemente dos estudos relacionados, este abordou nas tarefas dos participantes, puramente o pseudocódigo, representando assim, uma contribuição inédita e relevante, tendo além do espaço das dicas em destaque, a confirmação, pelos dados gerados no ambiente TutorShop, se os participantes utilizaram as dicas, pois precisavam acionar o botão de solicitação, conforme apresentado em STIs padrões, contemplando o recurso de *inner loop*.

Talvez, por esta forma de abordagem ser tão diferente dos compiladores tradicionais, tenha impactado de forma mais coerente com o resultado apresentado, pois a maioria dos estudos relacionados citados permaneceram com dúvida no tocante a detectar se os participantes tinham ao menos visualizado as dicas. Foi possível detectar neste estudo, mesmo que não tenham lido atentamente a dica, pelo menos se foi feita a solicitação para este recurso pelos participantes.

Observa-se portanto, que este quesito de confirmação de utilização das dicas, representa outra relevante contribuição para os estudos da área, já que as métricas M1 (proporção de etapas com solicitações de dica, mas sem respostas incorretas) e M2 (avaliar proporção de etapas com respostas incorretas e solicitações de dicas) estavam diretamente relacionadas às proporções de etapas com solicitações efetivas de dicas.

No tocante a métrica M3 (avaliar o tempo médio/real gasto por problema), este também não surtiu diferença estatística neste estudo entre os tipos dicas (DET e DPC). Este fato poderia ser explicado, conforme já observado em alguns dos estudos relacionados, no sentido de que as submissões de participantes mais expert não chegam a solicitar dicas, diminuindo assim o tempo gasto e consequentemente balanceando as submissões que os participantes utilizaram as solicitações de dicas.

Por fim, vislumbra-se que ao mesmo tempo, aprende-se que há outros fatores que poderiam interferir no estudo e de alguma forma modificar o resultado, seja pelo incremento de uma abordagem qualitativa, seja por outro tipo de apresentação do formato das mensagens de dicas, seja por outras métricas não apresentadas entre outros, adotando-se portanto a postura de tornar este resultado inconclusivo.

6.1. Limitações, Ameaças e Validade

A presente pesquisa limita-se por não haver indícios de quanto a categorização das DET são pedagogicamente úteis, conforme já apresentado por [Suzuki et al. 2017]. Porém, é importante frisar, que a criação das dicas nos STIs deste presente estudo buscou seguir itens contidos no recente guia de referência [Becker et al. 2019] para geração/construção de relevantes mensagens de erro de compilador, rechaçados por [Denny, Prather e Becker 2020], seguindo também a prática padrão que envolve o aprendizado incremental com a visualização sequencial das dicas [VanLehn et al. 2005].

Outro fator que também pode ter sido limitante ao resultado, refere-se a usabilidade dos equipamentos que os participantes utilizaram. Embora eles tenham sido encorajados a usar o computador/notebook, por ter sido realizado remotamente, sendo a grande maioria em suas casas, alguns não dispunham deste equipamento por diversos motivos. Por isso, dos 42 participantes, 27 (64,3%) realizaram no computador/notebook, 14 (33,3%) no

celular e 1 (2,4%) no *tablet*.

Também assume-se que a abordagem qualitativa não realizada neste estudo, poderia apresentar alguma relevância, seja no sentido de corroborar com os resultados ou até mesmo diferir conforme ocorreu em alguns estudos aqui relacionados.

Quanto a amostra, teve 2 sujeitos (fora dos n=42 participantes) que seus dados não foram considerados neste experimento, pois não conseguiram fazer a primeira questão (com DPC). Já todos os sujeitos que iniciaram a questão com DET, conseguiram fazer a segunda questão com DPC.

No geral, observou-se também que o quantitativo dos participantes, em seu espaço amostral (n=42), tenha sido um fator limitante, ao qual se tivessem mais sujeitos talvez haveria um distribuição normal dos dados e consequentemente poderiam ser mudados os resultados dos testes estatísticos com maiores significâncias. No tocante as ameaças da validade, no quesito de conclusão, verifica-se que por conta do *p-value* terem sidos altos para todas as métricas utilizadas (M1, M2 e M3) sendo talvez acarretado pelo tamanho amostral (diminuindo o poder estatístico) e consequentemente pelo teste utilizado, violação de suposições entre outros.

Quanto a validação interna, embora não tenha chegado positivamente de forma estatística, o relacionamento causal entre tratamento e o seu resultado pode ter sido ameaçado, por conta das métricas escolhidas, sendo este fator limitante devido ao ambiente selecionado, denominado TutorShop. A validação externa pode estar ameaçada por conta do público-alvo e ambiente selecionado, sendo comprometida a generalização dos resultados deste experimentos com outros contextos.

7. Conclusão e Trabalhos Futuros

Diante do problema de pesquisa apresentado neste estudo, que envolvia a falta de conheci-mento sobre o efeito das DET em comparação com as DPC, de acordo com as métricas investigadas, sugere-se que os resultados não obteve diferença estatisticamente significante, portanto, neste prisma não se pode afirmar que as DET promovem um maior efeito no de-sempenho discente para a realização de questões-problemas em algoritmos computacionais em detrimento com as DPC.

Porém, corroborando com o recente estudo que evidenciou de forma empírica [Denny, Prather e Becker 2020] a relevância das dicas para o EIP, percebe-se claramente que ainda há muito a ser feito com experimentos no futuro, principalmente por este presente estudo representar uma nova abordagem para analisar as dicas, sendo aqui apresentadas em STIs.

Essas dicas relatam detalhes sobre erros que o programador cometeu e, na maioria das vezes, são o único feedback que o programador obtém do compilador [Becker et al. 2016], destacando-se portanto a sua importância para o EIP.

Tendo vistas a contínua necessidade de trabalhos futuros, vislumbra-se a relevância de novos relatos empíricos, incluindo outras métricas e diferentes abordagens, conforme já discutidos, não só por novos STIs, mas seja, por exemplo, pela geração de dicas automáticas com qualidade a partir de aprendizado de máquina, conforme sugerido em

[Price et al. 2018] para serem utilizadas em cursos massivos a partir de diversas técnicas [Phothilimthana e Sridhara 2017], entre outras possibilidades.

Referências

Aleven, Vincent et al. (2006). "Toward Meta-cognitive tutoring: A model of help seeking with a cognitive tutor". Em: *International Journal of Artificial Intelligence in Education* 16.2, pp. 101–128.

Aleven, Vincent et al. (2009). "A new paradigm for intelligent tutoring systems: Example-tracing tutors". Em: *International Journal of Artificial Intelligence in Education* 19.2, pp. 105–154.

Aleven, Vincent et al. (2016). "Help Helps, but only so Much: Research on Help Seeking with Intelligent Tutoring Systems". Em: *International Journal of Artificial Intelligence in Education* 26.1, pp. 205–223.

Aleven, V. and McLaughlin, E. A. and Glenn, R. A. and & Koedinger e K. R (2017). "Instruction Based on Adaptive Learning Technologies". Em: *Handbook of research on learning and instruction*. Ed. por R. E. Mayer e P. Alexander. 2nd. Routledge. Cap. 24, pp. 522–560.

Antonucci, Paolo et al. (2015). "An incremental hint system for automated programming assignments". Em: *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE* 2015-June, pp. 320–325.

Basili, Victor R. e H. Dieter Rombach (1988). "The TAME Project: Towards Improvement-Oriented Software Environments". Em: *IEEE Transactions on Software Enginee-ring* 14.6, pp. 758–773.

Becker, Brett A. et al. (2016). "Effective compiler error message enhancement for novice programming students". Em: *Computer Science Education* 26.2-3, pp. 148–175.

Becker, Brett A. et al. (2019). *Compiler error messages considered unhelpful: The lands-cape of text-based programming error message research*, pp. 177–210.

Bilabila, A. M. (2017). "CompAlg-Ferramenta de Ensino e Aprendizagem da Lógica de Programação". Tese de dout. Universidade do Porto, p. 103.

Borba, Fabrício Hartmann (2018). "O Software Visualg Como Recurso Didático no ensino da lógica de programação". Tese de dout. Universidade do Vale do Taquari - UNIVATES, pp. 1–113.

Bruce, Kim B. (2018). "Five big open questions in computing education". Em: *ACM Inroads* 9.4, pp. 77–80.

Burton, Richard R. e John Seely Brown (1979). "An investigation of computer coaching for informal learning activities". Em: *International Journal of Man-Machine Studies* 11.1, pp. 5–24.

Crow, Tyne, Andrew Luxton-Reilly e Burkhard Wuensche (jan. de 2018). "Intelligent Tutoring Systems for Programming Education: A Systematic Review". Em: *ACM International Conference Proceeding Series*. Association for Computing Machi-nery, pp. 53–62.

D'Antoni, Loris et al. (2015). "How can automatic feedback help students construct automata?" Em: *ACM Transactions on Computer-Human Interaction* 22.2.

Denny, Paul, Andrew Luxton-Reilly e Dave Carpenter (2014). "Enhancing syntax error messages appears ineffectual". Em: *ITICSE 2014 - Proceedings of the 2014 In-*

novation and Technology in Computer Science Education Conference, pp. 273–278.

Denny, Paul, James Prather e Brett A. Becker (2020). "Error Message Readability and Novice Debugging Performance". Em: *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, pp. 480–486.

Dermeval, Diego et al. (2018). *Authoring Tools for Designing Intelligent Tutoring Systems: a Systematic Review of the Literature*. Vol. 28. 3. International Journal of Artificial Intelligence in Education, pp. 336–384.

EEF (2018). *Teaching & Learning Toolkit*. Rel. técn.

Head, Andrew et al. (2017). "Writing reusable code feedback at scale with mixed-initiative program synthesis". Em: *L@S 2017 - Proceedings of the 4th (2017) ACM Confe-rence on Learning at Scale*, pp. 89–98.

Khouri, Cátia Mesquita Brasil, Gidevaldo Novais dos Santos e Maria Silva Santos Barbosa (2020). "Mapeamento Sistemático em Metodologias de Ensino-aprendizagem de Programação". Em: pp. 13–27.

Koedinger, Kenneth R. et al. (2013). "Using data-driven discovery of better student models to improve student learning". Em: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinforma-tics)* 7926 LNAI, pp. 421–430.

Lahtinen, Essi, Kirsti Ala-Mutka e Hannu Matti Järvinen (2005). "A study of the difficulties of novice programmers". Em: *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, pp. 14–18.

Lane, H. Chad e Kurt Vanlehn (2004). "A dialogue-based tutoring system for beginning programming". Em: *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004* 2, pp. 449–454.

Leite, Vanessa Matias et al. (2013). "VisuAlg : Estudo de Caso e Análise de Compilador destinado ao ensino de Programação". Em: *Nuevas Ideas en Informática Educativa TISE*, pp. 637–640.

Luxton-Reilly, Andrew et al. (jul. de 2018). "Introductory programming: A systematic literature review". Em: *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*. Association for Computing Machinery, pp. 55–106.

Marwan, Samiha, Joseph Jay Williams e Thomas Price (2019). "An evaluation of the impact of automated programming hints on performance and learning". Em: *ICER 2019 - Proceedings of the 2019 ACM Conference on International Computing Education Research* Study 2, pp. 61–70.

Milková, Eva (2015). "Development of programming capabilities inspired by foreign language teaching". Em: *Procedia - Social and Behavioral Sciences* 171, pp. 172– 177.

Morgan, Michael et al. (jan. de 2017). "Understanding international benchmarks on student engagement: Awareness and research alignment from a computer science perspec-tive". Em: *ITiCSE-WGR 2017 - Proceedings of the 2017 ITiCSE Conference on Working Group Reports*. Vol. 2017-Janua. Association for Computing Machinery, Inc, pp. 1–24.

Pettit, Raymond, John Homer e Roger Gee (2017). "Do enhanced compiler error messages help students? Results inconclusive". Em: *Proceedings of the Conference on*

Integrating Technology into Computer Science Education, ITiCSE March 2017, pp. 465–470.

Phothilimthana, Phitchaya Mangpo e Sumukh Sridhara (2017). "High-Coverage Hint Generation for Massive Courses". Em: x, pp. 182–187.

Price, Thomas W. et al. (2018). "The impact of data quantity and source on the quality of data-driven hints for programming". Em: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10947 LNAI, pp. 476–490.

Qian, Yizhou e James Lehman (2017). "Students' misconceptions and other difficulties in introductory programming: A literature review". Em: *ACM Transactions on Computing Education* 18.1, pp. 1–24.

Robins, Anthony, Janet Rountree e Nathan Rountree (2003). "Learning and teaching programming: A review and discussion". Em: *International Journal of Phytoreme-diation* 21.1, pp. 137–172.

Souza, Cláudio Morgado de (2009). "VisuAlg - Ferramenta de Apoio ao Ensino de Progra-mação". Em: *Revista Eletrônica TECCEN* 2.2, p. 01.

Souza, Marcelo Batista De et al. (2013). "Uma Abordagem Metodológica voltada para o Ensino-Aprendizagem de Algoritmos". Em: *Novas Tecnologias na Educação* 11.1, pp. 1–10.

Souza, Márcia V. R. de e A. César C. França (2013). "Ferramentas de Auxílio ao Aprendi-zado de Programação : Um Estudo Comparativo". Em: *Proceedings of WEIBASE - Workshop de Educação em Computação do ERBASE* August 2015.

Souza, Naidú Gasparetto de, Sidnei Renato Silveira e Fábio José Parreira (2018). "Proposta de uma Metodologia para Apoiar os Processos de Ensino e de Aprendizagem de Lógica de Programação na Modalidade de Educação a Distância". Em: *Educaçao Cultura e Comunicação* 9.18, pp. 207–232.

Suzuki, Ryo et al. (2017). "Exploring the design space of automatically synthesized hints for introductory programming assignments". Em: *Conference on Human Factors in Computing Systems - Proceedings* Part F1276, pp. 2951–2958.

VanLehn, Kurt (2006). "The Behavior of tutoring systems". Em: *International Journal of Artificial Intelligence in Education* 16.3, pp. 227–265.

VanLehn, Kurt (2011). "The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems". Em: *Educational Psychologist* 46.4, pp. 197– 221.

VanLehn, Kurt et al. (2005). "The andes physics tutoring system: Lessons learned". Em: *International Journal of Artificial Intelligence in Education* 15.3, pp. 147–204.

Viana, Gracilene Américo e Carlos Dos Santos Portela (2019). "O Uso de Softwares Edu-cativos para Introdução de Lógica de Programação no Ensino de Base e Superior". Em: *Informática na educação: teoria & prática* 22.1, pp. 10–22.

Wohlin, Claes et al. (2012). *Experimentation in Software Engineering*. Ed. por Intergovern-mental Panel on Climate Change. Vol. 91. 5. Springer Berlin Heidelberg, pp. 1– 30.