UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

GUSTAVO CORREIA GONZALEZ

MONOGRAFIA

CAMPO MOURÃO

2016

GUSTAVO CORREIA GONZALEZ

Proposta de Trabalho de Conclusão de Curso de Graduação apresentado à disciplina de Trabalho de Conclusão de Curso 1, do Curso de Bacharelado em Ciência da Computação do Departamento Acadêmico de Computação da Universidade Tecnológica Federal do Paraná, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Igor Scaliante Wiese

Coorientador: Prof. Dr. Marco Aurélio Graciotto Silva

CAMPO MOURÃO

2016

Resumo

Gonzalez, Gustavo. . 2016. 13. f. Monografia (Curso de Bacharelado em Ciência da Computação), Universidade Tecnológica Federal do Paraná. Campo Mourão, 2016.

O elevado nível de reprovação em disciplinas onde são ensinados conceitos básicos de programação (Holcomb; Simone, 2016), em qualquer grau de ensino, é um problema enfrentado por muitos alunos e tem sido alvo de variadas pesquisas. Existe um conjunto de razões que estão relacionadas com a origem do problema, como o método de ensino e aprendizagem, falta de algumas competências e interesse por parte dos alunos (Sinclair *et al.*, 2015), e a própria dificuldade do tema.

Contexto: O desenvolvimento da pesquisa será realizado na Universidade Tecnológica Federal do Paraná, com alunos selecionados entre o primeiro e segundo ano da matéria de Algoritmos e Algoritmos e Estruturas de Dados 1.

Objetivo: Investigar o impacto do uso do mecanismo de dicas personalizadas no ensino de conceitos básicos de programação, mais expecificamente em estrutura de condição e laço de repetição. Para atingir esse objetivo, foram criadas duas questões de pesquisas descritas como:

- **QP**$_1$: *Existe impacto do uso do mecanismo de dicas para minimizar erros e melhorar soluções de exercícios?*
  Esta questão de pesquisa tem como objetivo investigar se o mecanismo de dicas irá ajudar os alunos a obterem melhores resultados na execução de exercícios de programação.
- **QP**$_2$: *Quais dicas personalizadas ajudam mais os alunos?*
  Esta questão de pesquisa avaliará qual tipo de dica é melhor, podendo ser randomizadas em função do tipo do exercício, geradas a partir de erros cometidos em exercícios passados e sem dicas. Levando em consideração a fonte da dica gerada, podendo ser por alunos iniciantes, alunos experientes ou professores.

Método: Será desenvolvido um software para auxiliar o aprendizado de programação que utilizá um sistema de dicas personalizadas aos alunos no momento da realização de exercícios. Para validar a eficiência do software será realizando um estudo controlado com três grupos de

alunos distribuídos de forma homogênea em relação ao nível de conhecimento na programação. O primeiro grupo receberá dicas classificadas a partir de erros anteriores, o segundo será com dicas randomizadas levando em consideração o assunto abordado no exercício e o terceiro sem dicas.

Resultados esperados: Esperamos que o software realmente ajude no aprendizado dos alunos na programação tornando o ensino mais dinâmico e personalizado. Para que no futuro possa diminuir o número de reprovações nas matérias introdutórias de programação e em desistências do curso de Ciência da Computação.

Palavras-chaves: Aplicação, Dicas, Dicas Personalizadas, Learnersourcing, Crowdsourcing

Lista de figuras

Lista de tabelas

Sumário

1 Introdução **6**

2 Referencial Teórico **8**

2.1 Reprovações em matérias de programação 8

2.2 Novas abrodagens de ensino 8

2.3 Sistemas de dicas 8

3 Proposta **10**

3.1 Descrição do Sistema de Dicas 10

3.1.1 Cadastro do Usuário 10

3.1.2 Realização de exercícios 10

3.1.3 Diário 11

3.2 Banco de Dicas 11

3.3 Formato do Estudo 11

Referências **12**

Capítulo 1

Introdução

O ensino de linguagens de programação tem o propósito de conseguir desenvolver nos aluno um conjunto de competências necessárias para conceber programas e sistemas computacionais capazes de resolver problemas reais. O insucesso na aprovação dos estudantes em disciplinas de programação, é um tema que tem sido alvo de alguns estudos (Bosse; Gerosa, 2015), (Cukierman, 2015).

O problema do insucesso é evidenciado por (Lahtinen *et al.*, 2005) realizaram uma pesquisa com 559 alunos e 34 professores de diferentes universidades para estudar as dificuldades na aprendizagem de programação, como resultado obtido foi percebido que as questões mais difíceis na programação são: a compreensão de como projetas um programa para resolver uma tarefa determinada, dividir as funcionalidades em procedimentos e encontrar erros de seus próprios programas. Estas são as capacidades que o aluno deve obter para entender as maiores entidades do programa em vez de apenas alguns detalhes sobre eles. Já os conceitos de programação mais difíceis foram recursão, ponteiros e referências, tipos de dados abstratos, manipulação de erro. Os professores apontaram como sendo os conteúdos mais difíceis os mesmos que os alunos.

O objetivo desse trabalho é criar um software de código aberto para auxiliar na aprendizagem de programação, implementando um sistema colaborativo de dicas escritas pelos próprios usuários. Para investigar se do uso do mecanismo de dicas personalizadas é capaz de melhorar o rendimentos dos alunos nos exercícios de estrutura de condição e laço de repetição.

No software serão implementadas funcionalidades que permitam o usuário resolver exercícios em qualquer linguagem de programação, fornecer dicas para a solução de exercícios, um diário para relatar as dificuldades enfrentadas durante a execução dos exercícios.

Para realizar a validação da ferramenta será realizado um estudo controlado na Universidade Tecnológica Federal do Paraná com três grupos de estudantes escolhidos aleatóriamente de forma homogênea em relação ao nível de conhecimento na programação, um grupo utilizará o software de dicas normal, outro irá dispor de dicas personalizado e o último utilizará o software sem o sistema de dicas personalizadas.

Os próximos capítulos estão organizados em três partes. O objetivo do capítulo de referencial teórico é apresentar e discutir os principais conceitos que envolvem este trabalho. Este procedimento é importante, pois discutiremos os pontos de vista de diversos autores, assim como diversas abordagens alternativas. O objetivo do capítulo de proposta é apresentar a metodologia que será utilizada para desenvolver o estudo e o software. O objetivo do capítulo de resultados preliminares é apresentar os avanços que o estudo obteve durante o seu desenvolvimento.

Capítulo 2

Referencial Teórico

Neste capítulo apresentaremos uma revisão da literatura agrupando trabalhos relacionados a reprovações e dificuldades de alunos em matérias de programação, novas abordagens de ensino, sistemas de dicas.

Reprovações em matérias de programação

O entendimento e aplicação dos conceitos estudados em disciplinas de programação é fundamental para que o aluno consiga desenvolver programas mais complexo. Helminen e Malmi (2010) afirmam que a programação é uma competência essencial no curso de Ciência da Computação. Segundo Bosse e Gerosa (2015) os estudantes normalmente apresentam uma grande dificuldade com os conteúdos abordados, ocasionando em reprovação ou desistência.

Podemos apresentar como uma das causas de reprovações e desistência a falta de engajamento nos estudos por parte dos alunos (Sinclair *et al.*, 2015)

Novas abrodagens de ensino

Sistemas de dicas

Price (2015) está utilizando um sistema de tutores inteligentes (STI) que podem manter os alunos no caminho certo, na ausência de instrutores, fornecendo sugestões e advertências para os alunos que precisam de ajuda. Além disso, as técnicas baseadas em dados pode gerar esse feedback automaticamente a partir de tentativas dos estudantes anteriores de um

problema. O autor está aplicando o seu estudo permitindo que os alunos escrevam códigos que se conectam com seus interesses, tais como jogos, aplicações e histórias. Como exemplo concreto, imagine um aluno que esteja desenvolvendo um jogo simple, quer requer a utilização de variáveis, laços de repetição, condicionais. Mas o aluno está com dificuldades em relação a uma funcionalidades que o jogo terá, ele poderá pedir uma dica para implementar essa funcionalidade que será gerada automaticamente pelo sistema com relação as tentativas anteriores de outros alunos.

Elkherj e Freund (2014) apontam que as sugestões utilizadas em sistemas de aprendizagem on-line para ajudar os alunos quando eles estão tendo dificuldades são fixados antes do tempo e não dependem das tentativas mal sucedidas que o aluno já fez. Isto limita severamente a eficácia das sugestões. Eles desenvolveram um sistema alternativo para dar dicas aos estudantes. A principal diferença é que o sistema permite um instrutor enviar uma dica para um estudante após o aluno ter feito várias tentativas para resolver o problema e falhou. Depois de analisar os erros do aluno, o instrutor é mais capaz de entender o problema no pensamento do aluno e enviar-lhes uma dica mais útil. O sistema foi implantado em um curso de probabilidade e estatística com 176 alunos, obtendo *feedback* dos alunos muito positivo. Mas o desafio que os autores enfrentam é como escalar efetivamente o sistema de forma que todos os alunos que precisam de ajuda obtenham dicas eficazes. Pois com uma grande quantidade de alunos ficaria exaustivo para os instrutores avaliarem cada submissão dos exercícios de cada aluno para retornarem uma dica especifica do erro cometido. Então eles criaram um banco de dados de dica que permite que os instrutores reutilizem, compartilhem e melhorem em cima de sugestões escritas anteriormente.

Cummins *et al.* (2016) investigaram o uso de dicas para 4.652 usuários qualificados em um ambiente de aprendizagem on-line de grande escala chamado Isaac, que permite aos usuários para responder a perguntas de física com até cinco dicas. Foi investigado o comportamento do usuário ao usar dicas, engajamento dos usuários com desvanecimento (o processo de tornar-se gradualmente menos dependentes das dicas fornecidas), e estratégias de dicas incluindo decomposição, correção, verificação ou comparação. Como resultados obtidos, os alunos apresentaram estratégias para as resoluções dos exercícios sendo a mais comum é ver o conceito da dica para realizar a decomposição do problema e, em seguida, enviar uma resposta correta. A outra estratégia é usar os conceitos da dica para determinar se a pergunta pode ser respondida, uma grande proporção dos usuários que utilizaram essa estratégia acabaram não tentando responder a pergunta

Capítulo 3

Proposta

Neste capítulo apresentaremos o método utilizado para a elaboração deste trabalho.

Descrição do Sistema de Dicas

Essa seção, descreve o sistema de dicas a ser desenvolvido. O processo de realização de exercícios, que é onde o sistema deve atuar, pode ser dividid em quatro fases: cadastro do usuário, realização de exercícios, utilização de dicas (providas por usuários do sistema) e validação do exercício. A descrição apresentada será usada para a modelagem do sistema.

Cadastro do Usuário

Quando um usuário desejar realizar o cadastro no nosso sistema, ele poderá escolher entre a opção de aluno ou professor. O aluno, realiza exercícios disponíveis no banco de dados do sistema, preenche um diário não obrigatório para cada exercício, cria uma dica para cada exercício e consulta seu perfil com seus dados pessoais e um relatório de suas submissões de exercícios. O professor, além de realizar as mesmas atividades do aluno, também poderá criar salas com exercícios pré definidos e convidar alunos para fazer parte dela e submeter exercícios para o sistema.

Realização de exercícios

Depois de concluído o cadastro de usuário, o aluno ou professor é encaminhado para a tela principal do sistema onde poderá realizar as atividades permitidas para cada tipo de usuário.

Caso a opção seja realizar um exercício, o usuário poderá escolher qual tipo de exercício ele deseja realizar e o nível de dificuldade, após preencher esses dados o sistema retornará uma lista com alguns exercícios que satisfazem os campos de busca. O usuário irá escolher um exercício, realiza-lo na linguagem de programação que deseja e submeter para a validação do sistema. O exercício pode ser aceito ou não sistema, caso seja rejeitado o sistema retornará qual tipo de erro ocorreu na compilação do exercício, o usuário poderá realizar submissões até que consiga obter sucesso na submissão. Cada submissão, tanto correta ou incorreta, é gravada no banco de dados na forma de *log*, sendo salvo o usuário que realizou o exercício, todas as submissões, os erros cometidos, o tempo demorado para obter sucesso no exercício.

Diário

Após o usuário ter realizado um exercício e obter sucesso, o sistema perguntará se o usuário deseja realizar um diáro para o respectivo exercício, caso aceite, o sistema redirecionará o usuário para a página do diário onde o usuário informará os dados referentes as submissões dos exercícios.

Banco de Dicas

Formato do Estudo

Referências

BOSSE, Yorah; GEROSA, Marco Aurélio. Reprovações e trancamentos nas disciplinas de introdução à programação da universidade de são paulo: Um estudo preliminar. In: *XXIII WEI–Workshop sobre Educação em Informática. Recife, Julho*, 2015.

CUKIERMAN, Diana. Predicting success in university first year computing science courses: The role of student participation in reflective learning activities and in i-clicker activities. In: *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, 2015. (ITiCSE '15), p. 248–253. ISBN 978-1-4503-3440-2. Disponível em: http://doi.acm.org/10.1145/2729094.2742623.

CUMMINS, Stephen; STEAD, Alistair; JARDINE-WRIGHT, Lisa; DAVIES, Ian; BERESFORD, Alastair R.; RICE, Andrew. Investigating the use of hints in online problem solving. In: *Proceedings of the Third (2016) ACM Conference on Learning @ Scale*, 2016. (L@S '16), p. 105–108. ISBN 978-1-4503-3726-7. Disponível em: http://doi.acm.org/10.1145/2876034.2893379.

ELKHERJ, Matthew; FREUND, Yoav. A system for sending the right hint at the right time. In: *Proceedings of the First ACM Conference on Learning @ Scale Conference*, 2014. (L@S '14), p. 219–220. ISBN 978-1-4503-2669-8. Disponível em: http://doi.acm.org/10.1145/2556325.2567864.

HELMINEN, Juha; MALMI, Lauri. Jype - a program visualization and programming exercise tool for python. In: *Proceedings of the 5th International Symposium on Software Visualization*, 2010. (SOFTVIS '10), p. 153–162. ISBN 978-1-4503-0028-5. Disponível em: http://doi.acm.org/10.1145/1879211.1879234.

HOLCOMB, Kayla M.; SIMONE, Nevan F. The role of chronology in analyzing introductory programming assignments (abstract only). In: *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 2016. (SIGCSE '16), p. 725–726. ISBN 978-1-4503-3685-7. Disponível em: http://doi.acm.org/10.1145/2839509.2851062.

LAHTINEN, Essi; ALA-MUTKA, Kirsti; JÄRVINEN, Hannu-Matti. A study of the difficulties of novice programmers. In: Acm. *ACM SIGCSE Bulletin*, 2005. v. 37, n. 3, p. 14–18.

PRICE, Thomas W. Integrating intelligent feedback into block programming environments. In: *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, 2015. (ICER '15), p. 275–276. ISBN 978-1-4503-3630-7. Disponível em: http://doi.acm.org/10.1145/2787622.2787748.

SINCLAIR, Jane; BUTLER, Matthew; MORGAN, Michael; KALVALA, Sara. Measures of student engagement in computer science. In: *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, 2015. (ITiCSE '15), p. 242–247. ISBN 978-1-4503-3440-2. Disponível em: http://doi.acm.org/10.1145/2729094.2742586.