Uma ferramenta gamificada de apoio à disciplina introdutória de programação

André Campos[1], Renato Gardiman[2], Charles Madeira[3]

[1] Departamento de Informática e Matemática Aplicada – DIMAp
[2] Departamento de Computação e Automação – DCA
[3] Instituto Metrópole Digital – IMD
Universidade Federal do Rio Grande do Norte (UFRN)
Caixa Postal 1524 – 59.078-970 – Natal – RN – Brazil

[1]andre@dimap.ufrn.br, [2]renatorqg@gmail.com, [3]charles@imd.ufrn.br

Abstract. *Learning computer programming in Brazilian high education courses is still a big issue for freshman students. In order to attenuate this difficulty, a gamified tool was developed aiming to support programming practice. This paper presents the results of the tool's application in an introductory programming course. It addresses the tool, its review by the students as well as students behaviors observed during their use over five semesters. Such behaviors outlined the development of a new version of the tool.*

Resumo. *O aprendizado da programação de computadores em cursos superiores de computação no Brasil ainda se apresenta como um grande problema para os alunos ingressantes. Para minimizar esse obstáculo, foi desenvolvida uma ferramenta gamificada de auxílio à prática de programação. Este artigo apresenta os resultados de sua aplicação em uma disciplina introdutória de programação. O texto discorre sobre a ferramenta, sua avaliação por parte dos alunos e comportamentos observados nos alunos durante o seu uso ao longo de cinco semestres. Tais comportamentos delinearam o desenvolvimento de uma nova versão da ferramenta.*

1. Introdução

Um dos principais problemas de retenção nos cursos de computação encontra-se nas disciplinas introdutórias de matemática e de programação. Vários artigos já foram publicados ressaltando esta dificuldade [Jenkins 2002] [Bergin e Reilly 2005] [Bennedsen e Caspersen 2007] [Gomes e Mendes 2007], bem como procurando identificar as razões pelas quais grande parte dos alunos ingressantes nos cursos esbarram nas disciplinas iniciais [Lahtinen *et. al*. 2005][Campello e Lins 2008][Silva, Tedesco e Melo 2014].

Entre as possíveis soluções para o problema, está a prática diária de resolução de problemas, normalmente pequenos e pontuais [Gomes, Henriques e Mendes 2008]. Para que essa prática seja eficaz, é necessário que os alunos estejam motivados na sua realização. Ou seja, o processo de aprendizado precisa ser uma experiência "interessante" sob o ponto de vista do aluno. A imprecisão do termo "interessante" serve para ressaltar a diversidade de percepção do que é motivador em diferentes alunos. Dois aspectos que julgamos essencial para tornar a prática em uma experiência "interessante" é: 1) envolver

os alunos em um ambiente baseado em mecânicas comumente presentes em jogos, estratégia conhecida como gamificação, devido à predisposição que os jovens têm pelo ato de jogar, e 2) possibilitar ao aluno saber em "tempo-real" a corretude de suas soluções, podendo assim aproveitar o *feedback* para perceber claramente a sua progressão na atividade e permitir reformular os conceitos praticados.

O presente artigo disserta sobre a experiência dos autores no desenvolvimento e aplicação de uma ferramenta de auxílio à realização de atividades de programação, nomeada Kodesh (Koding Shell). O desenvolvimento da ferramenta teve como premissa que os dois aspectos mencionados são essenciais para motivar alunos na prática de programação. Uma avaliação de aceitação foi realizada com os usuários iniciais da ferramenta e uma nova versão foi desenvolvida com o aprendizado de seu uso ao longo dos semestres.

O texto está organizado em 5 seções, contando com esta. A segunda seção expõe as motivações da criação da ferramenta, apresentando o contexto do momento em que foi criada e as ferramentas existentes na época. A seção seguinte apresenta a ferramenta e uma avaliação de aceitação feita com os alunos. Em seguida, discursamos sobre a experiência adquirida ao longo dos semestres usando a ferramenta, através de um conjunto de lições aprendidas. Por fim, apresentamos a nova versão da ferramenta, com melhorias oriundas desta experiência.

2. Contextualização

2.1. Definição de requisitos

No início de 2013, ocorreu a entrada da primeira turma do novo curso de Bacharelado em Tecnologia da Informação (BTI) da UFRN, que segue um modelo em dois ciclos em conjunto com os cursos tradicionais de Bacharelado em Ciência da Computação (BCC) e Bacharelado em Engenharia de Software (BES) já existentes na Universidade. O modelo em dois ciclos permite que os alunos ingressem em um curso de computação - BTI, postergando a sua escolha por Ciência da Computação ou Engenharia de Software para meados do primeiro ciclo do curso. Em seguida, os alunos podem completar o curso tradicional escolhido através de um segundo ciclo.

Para este primeiro ano de transição do modelo, o ingresso no BCC ainda foi feito de forma concomitante ao ingresso no novo BTI. Com isso, o número de vagas em computação passou de 90 (50 para o BCC e 40 para o BES) para 265 (25 diretamente para o BCC e 240 para o BTI). O aumento significativo de vagas no processo seletivo combinado ao alto índice de retenção nas disciplinas iniciais, principalmente nas disciplinas introdutórias de programação, exigiu dos professores um novo olhar sobre a metodologia de ensino/aprendizado a ser adotada.

No novo curso, seriam necessárias várias turmas em paralelo. Por essa razão, optou-se por adotar um material de ensino padronizado. O intuito da padronização do material de apoio didático foi de dar suporte igualitário às diferentes turmas, uma vez que as avaliações seriam realizadas de forma conjunta. Além da padronização, foram levantados três requisitos para uso de uma ferramenta de apoio metodológico no processo de ensino-aprendizado.

1. **Possuir mecanismos de engajamento e motivação para os alunos realizarem atividades fora de sala de aula**: Com o aumento significativo do número vagas, a expectativa era de elevação da retenção na disciplina inicial de programação. A hipótese assumida para esse requisito foi que, se os alunos se engajassem na realização de práticas de programação fora da sala de aula, a retenção poderia ser reduzida ou, no pior caso, ajudaria a não aumentá-la quando comparada aos semestres anteriores.
2. **Permitir aos alunos a prática de programação de forma *online***: Essa necessidade originou-se novamente da grande quantidade de novos alunos e a impossibilidade de ter laboratórios de acesso livre para todos em horários extra classe. Uma ferramenta *online* possibilitaria que os alunos pudessem iniciar suas práticas desde o primeiro dia de aula, independentemente das dificuldades técnicas ou logísticas para instalarem em seus computadores o compilador da linguagem C requerido na disciplina.
3. **Permitir aos professores estruturar as atividades segundo a organização do conteúdo previsto para a disciplina**: Dada a padronização do conteúdo estabelecido pelos professores, fazia-se necessário que a ferramenta fosse flexível o suficiente para que os professores organizassem as atividades segundo o cronograma pré-estabelecido para a disciplina.

2.2. Soluções existentes

Algumas ferramentas de suporte ao ensino de programação já tinham sido testadas em períodos anteriores. Porém, elas se enquadravam em duas categorias:

- **Ferramentas lúdicas voltadas a introdução a algoritmos**: São ferramentas que visam introduzir conceitos, como CodeStudio (code.org) e o Scratch [Resnick et al. 2009], bem como jogos e puzzles [Vahldick, Mendes e Marcelino 2014]. No uso de algumas dessas ferramentas em edições anteriores da disciplina, verificamos que estas ferramentas são interessantes para os primeiros dias de aula, quando noções de algoritmos e resolução de problemas estão sendo apresentados. Quando a sintaxe da linguagem e problemas mais complexos surgem, faz-se necessário migrar para outras ferramentas de apoio.
- **Ferramentas de correção automática**: a adaptação de ferramentas de correção automática de código baseadas nas maratonas de programação, como BOCA [Campos e Ferreira 2004], foram igualmente testadas em períodos anteriores. A dificuldade constatada no uso dessas ferramentas com alunos que estavam iniciando em programação foi a formalização necessária na correção automática e no nível de informação dada quando uma solução proposta encontrava-se errada. Em geral, tais ferramentas indicam que a solução não passou nos testes, porém sem informação adicional alguma que auxilie no processo de aprendizado do aluno.

As duas categorias de ferramentas trazem características importantes para o aprendizado de programação. Enquanto a primeira versa principalmente sobre o

engajamento do aluno, a segunda trata do *feedback* imediato. Certamente tais características não são mutuamente exclusivas. Porém, na época, apenas encontramos ferramentas unindo ambas características apenas em linguagens de scripts, tais como CodeAcademy (http://www.codecademy.com). A linguagem introdutória adotada pelo curso de BTI era e ainda é C. Atualmente, existem inúmeras iniciativas com esse propósito, envolvendo inclusive a linguagem C, a exemplo do URI [Bez et al. 2014]. Porém, na época, as iniciativas ainda eram incipientes.

Visando dar um maior suporte às disciplinas de programação, alguns trabalhos tais como o BOSS [Joy, Griffiths e Boyatt 2005], o JOnline [Santos e Ribeiro 2011], o MOJO [Chaves *et al.* 2013] e o Feeper [Alves e Jaques 2014] foram desenvolvidos e passaram a fazer uso de recursos automatizados para dar um melhor retorno aos alunos a partir das ferramentas de correção automática existentes. O BOSS suporta a avaliação do código dos alunos por meio de coleta de submissões com uma interface para obtenção de *feedbacks*. O JOnline agrega novas funcionalidades ao BOCA tais como apresentação de dicas em linguagem natural e possibilidade de programação colaborativa. O MOJO, por sua vez, tem um foco maior no professor visando diminuir sua sobrecarga de trabalho pelo fato de selecionar o que necessita de *feedback*. Enfim, o Feeder disponibiliza uma funcionalidade que permite registrar dúvidas e comentários diretamente nas linhas de código fonte do aluno, dando margem para uma melhor discussão sobre o código fonte gerado.

Como diferencial neste trabalho, além de automatizar o *feedback* das submissões das tarefas efetuadas pelos alunos, também tínhamos como objetivo gerar um maior engajamento deles no processo de resolução dos problemas. Portanto, optou-se por desenvolver uma ferramenta voltada para atender os requisitos mencionados anteriormente.

3. Kodesh: uma ferramenta gamificada de auxílio à prática de programação

3.1. Descrição da ferramenta

Foi desenvolvido uma plataforma *online* gamificada com o objetivo de facilitar e estimular a realização de atividades de programação. A plataforma teve como estratégia inicial de gamificação o padrão PBL (*Points, Badges* e *Leaderboards*), bastante criticado pela literatura pela simplicidade, porém fácil de implementar [Kapp 2012]. Por ter sido inicialmente desenvolvida como um protótipo, optamos por testar nossa ferramenta a partir de uma estratégia de gamificação simples. A partir das necessidades detectadas em nossos alunos, poderíamos vislumbrar outras possibilidades.

O conteúdo da disciplina foi organizado por tópicos ao longo de 18 semanas, o que totaliza um semestre acadêmico. A cada semana, o aluno tinha um conjunto de atividades disponibilizadas para se exercitar, com o assunto referente à semana em curso. As atividades são problemas de programação compostas de um enunciado e de um conjunto de testes de entrada para o programa a ser desenvolvido pelo aluno e sua correspondente saída esperada. O aluno deve, então, implementar sua solução na própria ferramenta e submetê-la para validação. Além dos testes disponíveis para o aluno, cada problema contém uma bateria de testes ocultos, que servem para generalizar a solução

esperada. O aluno pode, então, editar e submeter sua resposta, e ter um *feedback* dos erros de compilação, dos testes que falharam ou de sucesso na resolução do problema. Os testes visíveis aos alunos que falhavam são apresentados indicando as diferenças entre as saídas geradas e as esperadas (inclusive espaços em branco e saltos de linha), para uma maior clareza entre a diferença do que foi feito do esperado.

A realização das atividades somava pontos para o aluno, que era classificado em um *ranking* global (envolvendo todas as turmas) e local (envolvendo apenas sua turma). A realização das atividades também rendia "estrelas" para o aluno, fazendo-o evoluir em diferentes categorias de jogador: *Noob*, *Apprendice*, *Professional*, *Master* e *Expert*. As estrelas eram obtidas em função da solução apresentada para atender requisitos de recursos da linguagem (por exemplo, se um problema exigisse uma solução usando o 'while' ou usando recursão) e de tempo (alguns problemas exigem que a solução fosse enviada em um tempo pré-determinado, por exemplo em 20 minutos). Caso o aluno não conseguisse resolver um problema, ele poderia consultar uma possível solução, pré-cadastrada pelos professores. Nesse caso, porém, o aluno não obtinha pontos ou estrelas referentes à questão. A Figura 1 apresenta uma captura de tela da página inicial do Kodesh com os elementos de gamificação aqui indicados.

Figura 1. Tela inicial do Kodesh na qual são apresentadas informações do usuário com seus pontos, estrelas, *ranking* e percentual de realização de atividades cumpridas.

3.2. Avaliação

Foi realizada uma avaliação da aceitação por parte dos alunos da ferramenta proposta, através de um questionário aplicado de forma anônima. O questionário, composto de 9 questões objetivas e 3 questões abertas (texto livre), foi aplicado durante o semestre, antes do período de trancamento de disciplinas, havendo um total de 170 respondentes. Desses, 155 alunos usavam o Kodesh de forma regular para se exercitar (11 não o usavam e 4 não responderam a questão), o que corresponde a 91% do total. Dos que usavam o Kodesh, apenas 12% dos alunos achavam que teriam o mesmo desempenho nas avaliações mesmo se não houvesse a ferramenta. Ou seja, 88% dos alunos tinham em mente que a ferramenta produziu algum impacto no seu rendimento. Da mesma população que usou o Kodesh, nenhum aluno achou que a ferramenta complicou o aprendizado, 9% constatando que o uso da ferramenta não influenciou e 91% constando que a ferramenta tornou o aprendizado mais fácil. A Figura 2 apresenta estes dados graficamente.

Figura 2. Avaliação do aprendizado feita pelos alunos que praticaram a programação com o auxílio do Kodesh.

Em relação às funcionalidades que motivaram a criação do Kodesh, 81% dos alunos se mostraram motivados com os elementos de gamificação presentes na ferramenta, assim como 83% dos mesmos se mostraram motivados com os mecanismos de correção automática, validação de testes e comparação de saídas. Esses percentuais, ilustrados na Figura 3, demonstram que as escolhas pelas funcionalidades foram adequadas para o público-alvo.

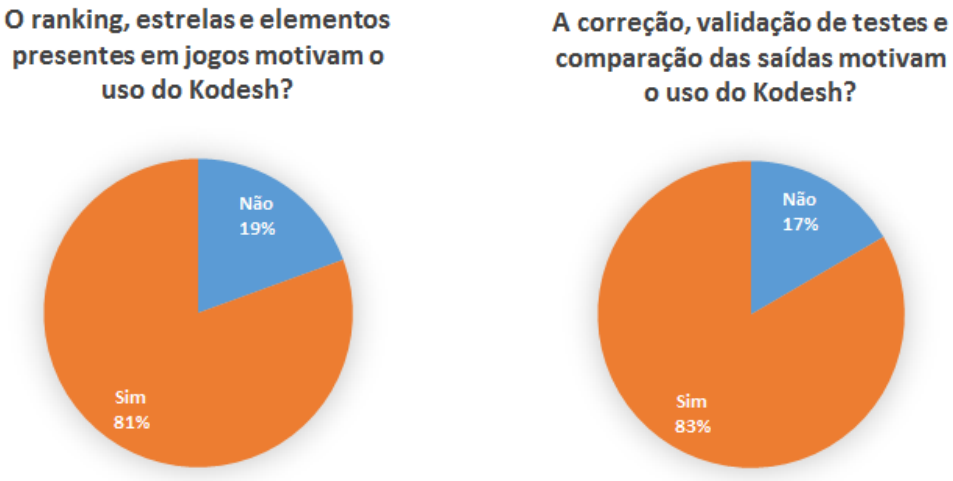

Figura 3. Avaliação das funcionalidades de gamificação e correção automática feita pelos alunos que praticaram a programação com o auxílio do Kodesh.

Além das questões objetivas, também foram disponibilizadas três questões subjetivas nas quais os alunos podiam fazer seus comentários sobre os aspectos positivos e negativos da ferramenta, bem como deixar sugestões para versões futuras. As respostas foram categorizadas e quantificadas. O gráfico à esquerda da Figura 4 apresenta as categorias que obtiveram pelo menos dois comentários positivos. Nela, vemos que a gamificação é a categoria positivamente mais citada pelos alunos, com cerca de 36% dos comentários, seguida por verificação de erros (16%) e interface intuitiva (15%). Nem

todos os alunos sabem, entretanto, o que é gamificação. Seus comentários foram essencialmente sobre elementos presentes na ferramenta sem usarmos diretamente o termo propriamente dito. O gráfico à direita da Figura 4 apresenta o percentual dos comentários positivos dos alunos referentes aos aspectos de gamificação no Kodesh.

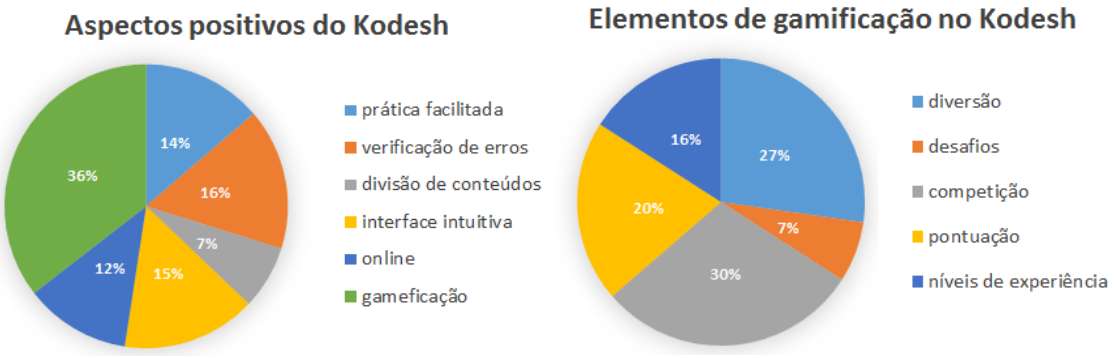

Figura 4. Aspectos positivos agrupados em categorias a partir dos comentários feitos pelos alunos e elementos citados pelos alunos referentes à gamificação.

Além dos aspectos positivos, procuramos avaliar os aspectos negativos. O aspecto negativo mais citado foi a presença de *bugs* no sistema, o que se deve a ser uma versão de protótipo em que os alunos foram os próprios testadores da ferramenta. Em seguida, os dois aspectos negativos mais mencionados foram a correção rigorosa (29%) e o sistema de pontuação confuso (25%). De fato, por serem iniciantes em programação, muitos alunos tinham dificuldade em aceitar que um espaço em branco gerasse uma saída errada. Apesar da ferramenta apresentar claramente a localização do espaço em branco erroneamente existente, o sentimento era que, se a lógica do programa estivesse correta, então o sistema deveria considerá-la. O terceiro ponto negativo deve-se a uma má escolha no sistema de pontuação e, consequentemente, à classificação dos alunos. Optamos inicialmente por deixar a pontuação dinâmica, na qual os pontos de uma atividade eram definidos em função do número de submissões que obtiveram sucesso na atividade. Essa estratégia demostrou ser inconveniente para os alunos que não queriam ver sua pontuação alterar de um dia para o outro sem, para isso, terem realizado alguma atividade.

4. Evolução da ferramenta

Uma das regras essenciais em *game design* é definir um fluxo de jogo que evite no jogador estados de tédio (se os desafios forem extremamente fáceis de serem alcançados) e ansiedade (se os desafios forem dificilmente alcançáveis) [Schell 2014]. O preço de não se estabelecer um fluxo razoável é o desinteresse por parte do jogador. Projetar esse fluxo é um desafio quando lidamos com jogos voltados à educação, em especial quando há uma grande heterogeneidade nas turmas. Em jogos ou sistemas gamificados em formato de campeonato, onde a pontuação obtida em diferentes fases é cumulativa, a meta traçada é, muitas vezes, definida pela posição do jogador no *ranking*. Vimos no Kodesh que, para alguns alunos, ganhar o máximo possível de pontos é algo muito mais instigante que o próprio fato de aprender a resolver problemas (para alguns, estar entre os primeiros do *ranking* é o objetivo principal e aprender é secundário). Constatamos também outros alunos se desmotivarem totalmente pela ferramenta quando começaram a perceber que estar entre os primeiros do *ranking* tinha passado a ser um objetivo inalcançável.

A mecânica de pontuação e classificação precisou, então, ser revista, de forma a permitir diferentes comparações entre os alunos, i.e. diferentes perspectivas sobre a evolução do aprendizado. Foram definidas, por exemplo, a classificação por pontos já obtidos, bem como a dos que mais evoluíram percentualmente na última semana. Essa última mecânica premia de forma relativa quem mais se esforçou nos últimos dias, independente de uma pontuação absoluta. Dividimos também o semestre em três pequenos campeonatos. A cada novo campeonato, a pontuação é reiniciada, dando oportunidade de igualdade aos alunos a cada reinício. Separamos as atividades em "atividades de treinamento", que servem exclusivamente para os alunos se prepararem, sem necessariamente estarem vinculados a um campeonato, e "atividades de campeonato", que possuem tempo e número de submissões limitadas para resolvê-las e contar pontos nos campeonatos. Para facilitar quem tem dificuldade em uma atividade, foi introduzida uma moeda virtual. A cada atividade resolvida ou meta alcançada, os alunos também ganham dinheiro virtual, que podem ser usados para adquirir itens durante a realização de atividades. Exemplos de itens são: tempo, número de submissões, soluções parciais e completas. A Figura 5 apresenta uma das telas da nova versão do Kodesh com alguns elementos citados.

Além dos elementos de gamificação, inserimos na ferramenta um componente social, no qual os alunos podem postar dúvidas e obter respostas de seus colegas. Essa demanda originou-se da observação, ao longo do uso da versão inicial, que outras plataformas (redes sociais) eram usadas em paralelo com o propósito de compartilhar e trocar informações. Constatamos que a interação social é um componente essencial nesse tipo de ferramenta.

Figura 5. Tela principal da nova versão do Kodesh.

4. Resultados e considerações finais

Em 2013.1, houve um total de 118 alunos ingressantes na disciplina de Prática de Técnicas de Programação (PTP), componente correspondente ao laboratório da disciplina inicial de programação do curso. Desses, 110 eram provenientes do novo curso de Bacharelado em Tecnologia da Informação (BTI) e 8 do curso de Ciência da Computação.

No ano anterior, quando o Kodesh ainda não tinha sido introduzido, houve um total de 30 alunos ingressantes na disciplina. Estes números referem-se apenas aos alunos ingressantes (calouros) que seguiram a disciplina até o final. Reprovações por falta, cancelamentos e trancamentos foram desconsiderados. Para efeito de comparação, a média dos alunos em 2013 foi de 5,82 com desvio padrão de 2,73, e a média dos alunos em 2012 de 5,63 com desvio padrão de 2,82. Apesar de positiva, a diferença não foi significativa.

Em 2014, foi introduzida uma disciplina de nivelamento em matemática no semestre inicial, o que levou a disciplina de programação para o segundo semestre. Assim, em 2014.2, houve 78 alunos ingressantes em PTP, cuja média foi de 6,95 com desvio padrão de 2,56. Nesse caso, houve um salto positivo nos resultados finais. Porém, tanto a inclusão de uma disciplina de nivelamento em matemática, como a passagem para o segundo semestre podem ter exercido um forte impacto nos resultados. Assim, devido às variáveis externas, não podemos necessariamente atribuir o resultado de 2014 ao uso do Kodesh. Podemos, entretanto, afirmar que o Kodesh tem alcançado seu objetivo de manter ou melhorar o rendimento das turmas dos anos anteriores, mesmo com um acréscimo significativo do número de alunos.

O Kodesh tem também demonstrado um alto nível de aceitação por parte de professores e alunos. Mesmo a versão inicial, que se baseou em uma estratégia simples de gamificação, demonstrou exercer um impacto positivo sobre os alunos, resultando em índices de aceitação de 81% e 83% para os elementos de gamificação e correção automática, respectivamente. Inúmeras observações foram realizadas sobre o comportamento dos alunos frente às mecânicas de gamificação da versão inicial, de forma a compor um conjunto de requisitos que serviram para guiar a construção de uma nova versão, a ser aplicada em 2015.2.

Referências

Alves, F., Jaques, P. (2014). "Um Ambiente Virtual com Feedback Personalizado para Apoio a Disciplinas de Programação". Em Anais do XXV Simpósio Brasileiro de Informática na Educacão, pages 1078-1082.

Bennedsen, J., Caspersen, M. (2007) "Failure rates in introductory programming", In: ACM SIGCSE Bulletin, vol. 39(2), pages 32-36.

Bergin, S., Reilly, R. (2005). "The influence of motivation and comfort-level on learning to program". In: Proceedings of the 17th Annual Workshop of the Psychology of Programming Interest Group, pages 293-304.

Bez, J.L., Tonin, N.A., Rodegheri, P.R. (2014) "URI Online Judge Academic: A tool for algorithms and programming classes". In: Proceedings of the 9th International Conference on Computer Science Education (ICCSE), pages 149–152.

Campello, A., Lins, L. (2008). "Metodologia de análise e tratamento da evasão e retenção em cursos de graduação de instituições federais de ensino superior". Em Anais do XXVIII Encontro Nacional de Engenharia de Produção, pages 1-13.

Campos, C.; Ferreira, C. (2004). “BOCA: um sistema de apoio a competições de programação”. Em Anais do Workshop de Educação em Computação.

Chaves, J., Castro, A., Lima, R., Lima, M., Ferreira, K. (2013). “MOJO: uma ferramenta para auxiliar o professor em disciplinas de programação”, Em Anais do Congresso Brasileiro de Ensino Superior a Distância.

Gomes, A., Mendes, A. (2007). “Learning to program - difficulties and solutions”. In: Proceedings of the International Conference on Engineering Education.

Gomes, A., Henriques, J., Mendes, A. (2008). “Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores”. Em Educação, Formação & Tecnologias, vol. 1(1), pages 93-103.

Jenkins, T. (2002). “On the Difficulty of Learning to Program”. In Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences, pages 53-58.

Joy, M., Griffiths, N., Boyatt, R. (2005). “The BOSS Online Submission and Assessment System”, ACM Journal on Educational Resources in Computing, v. 5(3), pages 1-28.

Kapp, K. (2012). “The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education”. Pfeiffer.

Lahtinen, E., Ala-Mutka, K., Järvinen, H.-M., (2005). “A study of the difficulties of novice programmers”. In: ACM SIGCSE Bulletin. ACM, pages 14-18.

Resnick, M., Maloney, J., Monroy-Hernandez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y. (2009). “Scratch: Programming for All”. Communications of the ACM 52, pages 60–67.

Santos, J., Ribeiro, A. (2011). “JOnline: proposta preliminar de um juiz online didático para o ensino de programação”. Em Anais do XXII Simpósio Brasileiro de Informática na Educação, pages 964–967.

Schell, J. (2014). “The Art of Game Design: A Book of Lenses”. A K Peters/CRC Press.

Silva, T.; Tedesco, P.; Melo, J. (2014). “A importância da motivação dos estudantes e o uso de técnicas de engajamento para apoiar a escolha de jogos no ensino de programação”. Em Anais do XXV Simpósio Brasileiro de Informática da Educação, pages 11-15.

Vahldick, A., Mendes, A.J., Marcelino, M.J. (2014). “A review of games designed to improve introductory computer programming competencies”. Proceedings of IEEE Frontiers in Education Conference (FIE), pages 1-7.