SimProgramming: uma abordagem motivacional para a aprendizagem de alunos intermediários de programação

Ricardo Rodrigues Nunes[1,2], Daniela Pedrosa[1,3], Leonel Morgado[2,4], Hugo Paredes[1,2], Paulo Martins[1,2], José Cravino[1,3], Carlos Barreira[5]

[1]Universidade de Trás-os-Montes e Alto Douro
(UTAD)
Vila Real, Portugal

[2]Instituto de Engenharia de Sistemas e Computadores - Tecnologia e Ciência
(INESC TEC)
Porto, Portugal

[3]Centro de Investigação em Didática e Tecnologia na Formação de Formadores
(CIDTFF)
Aveiro, Portugal

[4]Universidade Aberta
(UAb)
Lisboa, Portugal

[5]Universidade de Coimbra
(UC)
Coimbra, Portugal

{rrnunes,dpedrosa,hparedes,pmartin,jcravino}@utad.pt,
leonel.morgado@uab.pt, cabarreira@fpce.uc.pt

Abstract. *In this paper, an action research is presented to motivate students to develop their learning of computer programming in higher education, particularly in the transition from beginner to advanced programming. To achieve this goal, a motivational approach was developed called SimProgramming. From the reflections on the process of this research, it is concluded that SimProgramming in its application to the teaching of computer programming in intermediate classes is promising and still presents potential to be used in other educational contexts.*

Resumo. *Neste artigo, é apresentada uma pesquisa-ação com o objetivo de motivar os alunos a desenvolverem suas aprendizagens de programação de computadores no ensino superior, particularmente na transição da programação de nível iniciante para a programação avançada. Para alcançar este objetivo, foi desenvolvida uma abordagem motivacional denominada SimProgramming. A partir das reflexões sobre o processo desta pesquisa, conclui-se que SimProgramming em sua aplicação ao ensino de programação de computadores em turmas intermediárias é promissor e ainda apresenta potencial para ser usado em outros contextos educacionais.*

DOI: 10.5753/cbie.wcbie.2017.1099

1. Introdução

Para os alunos de computação, a aprendizagem de técnicas avançadas de programação é um desafio complexo [Zschaler *et al.* 2014]. Nos cursos introdutórios de programação, geralmente, os alunos aprendem via instrução formal [Robins *et al.* 2003] e alguns tornam-se capazes de desenvolver pequenos programas, bem como adaptar e combinar pedaços de códigos existentes. Porém, eles parecem não entender claramente a importância de escrever códigos bem estruturados a partir de estruturas pré-existentes, como *frameworks*, bibliotecas e Interface de Programação de Aplicações (API do inglês *Application Programming Interface*) [Jenkins 2002]. Este comportamento pode ocorrer devido ao fato de que em situações de programação mais avançadas os alunos precisam desenvolver outro conjunto de conhecimentos, técnicas e competências complexas [Robins *et al.* 2003]. Além das competências de programação necessárias para aplicarem essas boas práticas durante o desenvolvimento de um sistema, os alunos também precisam desenvolver competências sociais a fim de colaborar com outros programadores, como parte do processo baseado em trabalho de equipe para o desenvolvimento de sistemas complexos e de larga escala [Jenkins 2002] [Zschaler *et al.* 2014]. Ao encontro desta realidade, tem sido relatado na literatura que as atuais abordagens de aprendizagem relacionadas à computação não são alinhadas à prática profissional exigida pelo mercado de trabalho [Sheppard *et al.* 2008]. Estas abordagens estão estreitamente centradas na aquisição de conhecimentos técnicos suportados por pesadas cargas de trabalho e promovem um sistema de crenças baseado numa "meritocracia da dificuldade" em vez de priorizarem a aprendizagem ativa com a integração do conhecimento e competências mais alinhada às realidades profissionais [Stevens *et al.* 2008] [Adams *et al.* 2011].

Com objetivo de propor alternativas à instrução formal de programação, as pesquisas sobre a aprendizagem vêm passando por uma mudança de foco, partindo de uma perspectiva de aquisição de conhecimento para uma perspectiva de participação em comunidades de prática [Barab and Duffy 2000]. Por esta ótica, conhecida por aprendizagem situada [Johri and Olds 2011], o conhecimento é situado na própria atividade, no contexto e na cultura no qual o aluno está inserido, em vez de ser visto como um processo de aquisição e manipulação de representações mentais simbólicas. Em paralelo, pesquisas sobre a motivação dos alunos vêm sendo conduzidas através de várias abordagens que focam no comportamento e na cognição humana, bem como em aspectos não-cognitivos, tais como as percepções, as crenças, as atitudes, as emoções *etc.* [Lai 2011]. A motivação é um constructo importante uma vez que ela resulta da interação do indivíduo com uma situação e é a partir dela que o aluno irá desenvolver sua direção (escolha), intensidade (nível de atividade), persistência (tempo despendido) e qualidade (estratégias cognitivas) do comportamento despendido nas atividades relacionadas à sua aprendizagem [Brophy 2010]. É importante ressaltar que o contexto pedagógico em que o aluno aprende influencia o seu engajamento nas atividades de aprendizagem [Johri and Olds 2011] e muitas pesquisas examinam abordagens de aprendizagem baseada em projeto (PBL do Inglês *Project Based Learning*) e ambientes de trabalho em equipe.

2. Do aluno iniciante a programador especialista

Na literatura, é reportado que pode levar cerca de dez anos de experiência para um aluno iniciante tornar-se num programador especialista [Winslow 1996]. Durante esse tempo, os alunos passam por uma transformação contínua, descrita na literatura através de cinco níveis de desenvolvimento: iniciante, iniciante avançado, competente, proficiente e especialista [Dreyfus and Dreyfus 2005].

Durante os dez anos de experiência que são tidos como necessários para se alcançar a mestria de um especialista, o aluno necessita de uma enorme quantidade de prática para o seu desenvolvimento. É reportado que é preciso, independente da área do conhecimento, praticar durante aproximadamente dez mil horas [Gladwell 2008] [Weiss and Shanteau 2014], o que pode ser equivalente a três horas por dia, ou vinte horas por semana, ao longo destes dez anos [Yates and Hattie 2013]. Uma expectativa otimista é que ao fim de uma graduação com duração de quatro anos, o aluno de programação consiga tornar-se competente ou proficiente [Robins *et al.* 2003] [Winslow 1996].

Ao se iniciarem num determinado domínio, os alunos iniciantes e os iniciantes avançados passam por um período de aclimatação, onde necessitam de orientações para determinarem os conteúdos que são centrais ou periféricos para não desenvolverem um conhecimento fragmentado devido ao pouco investimento pessoal despendido sobre o domínio. Estes alunos também precisam de auxílio para definirem estratégias para resolução de problemas, pelo fato de as suas estratégias serem muitas vezes ineficazes quando deixados por conta própria [Alexander 2003].

Geralmente, depois de completarem sua educação formal e a sua formação básica, os alunos tornam-se profissionais e trabalham como aprendizes (por vezes até formalmente, com a designação “estagiários”) e são supervisionados por profissionais mais experientes. Pouco tempo depois, podem alcançar um nível aceitável de competência. Muitos destes profissionais mantêm esse patamar de desempenho ao longo da carreira, porém alguns continuam a melhorar até se tornarem especialistas [Ericsson 2006]. É importante destacar que as limitações físicas ou cognitivas podem influenciar negativamente o desenvolvimento da especialização. Por outro lado, um estado afetivo equilibrado e o interesse individual ou situacional, refletido na motivação, contribuem significativamente para o seu desenvolvimento [Alexander 2003]. Outros fatores de suma importância são a prática deliberada e o *coaching* [Ericsson *et al.* 2007]. Em outras palavras, a realização de atividades especificamente projetadas para a melhoria e o acompanhamento do desempenho por parte de um professor ou mentor são muito úteis para o desenvolvimento da especialização. Além disso, a comunicação oral e escrita, bem como as competências para o trabalho em equipe são consideradas fundamentais para os futuros informáticos [Adams *et al.* 2011]. Assim, muitos autores propõem atividades orientadas a projetos e o trabalho em equipe para a motivação e o desenvolvimento de competências que permitam a resolução de problemas mais complexos e a capacidade para inovação que são necessárias para ingressarem no mercado de trabalho [Johri and Olds 2011].

3. Trabalhos relacionados

Um trabalho relacionado com a pesquisa aqui apresentada [Pascual 2010] descreve uma extensão da aprendizagem baseada em projetos para aumentar o desenvolvimento social

do conhecimento e da aprendizagem. Sua abordagem visa maximizar as oportunidades dos alunos de compartilharem o conhecimento com outros programadores, a fim de unir a academia e as comunidades de prática. Várias atividades foram desenvolvidas durante o estudo de Pascual, incluindo encontros entre comunidades de estudantes e engenheiros profissionais, atividades acadêmicas e recreativas, tanto dentro quanto fora do campus, bem como o desenvolvimento de um sistema de suporte à decisão baseado na *Web*. O autor levantou a hipótese de que esta abordagem multimodal aumenta a aprendizagem ativa e as relações sociais. A partir dos resultados desta pesquisa foi identificado o aumento da motivação dos alunos, confirmando assim que as comunidades de prática e as necessidades de pertencimento são fatores relevantes para os resultados de aprendizagem.

Outro trabalho relacionado tem sido desenvolvido numa unidade curricular lecionada na Faculdade de Engenharia da Universidade do Porto (FEUP) [Challenge 2017]. Na unidade curricular, denominada Laboratório de Gestão de Projetos (LGP), estudantes de licenciatura e mestrado desenvolvem projetos para empresas de negócios reais. Durante 1 semestre, os estudantes dividem-se em equipas e criam empresas para responder a problemas de clientes parceiros do projeto. As atividades estão divididas em 4 fases com duas apresentações públicas em momentos chaves ao longo do projeto. O projeto é iniciado pela fase de arranque onde as empresas são criadas a partir de *kick-off meetings*. A segunda fase é a concepção, onde são realizados, além de planeamentos semanais sobre os projetos, o controlo de qualidade e a gestão de riscos, acompanhados por relatórios semanais. No final desta fase são feitas as apresentações intermediárias em sessões públicas. A terceira fase é o desenvolvimento em si, também com a utilização de planejamento e relatórios semanais, além de controle de qualidade e gestão de riscos. Após esta fase é a apresentação final, também em sessões públicas. E, por fim, a fase de encerramento, com autoavaliação e reunião de entrega do projeto com os clientes. Durante todo o projeto os estudantes desenvolvem conhecimentos sobre gestão de projetos, empreendedorismo, *marketing*, comunicação, interação com o cliente e trabalho de equipe em projetos de *software*.

4. Método de pesquisa

Foi realizada uma pesquisa-ação no âmbito da Unidade curricular "Metodologias de Programação III" (MPIII), ao longo de 3 anos, na Universidade de Trás-os-Montes e Alto Douro (UTAD) em Portugal. A unidade curricular é uma cadeira obrigatória do segundo ano curricular dos programas de graduação em Engenharia Informática (IE) e em Tecnologias de Informação e Comunicação (TIC) que introduz conceitos de arquitetura de *software* para apoiar o desenvolvimento dos alunos em suas habilidades de organização de código.

Uma abordagem foi desenvolvida a partir de um conjunto de problemas relacionados com temas abordados na unidade curricular. Na Tabela 1, são apresentadas algumas instâncias dos problemas que os alunos, em grupos, deveriam propor uma solução. As atividades baseavam-se em aprendizagem baseada em problemas, onde foram atribuídos a cada grupo um problema específico envolvendo um padrão arquitetônico relacionado com MVC (*Model-View-Controller*) e estruturas pré-existentes, tais como *frameworks*, bibliotecas ou Interface de Programação de Aplicações. No final do projeto, os grupos deveriam desenvolver um documento escrito,

explicando em detalhes as abordagens de codificação utilizadas para aplicar tal padrão arquitetônico com a estrutura pré-existente relacionados ao problema atribuído.

Tabela 1. Alguns dos problemas específicos atribuídos às equipes de alunos

Problemas
Escrever um documento pormenorizado que explique como aplicar o padrão arquitetônico **MVC** (*Model*-*View*-*Controller*) ao desenvolvimento de aplicações com a **libOpenMetaverse**. Esse documento deve complementar essa explicação com exemplos concretos das várias formas de aplicação que concebam.
Escrever um documento pormenorizado que explique como aplicar o padrão arquitetônico **MVP** (*Model*-*View*-*Presenter*) ao desenvolvimento de aplicações na Windows Phone Application Platform, com a *framework* **XNA**. Esse documento deve complementar essa explicação com exemplos concretos das várias formas que concebam para aplicação do padrão.
Escrever um documento pormenorizado que explique como aplicar o padrão arquitetônico **MVVM** (*Model*-*View*-*ViewModel*) ao desenvolvimento de aplicações com **Windows Forms**. Esse documento deve complementar essa explicação com exemplos concretos das várias formas que concebam para aplicação do padrão.

Ao longo do projeto de pesquisa, foram recolhidos dados quantitativos e qualitativos a partir de arquivos submetidos e *logs* da utilização nos sistemas de informação adotados na unidade curricular (*e.g. Wiki* PBWorks e Moodle), de questionários disponibilizados *online*, de entrevistas semiestruturadas, de gravações audiovisuais de algumas atividades presenciais e das observações dos pesquisadores.

Na primeira iteração [Morgado *et al.* 2012], os alunos foram desafiados a resolver o problema atribuído com componentes teóricos e práticos através de pesquisa da literatura técnico-científica, além do envolvimento com programadores mais experientes nas redes sociais e/ou em comunidades de prática *online*. O objetivo era que os alunos encontrassem motivação para desenvolverem suas competências de programação a partir das discussões com outros programadores em estágios mais avançados de conhecimento que encontrariam nas comunidades. Os resultados desta primeira iteração mostraram que a maioria dos grupos foram incapazes de resolver suas atribuições com sucesso. Dos vinte grupos (total de 62 alunos) que iniciaram o projeto, dezenove grupos (total de 59 alunos) tiveram alguma atividade. Sendo que sete grupos realizaram as atividades durante todas as fases do projeto, com quatro deles alcançando um nível aceitável de qualidade. A qualidade global dos relatórios dos grupos que finalizaram o projeto demonstrou que os alunos tinham absorvido aspectos significativos sobre os temas das atribuições propostas e foram capazes de fornecer exemplos úteis sobre as abordagens que desenvolveram, porém, este resultado não se refletiu nos resultados finais da unidade curricular, que eram muito mais dependentes dos testes do que do projeto em si, cujo peso na nota final foi de 20%.

A segunda iteração ocorreu no ano seguinte [Nunes *et al.* 2015]. As atividades foram implantadas com novas estratégias pedagógicas: os componentes das atividades foram mais estruturados ao longo do projeto (tarefas semanais com prazos), havia dois tutores para fornecerem apoio aos alunos através de acompanhamento e *feedback*, além de três dinâmicas de grupo conduzidas dentro do auditório onde eram ministradas as aulas. Em comparação ao ciclo anterior, verificou-se que mais grupos participaram

ativamente ao longo de todo o projeto. Mais especificamente, nove dos vinte e um grupos que iniciaram o projeto desenvolveram regularmente suas atividades e obtiveram algum *feedback* nas comunidades de prática *online*. A partir dos resultados desta segunda iteração, foi possível ter a percepção de possíveis soluções para os principais problemas sentidos pelos alunos, nomeadamente a falta de tempo para trabalharem, a falta de *feedback* sobre o desenvolvimento do projeto e a baixa motivação para completarem as atividades. A partir destes resultados, foi elaborada uma abordagem motivacional denominada SimProgramming que simula um contexto de negócios para ser implementada no terceiro ciclo desta pesquisa.

5. Abordagem motivacional do SimProgramming

Algumas mudanças significativas foram implementadas na terceira iteração que também foi relatada num artigo [Pedrosa *et al.* 2016c]. Uma delas foi a criação de uma comunidade de prática com alunos, ex-alunos e programadores externos convidados, em vez de pedir aos alunos que apenas participassem de comunidades externas, buscando assim, promover uma maior interação entre os participantes do projeto. Também foram adotadas melhores práticas de gestão de projetos, a fim de identificar problemas numa fase anterior, permitindo uma melhor orientação por parte da equipe docente para apoiar as necessidades dos alunos e, consequentemente, ajudá-los para obterem melhores resultados. Consequentemente, foi proposto que a interação e as estratégias de avaliação pedagógica fossem reformuladas para simular um ambiente empresarial, incluindo o método de gerenciamento de projetos, conhecido por SCRUM [Schwaber 2004]. Ainda foram incluídos outros aspectos de um ambiente profissional, o trabalho em equipe mais gerenciável, o *feedback* contínuo e estratégias de autoavaliação e heteroavaliação.

5.1. Fundamentos conceituais

A abordagem do SimProgramming está baseada em quatro fundamentos conceituais: (1) simulação um ambiente empresarial para aprendizagem (2) aprendizagem ativa; (3) aprendizagem situada e; (4) avaliação formativa.

5.1.1. Simulação de um ambiente empresarial para aprendizagem

Nesta simulação, cada participante assume um papel. O professor desempenha o papel de diretor geral, globalmente, responsável por avaliar o andamento dos projetos, esclarecer as dúvidas pontuais que os estudantes apresentem e orientá-los a partir das análises feitas sobre as apresentações nos encontros presenciais realizados e nos relatórios submetidos. Os tutores, ou assistentes de ensino, assumem o papel de gerentes de projeto, sendo responsáveis por um acompanhamento mais próximo, fornecendo *feedback* constante e exercendo mentoria. Em todas as equipes, um dos estudantes exerce o papel de coordenador, auxiliando os gestores de projeto (diretor e gerentes). Dentre as suas atribuições, ele necessita de assegurar a integração dos participantes, fazer com que os membros mantenham uma visão global do contexto do projeto e o seu *status*, bem como necessita de ser o elo de comunicação entre os gestores e as equipes. Os coordenadores, tal como os gerentes de projetos, são agentes motivadores para os demais estudantes. A proposta da simulação de um ambiente empresarial é baseada no que é reportado na literatura sobre a influência que o contexto pedagógico exerce na

motivação dos estudantes. Inclusive, com a integração de projetos que empregam a aprendizagem baseada em problemas e o trabalho em equipe.

5.1.2. Aprendizagem ativa

O SimProgramming se apresenta favorecendo a aprendizagem ativa e promovendo a rotina de estudo dos estudantes através das entregas semanais das tarefas, fazendo com que eles estejam sempre debruçados nas tecnologias relacionadas à resolução dos problemas. Os formulários semanais ainda permitem que os alunos reflitam sobre seus trabalhos, ponderando o que fazer na semana seguinte e identificando os fatores que os impedem de atingir os objetivos individuais e da equipe. É importante destacar que nesta abordagem são considerados o histórico de procrastinação para a realização das atividades e a necessidade de gestão do tempo que os estudantes apresentam. Assim, eles são constantemente encorajados a adotarem rotina de estudos, através da promoção de um contexto motivacional para a aprendizagem. Dessa forma, os alunos têm a possibilidade de desenvolver gradualmente o conceito de fazerem as suas atividades regularmente e não deixarem a execução para o último momento apenas.

5.1.3. Aprendizagem situada

Os estudantes com pouca experiência têm ideias gerais sobre a área e quando adquirem novas responsabilidades, realizando tarefas mais complexas, vão desenvolvendo, aos poucos, suas identidades perante aos outros participantes nas comunidades de prática. Conforme vão interagindo com os demais, vão aumentando suas oportunidades para aprender através das suas próprias contribuições. Assim, é importante que a equipe docente preste apoio, orientando a maneira que os estudantes devem participar e se envolver, de forma gradual, nestas comunidades, inclusive sugerindo formas específicas de interações. Este apoio ao desenvolvimento do sentimento comunitário dos estudantes, interações informais e debates também podem ser promovidos e monitorados através de um grupo do Facebook, criado especificamente para o curso. Esta comunidade interna pode ser mantida ao longo dos anos, tornando-se uma fonte de interação entre novos alunos, os ex-alunos ainda em formação e os alunos formados já inseridos no mercado de trabalho. Alguns desses ex-alunos podem desempenhar o papel de consultores de negócios, fornecendo apoio e aconselhamento para as novas equipes.

5.1.4. Avaliação formativa

Este processo inclui mentoria motivacional e *feedback* constante sobre o *status* do desenvolvimento do projeto, por exemplo, se o trabalho está progredindo ou desviando das expectativas e objetivos. Para isso, são realizadas reuniões semanais baseadas no SCRUM, método ágil para planejamento e gestão de projetos de desenvolvimento de *software*. Este apoio é presencial e as reuniões semanais são programadas com os coordenadores das equipes onde são abordados 3 tópicos específicos: 1) “o que a equipa fez durante a semana passada?”; 2) “o que será feito durante a semana corrente?” e; 3) “o que está impedindo a conclusão das atividades?”. Ao serem detectados problemas específicos (técnicos, pessoais ou outros), são agendadas reuniões com as equipes que apresentaram tais problemas. Além destas reuniões, com o objetivo de cruzar informações, todos os alunos precisam responder aos mesmos 3 tópicos específicos num formulário e submetê-lo, semanalmente, na plataforma *online* adotada. Além destes

relatórios, os estudantes ainda submetem relatórios sobre as suas interações nas comunidades de prática em duas ocasiões específicas do projeto (final das Fases 1 e 2). Estes relatórios também servem como instrumentos formativos. No final do projeto, os estudantes fazem a suas autoavaliações e as heteroavaliações dos seus colegas de equipe. Todos estes relatórios são excelentes oportunidades para os alunos refletirem sobre as suas atividades. Ainda são realizadas 3 apresentações presenciais, no final de cada uma das fases do projeto. Estás apresentações são avaliadas pela equipe docente e servem como recurso formativo para orientação sobre o andamento dos projetos dos estudantes. É importante destacar que cada encontro com os alunos, seja em sala de aula ou nos corredores da universidade, é uma oportunidade para conversar informalmente sobre o projeto e orientá-los sobre os mesmos.

5.2. Aplicação

A abordagem do SimProgramming desenvolve-se ao longo de 4 fases, com base nos fundamentos conceituais apresentados anteriormente. Estas fases representam o resultado das reflexões sobre todo o processo de pesquisa-ação. Na Tabela 2 é apresentada um resumo sobre as atividades realizadas durante as quatro fases.

Tabela 2. Resumo da abordagem do SimProgramming

Fases	**Atividades**
Fase 1 (concepção)	Organização das equipes; levantamento da literatura; interação nas comunidades de prática; apresentação inicial; relatório semanal.
Fase 2 (desenvolvimento)	Interação nas comunidades de prática; apresentação intermediária; relatório semanal; relatório das interações nas comunidades.
Fase 3 (refinamento)	Apresentação final; relatório final
Fase 4 (encerramento)	Relatório final melhorado; autoavaliação e heteroavaliação

Ao longo de todas as fases do projeto, são realizadas reuniões semanais entre os gerentes de projeto (tutores) e os coordenadores das equipas, proporcionando suporte a motivação e apoio para dúvidas técnicas. Ao serem identificados problemas internos em alguma das equipas ou situações de risco, os gerentes de projeto também podem agendar reuniões com estas equipes ou com algum aluno específico para fazerem intervenções mais direcionadas. Os estudantes ainda submetem relatórios no final de cada *sprint*, com duração de 1 semana, detalhando o que fizeram, o que irão fazer e o que, eventualmente, está impedindo a realização das atividades. E, em duas ocasiões (final da Fase 1 e Fase 2), os estudantes ainda submetem relatórios sobre as suas interações nas comunidades de prática.

6. Resultados e conclusão

Especificamente no terceiro ciclo desta pesquisa-ação, onde o SimProgramming foi implementado, os dados foram recolhidos a partir dos formulários preenchidos e submetidos, das anotações sobre as observações dos investigadores, das gravações audiovisuais dos encontros presenciais e dos registos das trocas de mensagens, bem como dos *logs* das interações nos sistemas de informação adotados. Durante a execução da abordagem e a partir dos seus resultados foi possível levantar evidências da sua adequação e concluiu-se que o SimProgramming mostrou ser um importante instrumento motivacional neste contexto de pesquisa. Neste ciclo, mais alunos participaram ativamente do projeto e os resultados se refletiram nas notas finais da unidade curricular. Das quinze equipes, onze mantiveram atividades durante todo o projeto. Das quatro equipas restantes, duas trabalharam regularmente, porém a qualidade dos relatórios estava aquém do esperado e as outras duas equipas nem iniciaram o projeto. Num total de 97 alunos, 66 completaram as tarefas e 59 apresentaram bom desempenho nas atividades de aprendizagem.

É importante ressaltar que a abordagem do SimProgramming implicou em mudanças significativas na forma tradicional de relacionamento entre docentes e alunos, dentro e fora da sala de aula. A manutenção de uma boa convivência garantiu um ambiente saudável que foi de suma importância para o desenvolvimento do aprendizado dos alunos. Ao longo de todo o projeto foi construída uma relação de confiança entre equipe docente e os estudantes.

Assim, como principal contribuição da presente pesquisa, a abordagem do SimProgramming é apresentada como ferramenta para dar suporte a motivação para aprendizagem de programação de computadores na transição da programação de nível básico para o avançado. Outro contributo é a descrição de todo o processo de pesquisa-ação, que culminou na abordagem proposta, para que educadores possam utilizá-lo como um guia para o desenvolvimento de abordagens pedagógicas inovadoras em diferentes contextos de Educação em Informática.

Após os três ciclos de pesquisa-ação, reportados na tese de doutorado do primeiro autor do artigo aqui apresentado, a abordagem do SimProgramming foi utilizada em outros contextos, tanto para a prática de ensino quanto para pesquisa. A abordagem foi estendida, no âmbito de outro doutorado [Pedrosa 2017], sendo desenvolvida em outra unidade curricular da UTAD, denominada "Metodologias de Programação IV" (MPIV) [Pedrosa *et al.* 2016a] [Pedrosa *et al.* 2016b] [Pedrosa *et al.*

2017]. A abordagem ainda foi adaptada na unidade curricular "Plataformas Sociais e Colaborativas", também da UTAD, além de ter sido transposta para a modalidade a distância, no âmbito de um mestrado, numa unidade curricular denominada "Desenvolvimento de *Software*" num curso de graduação a distância na Universidade Aberta em Portugal.

7. Trabalhos futuros

A principal proposta de trabalho futuro é o refinamento da abordagem do SimProgramming. Algumas propostas mais específicas para este refinamento, podem ser sugeridas de antemão: (1) a adaptação do SimProgramming ao contexto de outras universidades, inclusive em outras unidades curriculares que não sejam relacionadas à programação; (2) a análise da viabilidade de retirar a palavra "problema" do projeto e trocá-la para "desafio". Esta mudança ocorreria pela possível carga negativa que a palavra "problema" poderia trazer para alguns dos alunos. Uma possível solução seria usar aprendizagem baseada em desafios (CBL do inglês *Challenge Based Learning*) no lugar da aprendizagem baseada em problemas, uma vez que CBL prioriza a relevância do projeto para os alunos já que os mesmos são os que identificam os problemas a serem abordados; (3) a introdução de mais dinâmicas de grupo. As dinâmicas realizadas no segundo ciclo desta pesquisa foram bem aceitas pelos alunos e mostraram-se úteis durante a abordagem, porém não puderam ser mais exploradas por causa do escopo do projeto e; (4) aplicar técnicas de ludificação/gameficação no projeto, tais como atribuição de pontos, incrementação de níveis e emblemas de conquistas para estimular a motivação dos alunos.

Agradecimentos

Nunes, R. R. & Pedrosa D. agradecem a Fundação para a Ciência e Tecnologia (FCT), Portugal, pelas bolsas de doutorado SFRH/BD/91309/2012 e SFRH/BD/87815/2012.

Referências

Adams, R., Evangelou, D., English, L., De Figueiredo, A. D., Mousoulides, N., Pawley, A. L., Schiefellite, C., Stevens, R., Svinicki, M., Trenor, J. M. and Wilson, D. M. (2011), Multiple Perspectives on Engaging Future Engineers. Journal of Engineering Education, 100: 48–88. doi: 10.1002/j.2168-9830.2011.tb00004.x.

Alexander, P. A. (2003). The development of expertise: The journey from acclimation to proficiency. Educational Researcher, 32(8), 10-14.

Barab, S. A., & Duffy, T. (2000). From practice fields to communities of practice. Theoretical foundations of learning environments, 1(1), 25-55.

Brophy, J. E. (2010). Motivating students to learn (3rd ed.). New York, NY: Routledge.

Challenge, L. (2017). Laboratório de Gestão de Projetos. 2017.

Dreyfus, H. L., & Dreyfus, S. E. (2005). Peripheral vision expertise in real world contexts. Organization studies, 26(5), 779-792.

Ericsson, K. A., Prietula, M. J., & Cokely, E. T. (2007). The making of an expert. Harvard Business Review, 85(7/8), 114.

Gladwell, M. (2008). Outliers: The story of success: Hachette UK.

Jenkins, T. (2002) On the difficulty of learning to program, Proceedings of 3rd annual conference of the LTSN-ICS, 27–29 August 2002, Loughborough (University of Ulster, LTSN Centre for Information and Computer Sciences).

Johri, A., & Olds, B. M. (2011). Situated engineering learning: Bridging engineering education research and the learning sciences. Journal of Engineering Education, 100(1), 151-185.

Lai, E. R. (2011). Motivation: A literature review Pearson's Research Reports New Jersey.

Morgado, L., Fonseca, B., Martins, P., Cruz, G., Maia, A. M., Nunes, R. R., & Santos, A. (2012). Social networks, microblogging, virtual worlds, and Web 2.0 in the teaching of programming techniques for software engineering: A trial combining collaboration and social interaction beyond college. In Proceedings of the Global Engineering Education Conference (EDUCON) (pp. 1-7), Marrakech, Morocco.

Nunes, R. R., Pedrosa, D., Fonseca, B., Paredes, H., Cravino, J., Morgado, L., & Martins, P. (2015). Enhancing students' motivation to learn software engineering programming techniques: a collaborative and social interaction approach. In M. Antona & C. Stephanidis (Eds.), Universal access in human-computer interaction: access to learning, health and well-being (pp. 189-201). Switzerland: Springer International Publishing.

Pascual, R. (2010). Enhancing project-oriented learning by joining communities of practice and opening spaces for relatedness. European Journal of Engineering Education, 35(1), 3-16.

Pedrosa, D. (2017). Autorregulação e corregulação das aprendizagens no ensino superior: estratégias adotadas por alunos de programação de computadores. 2017. 317 f. Tese (Doutoramento em Didática de Ciências e Tecnologias) - Universidade de Trás-os-Montes e Alto Douro, Vila Real - Portugal. 2017.

Pedrosa, D., Cravino, J., Morgado, L., & Barreira, C. (2016a). Self-regulated learning in computer programming: strategies students adopted during an assignment. In International Conference on Immersive Learning (pp. 87-101). Springer International Publishing.

Pedrosa, D., Cravino, J., Morgado, L., & Barreira, C. (2016b). Self-regulated learning in higher education: strategies adopted by computer programming students. In Proceedings of the PAEE/ALE'2016, 8th International Symposium on Project Approaches in Engineering Education (PAEE) and 14th Active Learning in Engineering Education Workshop (ALE) (pp. 588-595). PAEE–Project Approaches in Engineering Education Association/Universidade do Minho.

Pedrosa, D., Cravino, J., Morgado, L., Barreira, C., Nunes, R. R., Martins, P., & Paredes, H. (2016c). SimProgramming: the development of an integrated teaching approach for computer programming in higher education. In INTED 2016-10th International Technology, Education and Development Conference: Proceedings (pp. 7162-7172). IATED Academy.

Pedrosa, D., Cravino, J., Morgado, L., & Barreira, C. (2017). Self-regulated learning in higher education: strategies adopted by computer programming students when supported by the SimProgramming approach. Production, 27(spe), e20162255. http://dx.doi.org/10.1590/0103-6513.225516

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. Computer Science Education, 13(2), 137-172. doi: 10.1076/csed.13.2.137.14200.

Schwaber, Ken (2004). Agile project management with Scrum. Microsoft Press.

Sheppard, S. D., Macatangay, K., Colby, A., & Sullivan, W. M. (2008). Educating engineers: Designing for the future of the field (Vol. 2): Jossey-Bass.

Stevens, R., O'Connor, K., Garrison, L., Jocuns, A., & Amos, D. M. (2008). Becoming an engineer: Toward a three dimensional view of engineering learning. Journal of Engineering Education, 97(3), 355-368.

Weiss, D. J., & Shanteau, J. (2014). Who's the Best? A Relativistic View of Expertise. Applied Cognitive Psychology, 28(4), 447-457. doi: 10.1002/acp.3015.

Winslow, L. E. (1996). Programming pedagogy - a psychological overview. SIGCSE Bull., 28(3), 17-22. doi: 10.1145/234867.234872.

Yates, G., & Hattie, J. A. (2013). Experts amongst us: What do we know about them? The Journal of Educational Enquiry, 12(1).

Zschaler, S.; Demuth, B. & Schmitz, L. (2014). Salespoint: A Java framework for teaching object-oriented software development. Science of Computer Programming, Volume 79, pp. 189-203.