Educação, Formação & Tecnologias (janeiro-junho, 2015), 8 (1),13-27
Submetido: março, 2015 / Aprovado: maio, 2015

À procura de um contexto para apoiar a aprendizagem inicial de programação

ANABELA DE JESUS GOMES[i]
Instituto Superior de Engenharia de Coimbra e Centro de Informática e Sistemas da Universidade de Coimbra, Portugal
anabela@isec.pt

ANTÓNIO JOSÉ MENDES[ii]
Centro de Informática e Sistemas da Universidade de Coimbra, Portugal
toze@dei.uc.pt

Resumo: As elevadas taxas de insucesso em disciplinas introdutórias de programação são um problema comum por todo o mundo, motivando muitos investigadores a propor metodologias e ferramentas na tentativa de resolver o problema. No entanto, o panorama tem-se mantido praticamente inalterado. Têm sido referidas na literatura muitas causas para esses problemas. O nosso grupo de investigação tem vindo a realizar um conjunto de experiências para identificar e compreender as dificuldades dos alunos em disciplinas introdutórias de programação. Este documento inclui os resultados de um conjunto de estudos que realizamos nos últimos anos com esse intuito. Os resultados serviram de base para definir uma nova abordagem pedagógica que consiste num conjunto de práticas educativas, procurando criar contextos de aprendizagem que motivam os alunos, aumentam o seu envolvimento com as atividades do curso, e maximizam as suas possibilidades de aprendizagem. É também apresentada a aplicação de um método pedagógico num ambiente real e são descritos os resultados obtidos com a sua aplicação ao longo dos últimos anos.

Palavras-chave: Educação em Ciências Computacionais, Estudantes de Engenharia, Ensino da Engenharia, Resolução de Problemas, Motivação, Taxonomias de Aprendizagem.

1. INTRODUÇÃO

Muitas são as dificuldades que os estudantes habitualmente enfrentam ao aprender a programar pela primeira vez. As causas dessas dificuldades têm sido objeto de estudo, sendo apontadas diferentes razões. A maioria dos autores concorda que aprender a programar requer competências como abstração, generalização, transferência, pensamento crítico, resolução de problemas, entre outras (Lister, 2000; Byrne & Lyons, 2001; Lahtinen, Ala-Mutka & Järvinen, 2005; Bennedsen & Caspersen, 2006).

Para ser capaz de programar, o estudante deve adquirir capacidades que vão muito além do conhecimento da sintaxe de uma linguagem de programação. Há a perceção de que o maior obstáculo de muitos estudantes consiste em saber por onde começar a resolver um problema e quais os passos necessários para a solução. Ou seja, a principal questão relacionada com as dificuldades iniciais de aprendizagem de programação coloca-se ao nível do desenvolvimento de algoritmos. Essas dificuldades não são

específicas de qualquer linguagem de programação ou mesmo do paradigma de programação utilizado. Antes, considera-se que essas dificuldades advêm da incapacidade dos estudantes em resolver problemas, em especial os que apresentam uma base matemática. Têm sido propostas diversas abordagens para minorar estas dificuldades, como seja a utilização de técnicas de resolução de problemas, de desenvolvimento do pensamento crítico, aprendizagem baseada em problemas, novas estratégias motivacionais, abordagens ativas, jogos, aprendizagem cooperativa e colaborativa (García & Moreno, 2004; Fagin et al., 2006; Moura & van Hattum-Janssen, 2011; Liu, Cheng & Huang, 2011; Hwang, Wu & Chen, 2012; Hwang et al., 2012; Tasneem, 2012; Verdú et al., 2012; Mills & Treagust, 2013; van Merriënboer, 2013).

Este trabalho parte do pressuposto de que é possível ajudar os estudantes com dificuldades em aprender a programar. Para isso, é fundamental que sejam criadas condições adequadas à aprendizagem de cada estudante. Considera-se que os métodos de ensino mais tradicionais não são os mais adequados para apoiar a aprendizagem de programação, nomeadamente devido às turmas com número elevado de estudantes e à pouca produtividade das aulas expositivas no campo da programação. Assim, a criação de contextos motivadores para a aprendizagem e a utilização de métodos de ensino adequados são fundamentais para um suporte adequado aos estudantes, em particular os que apresentam algumas dificuldades de aprendizagem. Foi com base nestes pressupostos que os autores deste documento realizaram um conjunto de estudos apresentados na secção II deste trabalho. Os resultados obtidos estiveram na base de um conjunto de princípios que deram origem a uma estratégia pedagógica descrita na secção III. Esta estratégia tem vindo a ser aplicada nos últimos anos com resultados satisfatórios, como se descreve na secção IV. Finalmente, na secção V apresentam-se as conclusões deste trabalho.

2. ESTUDOS

2.1 Resolução de problemas e bases matemáticas

Nesta investigação pretendeu-se estudar a relação entre as dificuldades dos estudantes de programação na conceção de algoritmos e a falta de competências de resolução de problemas. Para estudar esse aspeto realizaram-se dois estudos. O primeiro, estudo A (Gomes et al., 2006), envolveu um grupo de 33 estudantes voluntários da Licenciatura em Engenharia Informática e de Sistemas do Instituto Superior de Engenharia de Coimbra (LEIS-ISEC). Os estudantes envolvidos tinham reprovado à primeira disciplina de programação, Algoritmos e Programação-AP, no primeiro semestre de 2005/2006. Assim, durante o segundo semestre desse ano, os estudantes seguiram um curso especial de resolução de problemas. Pretendia-se estudar a relação entre as variáveis capacidade de resolução de problemas e competências matemáticas e as dificuldades de aprendizagem de programação. A experiência incluiu várias sessões em que os estudantes resolviam problemas de base matemática e lógica, com alguma relação com programação, bem como um teste inicial e outro final de programação. Os testes e as atividades realizadas durante as sessões confirmaram que os estudantes apresentavam capacidades de programação muito limitadas, bem como uma ausência quase completa das aptidões matemáticas esperadas em estudantes que ingressam no ensino superior. Gomes et al. (2006) apresentam e discutem detalhadamente os resultados obtidos neste estudo. Em geral, pode dizer-se que os estudantes demonstraram muitas dificuldades de carácter matemático (falta de conceitos básicos; dificuldades de cálculo; dificuldade em traduzir a descrição textual de um problema numa fórmula matemática que o resolvesse; falta de conhecimentos básicos sobre figuras geométricas e sobre trigonometria; dificuldade de orientação no plano cartesiano; incapacidade de aplicação de conceitos na resolução de problemas). Para além disso, foram encontradas dificuldades de interpretação do enunciado

dos problemas, fracos níveis de abstração e falta de treino lógico e de raciocínio. Neste estudo, a intenção das sessões era não só a de identificar pontos fracos na resolução de problemas, mas também proporcionar aos estudantes alguma formação neste domínio, verificando a sua influência sobre a sua aprendizagem de programação. No entanto, os resultados obtidos no teste final de programação foram desencorajadores. Aparentemente, a prática de resolução de problemas realizada não gerou benefícios significativos. Talvez os exercícios feitos não tenham sido suficientes para produzir um impacto relevante em nenhum dos domínios (resolução de problemas e programação).

A influência da capacidade de resolução de problemas e das capacidades matemáticas na aprendizagem da programação foi o objeto de outro estudo (estudo B), realizado no ano letivo de 2007-2008, envolvendo dois grupos de estudantes (Gomes & Mendes, 2008). O primeiro grupo (Grupo_C) incluiu 87 estudantes matriculados na LEIS-ISEC. O segundo grupo (Grupo_Python) incluiu 102 estudantes matriculados na Licenciatura em Engenharia Informática da Universidade de Coimbra (LEI-UC). Todos os estudantes envolvidos estavam pela primeira vez no ensino superior. O primeiro grupo usou a linguagem C, enquanto que o segundo grupo usou Python. Neste estudo foram utilizadas análises quantitativas descritivas e de correlações entre os aspetos questionados considerados de interesse e os resultados obtidos pelos estudantes nas disciplinas introdutórias de programação. Os instrumentos utilizados neste estudo foram um inquérito e um teste de diagnóstico. Foram obtidos dados demográficos, dados académicos e algumas informações sobre o conhecimento de base dos estudantes em áreas que se pensou poderem influenciar a aprendizagem de programação dos estudantes. Foram analisados parâmetros como a nota média de entrada dos estudantes no ensino superior, a classificação obtida na prova de acesso de Matemática, a capacidade de resolução de problemas e o conhecimento prévio de programação. Foi usado um teste de diagnóstico para avaliar alguns aspetos relacionados com a capacidade matemática, de cálculo e de raciocínio lógico dos estudantes no início do curso. A ideia principal era a de tentar correlacionar cada um desses parâmetros com os resultados obtidos pelos estudantes na disciplina inicial de programação. Concluiu-se que, em ambos os grupos, a experiência anterior de programação foi o fator mais relevante nos resultados obtidos, o que sugere que aprender a programar exige tempo e maturidade. Considerando apenas os estudantes que indicaram não ter qualquer experiência anterior de programação, encontrou-se uma correlação positiva entre os seus resultados e a sua capacidade de cálculo (em ambas as amostras), considerando-se assim este fator relevante para a aquisição de capacidades de programação. Encontrou-se também uma correlação positiva entre a nota geral de acesso dos estudantes e os seus resultados a programação. Concluiu-se que os estudantes que entraram no curso com notas mais elevadas obtiveram também melhores classificações em programação. No Grupo Python também se encontrou uma correlação entre os resultados na disciplina introdutória de programação e a nota de Matemática para acesso ao ensino superior. Estes resultados podem ser observados na Tabela I.

TABELA I: Resultados principais do estudo b

	Grupo C	Grupo Python
Experiência de programação	r=1	r=1
Nota Geral de acesso ao ensino superior	r=0,282*	r=0,451**
Nota a Matemática para acesso ao ensino superior	—	r=0,373**
Capacidade de cálculo	r=0,492**	r=0,416*

*(*ao nível 0,05 (2-tailed), **ao nível 0,01 (2-tailed))*

Assim, foram obtidos alguns resultados que mostraram que a maioria dos estudantes com dificuldades de aprendizagem de programação apresentava também enormes dificuldades em resolver problemas, especialmente os de base matemática.

2.2 Estilos de Aprendizagem

Numa fase inicial também se refletiu sobre a razão pela qual muitos estudantes pareciam incapazes e desinteressados de resolver certos tipos de problemas, especialmente os relacionados com programação ou de base matemática. Porém, alguns deles consideravam outros tipos de problemas desafiantes e motivadores, por exemplo os que envolviam desafios lógicos, charadas, a descoberta de perguntas falaciosas ou raciocínios errados, em particular quando apresentados de determinadas formas. Isto levou ao interesse em procurar averiguar se existia algum padrão de perfil de aprendizagem nos estudantes com dificuldades de aprendizagem de programação. Assim, aproveitou-se a realização dos estudos (estudo A e estudo B), descritos na secção anterior, para obter informação sobre os perfis de aprendizagem dos participantes. Usou-se o modelo de Felder-Silverman (Felder & Silverman, 1988), uma vez que foi criado tendo em vista a sua utilização com estudantes de engenharia. Foi usado o Índice de Estilos de Aprendizagem (Felder & Soloman, 1988), um instrumento composto por 44 frases que permite classificar cada estudante em cinco dimensões, Sensorial/Intuitivo, Visual/Verbal, Ativo/Reflexivo, Sequencial/Global e Indutivo/Dedutivo. Verificou-se que a maioria dos estudantes que participou no estudo A apresentava uma preferência Visual (89,7%), Sensorial (72,4%), Ativa (51,72%) e Sequencial (68,9%). Neste estudo analisou-se a forma como os estudantes com diferentes estilos de aprendizagem abordavam a resolução de problemas, bem como o tipo de representações que preferiam usar para expressar as suas soluções. Apesar de terem emergido nas experiências alguns aspetos não referidos no modelo teórico, na generalidade concluiu-se que as estratégias utilizadas pelos estudantes correspondiam às características previstas no modelo teórico para o seu estilo de aprendizagem.

Aproveitou-se também a realização do estudo B para verificar o perfil dos estudantes com dificuldades na aprendizagem de programação e também o dos estudantes bem sucedidos neste domínio (Gomes et al., 2006). Foram obtidos resultados semelhantes em ambas as amostras. No Grupo_C a maioria dos estudantes era Visual (96,55%), Sensorial (79,27%), Ativo (67,07%) e Sequencial (78,83%). No Grupo_Python a maioria dos estudantes também era Visual (82,35%), Sensorial (73,33%), Ativo (67,78%) e Sequencial (65,91%). Embora os valores referidos fossem relativos à totalidade das amostras, incluindo estudantes com e sem dificuldades de aprendizagem de programação, análises separadas levaram à obtenção de perfis muito semelhantes. Procurou-se também encontrar alguma correlação entre os estilos de aprendizagem dos estudantes e os resultados por eles obtidos na primeira disciplina de programação. Para isso tentou-se estabelecer uma correlação entre as classificações e cada dimensão do estilo de aprendizagem. No Grupo_C apenas se encontrou uma correlação negativa (r=4,07 ao nível 0,05, *2-tailed*) ao cruzar a dimensão Ativa com as classificações em programação. No Grupo_Python não se encontrou qualquer correlação. Estes estudos permitiram estabelecer um padrão de estilos de aprendizagem dos estudantes participantes. No entanto, não confirmaram uma relação entre estilo de aprendizagem e desempenho na aprendizagem inicial de programação.

Os estilos de aprendizagem dos estudantes viriam novamente a ser estudados durante o estudo D (Gomes & Mendes, 2009), que será descrito mais em detalhe na secção seguinte. Os resultados foram semelhantes aos anteriores, obtendo-se novamente como perfil mais frequente entre os estudantes o Sensorial-Visual-Ativo-Sequencial. Também neste estudo se correlacionaram os resultados obtidos pelos estudantes com cada dimensão do estilo de aprendizagem, mas não se encontrou qualquer correlação.

2.3 Aspetos Motivacionais

Os aspetos motivacionais foram objeto de atenção desde o início dos nossos estudos. Em particular, questionou-se a motivação e persistência dos estudantes, especialmente aqueles que mostravam dificuldades de

aprendizagem de programação. Para este fim aproveitou-se o inquérito feito no estudo B, no qual se incluíram algumas perguntas com a intenção de reunir indicadores sobre a motivação dos estudantes para a área de estudo. No geral, os níveis de motivação destes estudantes revelaram-se satisfatórios. Procurou-se uma correlação entre os níveis de motivação dos estudantes e a sua classificação na disciplina introdutória de programação, mas não foi possível provar uma relação entre as duas. Uma análise detalhada mostrou situações, em ambos os grupos, de estudantes que não pareciam motivados, mas que conseguiram boas classificações e também o contrário.

Mais tarde, no contexto de outros estudos, Estudo E (Gomes, Santos & Mendes, 2012) e Estudo F (Gomes, 2010), (descritos mais à frente), utilizou-se o IACHE (acrónimo de Inventário de Atitudes e Comportamentos Habituais de Estudo) (Tavares, Almeida, Vasconcelos & Bessa, 2004), um instrumento que permite, entre outros aspetos, avaliar a motivação dos estudantes. Foram utilizadas análises quantitativas descritivas e de correlações, confrontando os dados obtidos com os resultados dos estudantes nas disciplinas introdutórias de programação. Foi também verificada a evolução da motivação dos estudantes ao longo do ano letivo, tendo o mesmo instrumento sido aplicado no início e numa fase adiantada do 2º semestre.

Os resultados obtidos com o IACHE mostraram que os níveis motivacionais dos estudantes de ambas as amostras envolvidas podiam ser considerados médio-elevados. O mesmo se verificou na segunda aplicação do IACHE, ainda que se tenha notado uma ligeira descida dos níveis de motivação quando comparados com os obtidos inicialmente.

No caso da amostra LEI-UC, a correlação dos níveis de motivação dos estudantes com os seus resultados nas duas primeiras disciplinas de programação (Introdução à Programação e Resolução de Problemas - IPRP e Princípios de Programação Procedimental - PPP) revelou resultados positivos em ambos os casos (r=0,328 ao nível 0,05 2-tailed, em IPRP e r=0,216 ao nível 0,01 2-tailed, em PPP). Não foi possível fazer o mesmo estudo com a outra amostra (LEIS-ISEC), dado que o número de estudantes que responderam ao inquérito e concluíram as disciplinas foi insuficiente.

Estes resultados reforçaram a necessidade de incluir preocupações com os aspetos motivacionais dos estudantes nas estratégias pedagógicas a utilizar nestas disciplinas.

2.4 Nível Cognitivo

Os primeiros estudos realizados reforçaram a ideia de que muitos estudantes apresentavam grandes dificuldades para resolver problemas, e muitos deles não conseguiam mesmo começar a resolver problemas simples. Com o intuito de conhecer melhor essas dificuldades foi realizado o estudo C (Gomes et al., 2008). Este decorreu no 2 º semestre de 2007/2008, envolvendo seis estudantes que frequentavam a disciplina introdutória de programação da Licenciatura em Matemática. Estes estudantes manifestavam dificuldades claras em aprender a programar. Considerou-se que este grupo de estudantes não deveria ter dificuldades em resolver problemas genéricos ou aqueles que incluíssem conceitos matemáticos implícitos ou explícitos. Mesmo havendo essa ideia inicial, os estudos foram desenhados para facilitar a identificação do tipo de dificuldades (matemáticas e/ou de programação) de cada estudante e propor atividades corretivas em consonância. A reduzida dimensão da amostra permitiu um acompanhamento direto de cada aluno e a observação das suas estratégias de resolução de problemas e das suas dificuldades. O estudo incluiu duas sessões em que os estudantes foram confrontados com diferentes exercícios, teóricos e práticos. Os exercícios eram propostos em função das dificuldades demonstradas, quer do ponto de vista matemático quer de programação. Para ajudar a determinar o nível de dificuldades dos estudantes a sequência de problemas apresentados a cada um seguiu a

taxonomia dos objetivos educacionais de Bloom (Bloom et al., 1956, Anderson et al., 2001) para que se pudesse verificar em que nível (em termos de matemática e/ou programação) cada estudante apresentava dificuldades. Apesar da Taxonomia de Bloom focar os domínios afetivo, psicomotor e cognitivo, as experiências realizadas incidiram apenas no último. As competências do domínio cognitivo incluem seis níveis: Conhecimento, Compreensão, Aplicação, Análise, Síntese e Avaliação.

Contrariamente ao que se esperava, os estudantes mostraram falta de conhecimentos sobre alguns conceitos básicos de matemática, demonstrando apenas uma compreensão superficial dos mesmos. Os estudantes, que se sabia terem dificuldades de aprendizagem de programação, também apresentavam enormes dificuldades em resolver problemas. Aconteceu com frequência que as dificuldades apresentadas pelos estudantes puderam ser superadas quando ajudados a desenvolver as capacidades matemáticas inerentes. Verificou-se também frequentemente que os estudantes, quando confrontados com a explicação dos conceitos envolvidos, conseguiam entender e fazer algumas aplicações que os utilizavam. No entanto, raramente conseguiam fazer generalizações ou abstrações, nunca atingindo os níveis mais elevados da hierarquia de Bloom, quer nos problemas de matemática quer nos de programação. Também se pode constatar que os estudantes não faziam muitos esforços para encontrar formas de resolver os problemas propostos, parecendo não ter a determinação necessária para avançar. Nas suas tentativas, os estudantes não cometeram muitos erros sintáticos de programação, sendo a maioria deles erros lógicos. Nas poucas codificações feitas revelaram conhecer a sintaxe de algumas instruções isoladas, mostrando mais dificuldades na compreensão de ciclos e variáveis. Apenas alguns estudantes compreenderam conjuntos relacionados de instruções. As dificuldades aumentaram quando os estudantes foram convidados a explicar a funcionalidade total de um programa. A partir do momento em que lhes foi solicitado que realizassem alterações do código, mesmo num contexto bem delimitado, os estudantes começaram a desistir ou a dar respostas erradas. As dificuldades em juntar partes de código ou produzir codificações foram as mais frequentes. Estas atividades requerem aptidões cognitivas mais exigentes para as quais os estudantes não estavam preparados. A maioria deles conseguiu traduzir partes isoladas do código, mas não conseguiu interpretá-las ou generalizá-las. Quando o código era constituído por várias instruções o seu desempenho diminuiu e agravou-se ainda mais na análise e desenvolvimento de programas completos. Os estudantes não conseguiam criar qualquer programa, mesmo que fosse semelhante a um previamente fornecido. Por vezes, escreviam códigos parcialmente corretos, mas a interação entre instruções estava errada, produzindo uma solução globalmente errada. Outras vezes os estudantes esqueciam-se de considerar casos especiais nos dados de entrada, fazendo com que os programas falhassem nessas situações. Ocasionalmente, notou-se que o código dos estudantes estava correto, mas no lugar errado do programa. Gomes et al. (2008) descrevem detalhadamente os tipos de erros e dificuldades que os estudantes apresentaram nas diversas tentativas para interpretar ou codificar um programa. Este tipo de análise só foi possível através do uso de uma taxonomia que mostrou claramente os níveis de dificuldades dos estudantes. Constatou-se que as questões de programação tipicamente colocadas pelos professores em aulas e exames encontram-se num nível de exigência igual ou superior ao das questões de nível mais elevado usadas neste estudo. Uma vez que as perguntas respondidas corretamente pelos estudantes corresponderam aos níveis mais baixos da taxonomia de Bloom, considera-se haver uma discrepância entre as atividades de programação geralmente propostas aos estudantes nas aulas e os níveis cognitivos de muitos deles.

A validação desta hipótese foi novamente considerada. Para tal, realizou-se o estudo D (Gomes & Mendes, 2008) no 4º trimestre do ano letivo 2007/2008, envolvendo 145 estudantes da LEIS-ISEC. A metodologia utilizada foi o estudo de registos, sendo baseado nos exames

finais da disciplina de Tecnologia da Informática (TI). Este exame foi estruturado com base na Taxonomia de Bloom, ligeiramente modificada. Isto é, em vez de pedir aos estudantes para, essencialmente, fazerem programas completos (como era habitual), estruturou-se o exame em três grupos de perguntas, incluindo em cada um questões de dificuldade crescente. O primeiro grupo de perguntas foi estruturado em diferentes níveis. O 1.º nível incluiu perguntas que testaram o nível de Conhecimento. O 2.º nível incluiu perguntas para testar o nível de Compreensão básica. Os 3.º e 4.º níveis tinham perguntas para testar os níveis de Compreensão avançada e Aplicação. No segundo grupo de perguntas os estudantes foram convidados a escrever um pequeno fragmento de código para completar um programa. No último grupo, os estudantes tinham que criar um programa completo de raiz. Para analisar o desempenho dos estudantes foi considerada satisfatória qualquer resposta classificada com mais de 50%. A grande maioria dos estudantes obteve resultados positivos nas perguntas do nível mais básico, nomeadamente as de nível de Conhecimento e Compreensão. No entanto cerca de 2/3 dos estudantes não obteve resultados satisfatórios nas perguntas de Compreensão avançada (interpretação de um programa completo muito simples) e Aplicação. Estes dados revelam a grande dificuldade dos estudantes em compreender programas completos, quando comparada com a compreensão do papel de instruções individuais dentro de um dado programa. Apenas 15,73% dos estudantes obteve resultados satisfatórios em questões que implicavam a escrita de pequenos fragmentos de código dentro de um contexto bem definido. Apenas 9,46% dos estudantes obteve resultados satisfatórios na questão que implicava a elaboração de um programa completo. Esses baixos valores demonstram a grande lacuna existente entre a compreensão de código já feito, a construção de pequenas partes de código e a construção integral de um programa. Este estudo forneceu alguns indicadores que confirmam a diferença entre as atividades de programação geralmente propostas aos estudantes e os seus níveis cognitivos. Desta forma, considera-se que a realização de atividades seguindo uma taxonomia de objetivos educacionais tem vantagens pedagógicas para os estudantes mais fracos. Para os professores, torna mais fácil a tarefa de identificar as dúvidas de cada estudante e assim ajudá-los a superar essas dificuldades.

2.5 Métodos de Ensino e de Estudo

À medida que fomos desenvolvendo estudos e conhecendo melhor os estudantes surgiu o interesse em verificar se os métodos de estudo utilizados seriam adequados para a aprendizagem da programação. Assim, surgiu o estudo E (Gomes, Santos & Mendes, 2012), realizado durante o ano letivo de 2008/2009, envolvendo estudantes da LEI-UC (nas primeira e segunda disciplinas de programação, IPRP e PPP, respetivamente) e da LEIS-ISEC (na primeira disciplina de programação, AP). Um dos objetivos era caracterizar os comportamentos de estudo das duas amostras referidas e relacioná-los com os resultados obtidos nas disciplinas de programação. O instrumento utilizado foi o IACHE, um questionário multidimensional sobre métodos de estudo composto por 44 itens. Inclui dimensões cognitivas, motivacionais e comportamentais, formadas por cinco subescalas: Enfoque Compreensivo (estudo mais profundo); Enfoque Reprodutivo (estudo mais superficial); Enfoque Perceções Pessoais; Enfoque Motivacional (envolvimento no estudo) e Enfoque Organizativo (organização das atividade de estudo). Em geral, e para ambas as amostras, os resultados mostraram que os estudantes usaram muito estudo superficial (Enfoque Reprodutivo médio-elevado). Os níveis referentes ao Enfoque Compreensivo foram considerados médios-elevados. Os níveis referentes à motivação estavam no limiar do moderado a alto, porém as perceções pessoais não se mostraram satisfatórias. Os níveis de organização revelaram-se médios. Cruzando cada subescala do instrumento com as classificações obtidas nas disciplinas de programação obtiveram-se correlações entre as perceções pessoais dos estudantes da amostra LEI-UC

e as classificações obtidas a IPRP e a PPP. Também se obtiveram correlações entre os aspetos motivacionais e as classificações nas duas disciplinas, como se mostra na Tabela II. No caso da amostra LEIS-ISEC não foi possível encontrar correlações devido ao número insuficiente de estudantes desta amostra que concluíram a disciplina.

Estes resultados mostraram que as perceções pessoais e a motivação são fatores importantes e estão relacionados com os resultados nas disciplinas de programação.

O instrumento utilizado também inclui um grupo de oito questões referentes às expectativas académicas e ao grau de satisfação dos estudantes com o curso e a instituição. Embora houvesse algumas oscilações entre amostras, pode-se considerar que os estudantes estavam, em geral, satisfeitos.

O IACHE possui ainda outro grupo de questões onde os estudantes devem indicar, nas suas perspetivas, quais as principais causas subjacentes às suas possíveis dificuldades de aprendizagem. As principais causas indicadas, pela maioria dos estudantes de ambas as amostras, foram "Falta de esforço/persistência pessoal" e "Falta de motivação". "Falta de bases de conhecimento" e "Métodos de estudo" também foram referidos por uma percentagem razoável de estudantes. Estes são alguns outros elementos que enfatizam a importância das questões motivacionais e de perceção pessoal de autoeficácia, persistência e autoestima.

TABELA II: Amostra LEI-UC – Correlação entre os Enfoques e os resultados nas duas disciplinas de programação

	IPRP	PPP
Enfoque Compreensivo	-	-
Enfoque Reprodutivo	-	-
Enfoque Perceções Pessoais	r=-0,507**	r=-0,460**
Enfoque Motivação	r=0,328*	r=0,216*
Enfoque Organização	-	-

*(*ao nível 0,05 (2-tailed), **ao nível 0,01 (2-tailed))*

Outro estudo (estudo F) foi feito durante o 2 º semestre do ano letivo de 2008/2009, envolvendo 130 estudantes matriculados na segunda disciplina de programação da LEI-UC. Este estudo teve como objetivo principal a análise do impacto de uma alteração à metodologia de ensino nos resultados finais na disciplina. Esta alteração consistiu na inclusão de atividades de revisão no início de cada aula prática, antes de passar para um novo tópico. Estas atividades eram facultativas e estavam organizadas com base na Taxonomia de Bloom. Foi utilizada uma metodologia experimental, de forma a procurar esclarecer a relação causal entre determinado método de ensino e os resultados na disciplina. Para tal constituíram-se dois grupos – um grupo experimental que foi submetido à realização de atividades no início de cada aula prática e um grupo de controlo onde essas atividades não foram realizadas. Os resultados não mostraram diferenças estatisticamente significativas entre as classificações obtidas pelos estudantes do grupo experimental e as do grupo de controlo. No entanto, verificou-se uma correlação positiva (r=0,320 ao nível 0,01 2-tailed e r=0,313 ao nível 0,01 2-tailed) entre as perceções pessoais dos estudantes e a quantidade e qualidade das atividades realizadas, respetivamente. Desta forma pode-se afirmar que os estudantes com melhores perceções pessoais se empenharam mais nas atividades propostas.

3. PREOCUPAÇÕES PEDAGÓGICAS E SUA APLICAÇÃO

Os estudos apresentados nas secções anteriores, os trabalhos relatados em Martins, Mendes & Figueiredo (2010) e a experiência pedagógica acumulada ao longo dos anos permitiram identificar um conjunto de preocupações da maior importância para que os docentes consigam melhorar as condições de aprendizagem dos seus estudantes, em particular:

- Dar uma grande importância à motivação dos estudantes, não só despertando o seu interesse para a programação como também procurando manter essa mesma motivação ao longo do tempo;

- Introduzir os conceitos através de exemplos familiares aos estudantes, procurando não incluir conceitos de outras áreas (por exemplo a Matemática) que, se não forem já dominados pelo estudante, adicionam dificuldades à tarefa em causa;
- Expor o estudante a problemas exigentes, mas com um nível de dificuldade que não seja muito superior às suas capacidades nesse momento;
- Utilizar ferramentas e estratégias que permitam aos estudantes entender a dinâmica inerente a qualquer programa;
- Considerar a diversidade de estilos e preferências de aprendizagem que podem coexistir na turma.

Estas preocupações pedagógicas induziram um conjunto de modificações organizacionais e pedagógicas efetuadas na disciplina de Introdução à Programação e Resolução de Problemas (IPRP) da Universidade de Coimbra, a partir do ano letivos de 2011/2012.

Tendo em conta os maus resultados que se vinham obtendo na disciplina introdutória de programação, que fazia parte dos curricula das licenciaturas em Engenharia Informática, Engenharia e Gestão Industrial e Design e Multimédia, as autoridades académicas decidiram passar a oferecer disciplinas separadas para cada um dos cursos e alterar a sua estrutura letiva. Assim, anteriormente havia aulas teóricas, teórico-práticas e laboratoriais, num total de 5 horas semanais, funcionando muitas vezes com professores diferentes. Estas foram substituídas por um único tipo aulas, com os alunos divididos em turmas de cerca de 24 alunos. Cada turma passou a ter uma aula com 2 horas e outra com 3 horas em cada semana, ambas da responsabilidade do mesmo professor. O grande objetivo destas alterações foi conseguir um contexto de aprendizagem mais adequado às características dos estudantes de cada curso. Com um tempo de aula mais alargado e menos estudantes em cada turma foi possível ao professor propor atividades adequadas a cada estudante individualmente e fazer a supervisão necessária às suas necessidades cognitivas. Esta estrutura permitiu também uma maior proximidade entre cada estudante e o seu professor.

Estas alterações organizacionais foram aproveitadas para alterar significativamente a estratégia pedagógica utilizada na disciplina introdutória de programação destinada aos estudantes da Licenciatura em Design e Multimédia - LDM.

A primeira alteração foi a linguagem de programação utilizada, que passou a ser Processing. O motivo não foi a simplicidade da linguagem (pelo contrário), mas a sua capacidade de facilmente suportar o desenvolvimento de aplicações que exibissem desenhos e animações, permitindo introduzir os conceitos de programação num contexto familiar aos estudantes, uma vez que as instruções utilizadas têm uma concretização visual. Por outro lado, é uma linguagem utilizada profissionalmente por designers e artistas, o que poderia contribuir para aumentar o interesse e motivação dos estudantes. Outra alteração consistiu na abordagem utilizada para iniciar a disciplina, tendo a primeira atividade (na 1ª aula) como objetivo despertar esse interesse. Em vez de se dizer aos estudantes qual a importância de aprenderem a programar, foi pedido a cada um que fizesse uma pesquisa na web sobre projetos feitos usando Processing, selecionasse um deles e o apresentasse na aula seguinte. Muitos estudantes manifestaram-se surpreendidos com a diversidade e qualidade de projetos existente, mostrando interesse em vir a ser capazes de fazer aplicações semelhantes.

A estratégia de introdução dos vários conceitos de programação passou sempre pela sua contextualização em pequenos exercícios, em vez de se usar uma simples explanação teórica. Ou seja, primeiro era criada a necessidade e depois introduzido o conceito ou instrução que poderia resolver essa necessidade. Logo após esta introdução eram propostos exercícios de reforço sobre o mesmo conceito. Desta forma foram

eliminados alguns problemas que eram frequentes anteriormente devido à separação entre aulas teóricas e práticas (alunos que não tinham entendido as explicações nas aulas teóricas, alunos que já tinham esquecido parte do que tinha sido explicado ou, mesmo, alunos que faltavam às aulas teóricas e chegavam às aulas práticas sem qualquer ideia sobre os conceitos relevantes para essa aula). Esta estratégia permitiu uma maior concretização dos diversos conceitos, bem como a utilização de abordagens mais interativas e participativas dado que o número de estudantes em cada aula era relativamente reduzido.

Adicionalmente, os estudantes tiveram sempre à sua disposição listas de exercícios para cada tema, organizados em três níveis de dificuldade, de modo a que pudessem escolher os mais adequados à sua situação em cada momento e pudessem trabalhar autonomamente fora do período das aulas. Os próprios exercícios propostos pelo docente a cada estudante nas aulas procuravam levar em conta as suas capacidades nesse momento. Por vezes houve necessidade de aconselhar alguns estudantes a rever conceitos, como por exemplo trigonometria básica, de modo a conseguirem resolver alguns problemas. Esta dinâmica gerou frequentemente situações em que numa mesma aula havia estudantes a tentar resolver problemas diferentes e de diversos níveis de dificuldade. Claro que esta estratégia e o acompanhamento que ela implica só foram possíveis considerando a dimensão das turmas, a qual permitiu que o professor tivesse um bom conhecimento do nível e evolução de cada estudante.

Tal como referido anteriormente, a manutenção de níveis de motivação dos estudantes foi uma das preocupações fundamentais dos docentes. Para além da utilização de uma linguagem de programação adequada e exercícios adaptados a cada estudante, o que por si só poderia contribuir para a motivação dos estudantes, foram tomadas outras medidas com os mesmos objetivos:

- Foram desenvolvidos pequenos projetos (animações em geral) em que parte da especificação foi deixada ao gosto de cada estudante. O enunciado tinha uma parte obrigatória, mas deixava a cada estudante a definição de outros elementos ou comportamentos da animação;
- Houve a preocupação de apresentar o erro como uma oportunidade de aprender e não como algo a esconder. Os estudantes foram estimulados a procurar as causas dos seus erros e valorizados quando o conseguiam fazer. Como por vezes os erros resultam em efeitos visuais interessantes, essas situações foram utilizadas para discutir os erros e como eles levavam aos efeitos visualizados;
- O esforço e a evolução de cada estudante foi frequentemente objeto de pequenas conversas entre o docente e o estudante, mostrando-lhe que o seu trabalho estava a ser apreciado. Outras vezes o professor pedia a determinados estudantes para resolver um exercício em que anteriormente tinham falhado, realçando assim a sua evolução.

A estratégia geral das aulas foi essencialmente baseada no trabalho autónomo dos estudantes, individualmente ou em grupo, acompanhados de muito perto pelo professor. No entanto, houve também situações em que os docentes usaram estratégias de demonstração, criando programas para resolver pequenos problemas, raciocinando em voz alta e discutindo com os estudantes as opções tomadas. Este tipo de trabalho serviu para promover bons métodos de programação e combater a tendência de muitos alunos em começar a escrever código sem um prévio planeamento e estruturação da solução.

Os docentes tiveram também a preocupação de diversificar a forma como apresentaram conceitos ou como discutiram estratégias e resultados com os estudantes. Como exemplos é possível referir a utilização de pseudo-código e de fluxogramas para descrever o funcionamento das instruções de controlo de fluxo ou a inclusão de trabalhos individuais e de

grupo. Dessa forma procurou-se contemplar os diversos estilos de aprendizagem existentes em cada grupo de estudantes.

Uma boa relação entre o professor e cada um dos estudantes é importante para criar um ambiente propício à aprendizagem. De igual modo, uma comunicação eficaz pode ajudar, nomeadamente na identificação rápida de dificuldades e situações de desistência próxima. Por outro lado, é importante que os estudantes sejam capazes de refletir sobre as suas aprendizagens e dificuldades. Com este duplo objetivo foi utilizada uma atividade menos comum em cursos de programação: os estudantes tinham que escrever sobre a sua experiência de aprendizagem a cada duas semanas, utilizando para isso a funcionalidade Diário do Moodle. As reflexões eram privadas, podendo ser lidas apenas pelo professor e pelo seu autor. Esta atividade permitiu aos professores conhecer melhor cada estudante, já que alguns deles pareciam revelar mais facilidade em escrever sobre os seus problemas do que em falar sobre eles. Foi possível identificar e resolver alguns problemas de aprendizagem que estavam a causar dificuldades, e também impedir algumas desistências através de intervenções diretas com os estudantes envolvidos.

Pode-se concluir que esta estratégia só pode ser posta em prática através de um acompanhamento mais próximo do estudante, facilitado pelas mudanças estruturais introduzidas na disciplina, as quais apesar de serem essencialmente organizacionais tiveram grandes impactos pedagógicos. Pode-se igualmente afirmar que as mudanças só por si não surtirão efeito se não forem implementadas por professores com fortes preocupações pedagógicas e interessados pelos seus alunos.

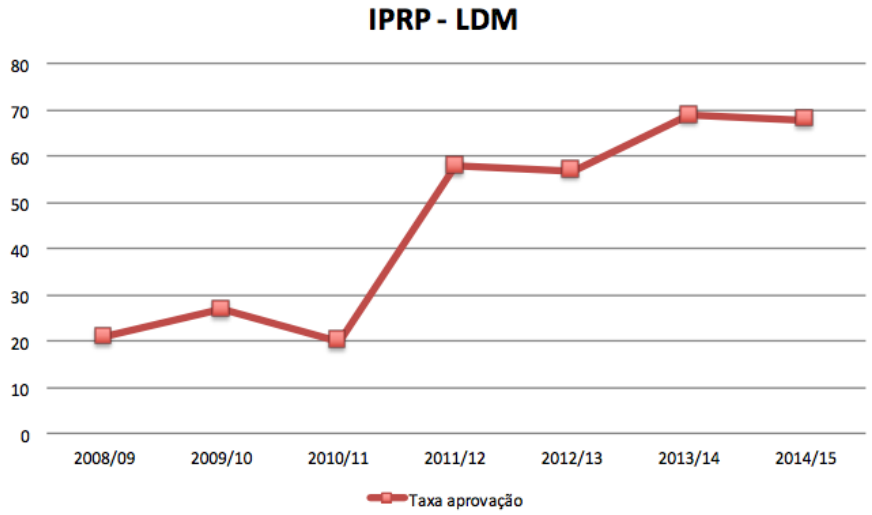
Numa primeira fase a adoção desta estratégia pedagógica implicou um aumento dos recursos docentes envolvidos. No entanto, num prazo curto o número de estudantes da disciplina começou a diminuir como consequência da melhoria das taxas de aprovação. Esta diminuição levou já o número de docentes necessários para valores semelhantes aos que eram precisos antes da alteração da estratégia pedagógica.

4. RESULTADOS

Os resultados obtidos com a utilização da estratégia descrita na secção anterior e detalhada em Gomes et al. (2012) foram aferidos usando três instrumentos: as classificações finais dos estudantes, o inquérito pedagógico oficial da Universidade de Coimbra e um questionário dirigido aos estudantes repetentes comparando esta nova estratégia com a utilizada anteriormente (apenas em 2011/12). A evolução verificada nas taxas de aprovação (número de estudantes aprovados/número total de estudantes matriculados) pode ser observada na figura 1. A evolução conseguida foi bastante positiva, pois as taxas de aprovação conseguidas com a nova estratégia (57%, 56%, 69% e 68% nos quatro últimos anos) foram bastante superiores às dos três anos anteriores (20%, 27% e 20% respetivamente). Ainda que não consideremos os resultados obtidos como excelentes, a evolução verificada foi significativa. É de notar que, tal como em anos anteriores, uma percentagem significativa de alunos não aprovados matriculou-se, mas não frequentou a disciplina.

O questionário respondido pelos estudantes repetentes em 2011/12 teve resultados igualmente positivos. Estes estudantes tinham frequentado a disciplina no modelo anterior, estando bem posicionados para avaliar a nova estratégia e modelo de organização da disciplina. No geral, foi manifestada uma visão positiva das alterações, destacando a melhor ligação entre a teoria e a prática, a maior proximidade entre aluno e professor, a maior disponibilidade de tempo para esclarecimento de dúvidas e para a prática de programação e a maior motivação para programar. Ligado com este último aspeto, os alunos referiram frequentemente a utilização da linguagem Processing por permitir a realização de atividades mais interessantes.

FIGURA I: Evolução da taxa de aprovação



Alguns estudantes consideraram, porém, o novo modelo mais cansativo devido à maior carga de trabalho envolvida. Outros (poucos) referiram que a existência de professores diferentes entre as turmas poderia levar a critérios de avaliação algo diferentes, podendo gerar alguma injustiça na avaliação. Adicionalmente, poderia acontecer que os professores enfatizassem diferentes aspetos nas várias turmas criando situações de desigualdade se esses assuntos surgissem no exame final. De referir que nos últimos anos a disciplina, apesar de ter incluído novos professores, tem conseguido resultados sempre positivos, tal como os inquéritos têm mantido a mesma tónica, resultantes da continuação da aplicação da mesma abordagem.

6. CONCLUSÃO

Aprender a programar é uma tarefa difícil para muitos estudantes, requerendo múltiplas competências implícitas e explícitas. As causas das dificuldades têm vindo a ser estudadas pelo grupo de investigação em que os autores se enquadram. Os resultados dos estudos realizados, os trabalhos relatados na literatura e a experiência pedagógica acumulada ao longo dos anos permitiram identificar um conjunto de preocupações da maior importância para que os docentes consigam melhorar as condições de aprendizagem dos seus estudantes. Essas preocupações prendem-se essencialmente com questões motivacionais, considerações referentes à diversidade de estilos de aprendizagem dos estudantes e a natureza e dificuldade dos exercícios que lhes são propostos. Pensamos poder afirmar que, o trabalho realizado até agora permitiu contribuir para o avanço do estado da arte relativamente ao ensino/aprendizagem de disciplinas introdutórias de programação. Nesse sentido apresentou-se uma estratégia pedagógica que através de mudanças organizacionais e pedagógicas tornou possível um melhor conhecimento e acompanhamento do estudante para combater as dificuldades identificadas. Esta experiência traduziu-se em resultados muito positivos medidos pelas classificações finais dos estudantes, pela sua satisfação e participação e pelas aprendizagens conseguidas.

7. REFERÊNCIAS BIBLIOGRÁFICAS

Anderson, L., Krathwohl, D., Airasian, P., Cruikshank, K., Mayer, R., Pintrich, P., et al. (2001). A Taxonomy for Learning and Teaching and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives. Addison Wesley Longman, Inc.. New York, USA.

Bennedsen, J. & Caspersen, M. E. (2006). Abstraction ability as an indicator of success for learning object-oriented programming? *SIGCSE Bulletin inroads*, *38* (2), 39-43.

Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). Taxonomy of Educational Objectives, Handbook I: Cognitive Domain. Longmans, Green and Co Ltd. London, UK.

Byrne, P. & Lyons, G. (2001). The effect of student attributes on success in programming. *SIGCSE Bulletin*, *33* (3), 49-52.

Fagin, B., Harper, J., Baird, L., Hadfield, S. & Sward, R. (2006). Critical thinking and computer science: implicit and explicit connections. *Journal of Computing Sciences in Colleges*, *21* (4), 171-177.

Felder, R. & Silverman, L. (1988). Learning and Teaching Styles in Engineering Education. *Journal of Engineering Education*, *78* (7), 674-681.

Felder, R. M., & Soloman, B. A. (1988). Learning styles and strategies. Obtido em 25 de Setembro de 2014, de Richard Felder's Education-Related Publications: http://www4.ncsu.edu/unity/lockers/users/f/felder/public/ILSdir/styles.htm

García, F. J. & Moreno, M. N. (2004). Software modeling techniques for a first course in software engineering: a workshop-based approach. *IEEE Transactions on Education*, *47* (2), 180-187.

Gomes, A. (2010). Dificuldades de aprendizagem de programação de computadores: contributos para a sua compreensão e resolução. Tese de Doutoramento, Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Gomes, A. & Mendes, A. J. (2008). A study on student's characteristics and programming learning. In *Proceedings of the World Conference on Educational Multimedia, Hypermedia & Telecomunications (ED-MEDIA'08)* (pp. 2895-2904). Vienna, Austria.

Pacheco, A., Gomes, A., Henriques, J., Almeida, A. & Mendes, A. J. (2008). A study on basic mathematics knowledge for the enhancement of programming learning skills. In *Proceedings of Informatics Education Europe III*. Veneza, Itália.

Gomes, A. & Mendes, A. J. (2009). Bloom's taxonomy based approach to learn basic programming. In *Proceedings of the World Conference on Educational Multimedia Hypermedia and Telecommunications (EDMEDIA'09)* (pp. 2547-2554). Honolulu, Hawaii, USA.

Gomes, A., Carmo, A., Bigotte, E. & Mendes, A. J. (2006). Mathematics and programming problem solving. In *Proceedings of the 3rd E-Learning Conference – Computer Science Education*. Coimbra, Portugal.

Gomes, A., Paquete, L., Cardoso, A. & Mendes, A. J. (2012). Increasing student commitment in introductory programming learning. In *Proceedings of the 42th ASEE/IEEE Frontiers in Education Conference (FIE'12)*. Seattle, USA.

Gomes, A., Santos, A. & Mendes, A. J. (2012). A study on students' behaviors and attitudes towards learning to program. In *Proceedings of the 17th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE'12)*. Haifa, Israel.

Hwang, Gwo-Jen, Wu, Po-Han, Chen, Chi-Chang. (2012). An online game approach for improving students' learning performance in web-based problem-solving activities. *Computers & Education*, *59* (4), 1246-1256.

Hwang, Wu-Yuin, Shadiev, Rustam, Wang, Chin-Yu & Huang, Zhi-Hua. (2012). A pilot study of cooperative programming learning behavior and its relationship with students' learning performance. *Computers & Education*, *58* (4), 1267-1281.

J. C. Perrenet, J. C., Bouhuijs, P. A. & Smits, J. G. (2000). The Suitability of Problem-based Learning for Engineering Education: Theory and practice. *Teaching in Higher Education*, *5* (3), 345-358.

Lahtinen, E., Ala-Mutka, K. & Järvinen, H-M. (2005). A study of difficulties of novice programmers. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (SIGCSE'05)*. Monte de Caparica, Portugal.

Lister, R. (2000). On blooming first year programming, and its blooming assessment. In *Proceedings of the Australasian Conference on Computing Education. (ACCE'00)*. Melbourne, Australia.

Liu, Chen-Chung, Cheng, Yuan-Bang, Huang, Chia-Wen. (2011). The effect of simulation games on the learning of computational problem solving. *Computers & Education*, *57* (3), 1907-1918.

Martins, S., Mendes, A. J. & Figueiredo, A. D. (2010). A strategy to improve student's motivation levels in programming courses. In *Proceedings of the 40th ASEE/IEEE Frontiers in Education Conference (FIE'10)*. Washington, USA.

Mills, J. E. & Treagust, D. F. (2013). Engineering education – is problem based or project-based learning the answer? *Australasian Journal of Engineering Education*, 1 (1), 2-17.

Moura, I. C. & van Hattum-Janssen, N. (2011). Teaching a CS introductory course: An active approach. *Computers & Education*, *56* (2), 475-483.

Pacheco, A., Gomes, A., Henriques, J., Almeida, A. & Mendes, A. J. (2008). A study on basic mathematics knowledge for the enhancement of programming learning skills. In *Proceedings of the Informatics Education Europe III (IEEIII'08)*. Venice, Italy.

Tasneem, S. (2012). Critical thinking in an introductory programming course. *Journal of Computing Sciences in Colleges*, *27* (6), 81-83.

Tavares, J., Almeida, L., Vasconcelos, R. & Bessa, J. (2004). Inventário de Atitudes e Comportamentos Habituais de Estudo – IACHE. Universidade de Aveiro e Universidade do Minho, Instituto de Educação e Psicologia, Centro de Investigação em Educação, 2004.

van Merriënboer, J. G. (2013). Perspectives on problem solving and instruction. *Computers & Education*, 64, 153-160.

Verdú, E., Regueras, L. M., Verdú, M. J., Leal, J. P., de Castro, J. P. & Queirós, R. (2012). A distributed system for learning programming on-line. *Computers & Education*, *58* (1), 1-10..

Looking for a context to support early learning programming

Abstract:

High failure rates are a common problem in many programming courses. This has motivated many researchers to propose methodologies and tools to help students. However, the panorama has remained mostly unchanged. Many causes for the learning problems have already been identified in the literature. Our research group has been conducting a set of experiments to identify and understand students programming difficulties. This paper includes the results of a set of studies conducted in the last few years. Their results served as a basis to define a new pedagogical approach consisting in a set of educational practices, looking to create learning contexts that motivate students, increase their involvement with course activities, and maximize their learning possibilities. The application of this pedagogical approach in a real setting is described and the results obtained are discussed.

Keywords: Computer Science Education, Enginefering students, Engineering education, Learning Styles, Problem Solving, Motivation, Learning Taxonomies

Texto:

- Submetido: março de 2015.
- Aprovado: maio de 2015.

Para citar este artigo:

Gomes, A. J., & Mendes, A. J. (2015). À procura de um contexto para apoiar a aprendizagem inicial de programação. *Educação, Formação & Tecnologias*, 8 (1), 13-27 [Online], disponível a partir de http://eft.educom.pt.

Notas biográficas dos autores

[i] **Anabela de Jesus Gomes** é Professora Adjunta no Departamento de Engenharia Informática e Sistemas do Instituto Politécnico de Coimbra. A sua investigação centra-se no Ensino/Aprendizagem em Ciências da Computação, tendo publicado mais de 40 artigos em revistas e conferências internacionais.

[ii] **António José Mendes** é Professor Associado no Departamento de Engenharia Informática da Universidade de Coimbra. Os seus interesses de investigação centram-se na Aprendizagem em Ciências da Computação, tendo publicado mais de 100 artigos em revistas e conferências internacionais.