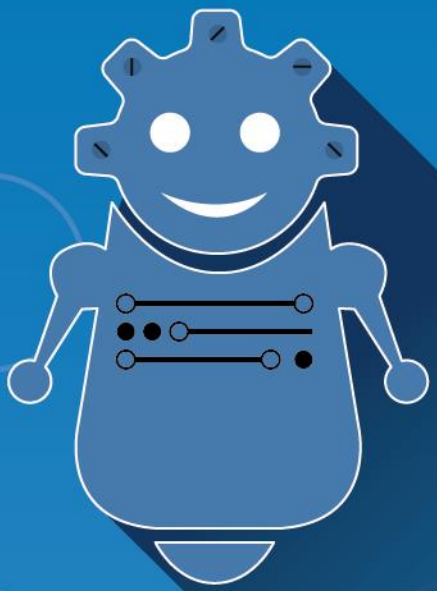


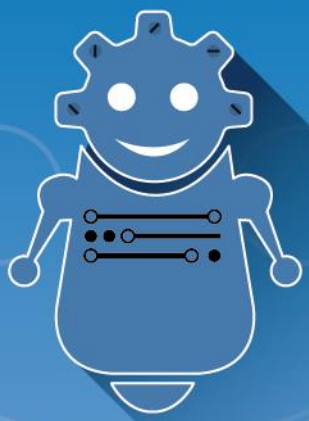
Probótica

Programação e Robótica no Ensino Básico

Linhas Orientadoras

2017

REPÚBLICA PORTUGUESA | EDUCAÇÃO
direção-geral da educação
PORTUGAL INCoDe.2030



Probótica

Programação e Robótica no Ensino Básico

Linhas Orientadoras

Autoria

Ana Pedro

João Filipe Matos

João Piedade

Nuno Dorotea

Instituto de Educação da Universidade de Lisboa

Design gráfico e ilustrações

Abel Silva

Instituto de Educação da Universidade de Lisboa

Parceiros

Microsoft

anpri
Associação Nacional de Professores de Informática

UNIVERSIDADE DE ÉVORA
CENTRO DE COMPETÊNCIA TIC

ccTIC
ESE/IPS - Setúbal

c2ti

ÍNDICE

INTRODUÇÃO

Iniciativa Programação e Robótica no Ensino Básico

Contextualização das Linhas Orientadoras

A iniciativa Programação e Robótica no Ensino Básico - Probótica - surge no ano letivo 2017/2018, decorrendo da implementação do projeto-piloto Iniciação à Programação no 1º ciclo do Ensino Básico entre 2015 e 2017. Tendo envolvido mais de setenta mil alunos durante os anos anteriores, inicia-se, neste momento, um novo ciclo de trabalho, onde se procura alargar as atividades desenvolvidas no âmbito da programação e robótica ao 2º e 3º ciclos do ensino básico.

Compreendendo os diferentes níveis de escolaridade e o consequente desenvolvimento cognitivo dos alunos, este documento apresenta-se organizado em quatro áreas das ciências da computação (Pensamento Computacional, Algoritmia, Programação e Robótica), estruturados de acordo com padrões de desenvolvimento – iniciais, intermédios e avançados – que pretendem estar articulados com os estádios de cada criança e jovem.

Considerou-se, assim, as preocupações trazidas por diferentes autores no que se refere ao ensino das ciências da programação a crianças e jovens (por exemplo, Armoni, 2012; Angeli, Voogt, Webb, Cox, Malyn-Smith & Zagami, 2016; Fluck, Webb, Cox, Angeli, Malyn-Smith, Voogt & Zagami, 2016), quando estes salientaram a necessidade de se considerar o desenvolvimento cognitivo das crianças quando se pensa nos conteúdos a abordar. Deste modo, enquadrou-se as perspetivas de desenvolvimento cognitivo aquando da identificação dos padrões de desempenho e respetivos descritores.

Os descritores identificados estão, assim, organizados em três padrões de desempenho para cada área de conhecimento, representando cada padrão de desempenho um referencial para as capacidades esperadas para cada estádio de desenvolvimento. Deste modo, não se tratando de áreas cumulativas, a necessária compreensão de alguns conceitos, bem com a capacidade de os implementar, apresenta-se numa sequencialidade essencial ao desenvolvimento de atividades nas diversas áreas.

Salienta-se igualmente que, de acordo, com o interesse e níveis de conhecimento dos alunos, bem como da disponibilidade de recursos, o professor poderá optar por desenvolver atividades de qualquer das áreas e de acordo com os padrões de desempenho que se pretenda atingir. Assim, as sugestões apresentadas deverão ser ajustadas à realidade de cada contexto em que a iniciativa será implementada, não existindo uma sequencialidade rígida nos conteúdos identificados.

Não se pretende, deste modo, definir uma sequencialidade obrigatória, mas sim considerar alguns conhecimentos de nível inicial como essenciais para a compreensão e aplicação de outros de nível mais avançado.

Procura-se, assim, identificar e contextualizar o que se pretende que os alunos desenvolvam, aprendendo programando, em projetos, ao criar histórias, animações e jogos e resolvendo desafios do quotidiano através da programação e da robótica, considerando diferentes cenários de aprendizagem suportados por metodologias ativas de ensino e de aprendizagem.

Apresenta-se assim possíveis orientações metodológicas e estratégias de operacionalização dos padrões de desempenho indicados, referindo igualmente alguns instrumentos e estratégias referentes à avaliação de atividades desenvolvidas no âmbito da iniciativa.

Competências para o séc. XXI e Iniciativa Programação e Robótica no Ensino Básico

Autores como Fluck et al. (2016) referem a necessidade de se considerar a existência do ensino das ciências da computação no ensino básico e secundário. No mesmo sentido, relatórios como o Horizon Report de 2017, indicam as ciências da computação como uma tendência a implementar no Ensino Básico e Secundário nos próximos dois anos.

Através desta iniciativa – Probótica – procura-se contribuir para o desenvolvimento de capacidades e competências-chave transversais ao currículo. Deste modo, recorrendo a metodologias ativas de aprendizagem, alicerçadas em cenários de aprendizagem, pretende-se estimular as aprendizagens, tornando-as simultaneamente mais significativas, possibilitando assim que os alunos desenvolvam competências multidisciplinares, nomeadamente as que se encontram referidas nos referenciais de competências do séc. XXI.

O conceito de competências do séc. XXI está associado à necessidade de corresponder às exigências da sociedade atual, e do futuro, onde a resolução de problemas, a tomada de decisões, o trabalho em equipa, o sentido ético, a gestão de projetos e a utilização de tecnologias digitais são consideradas competências essenciais.

Importa, deste modo, trazer como preocupação subjacente a todo o processo de construção das linhas orientadoras que aqui se apresentam, os domínios indicados pela literatura relativamente às competências necessárias. Procurou-se, ao longo da estruturação dos descritores identificados, organizados por sua vez em três padrões de desempenho, considerar o desenvolvimento de competências ligadas à comunicação, colaboração, criatividade e pensamento crítico (4cs – *Communication, Collaboration, Creativity, Critical Thinking*).

Deste modo, a construção dos padrões de desempenho, e a sua respetiva operacionalização, teve na sua génese a elaboração de estratégias e atividades que promovam competências ligadas aos domínios indicados, nomeadamente:

- Comunicação, através de estratégias que envolvam comunicação presencial e digital, escrita e falada;

- Colaboração, onde se focam capacidades desenvolvidas através da interação, discussão, diálogo e partilha;
- Criatividade, recorrendo a atividades colaborativas e individuais que promovam o desenvolvimento de pensamento crítico, fundamental para a reflexão e para a resolução de problemas;
- Pensamento crítico, ligado à capacidade de pensar e refletir sobre as diferentes situações, sendo essencial para a resolução de problemas.

LITERACIA DIGITAL

Mais do que a habilidade de utilizar computadores, a Literacia Digital abrange um conjunto de competências básicas que incluem o uso e produção de media digital, processamento e recuperação de informação, participação em redes sociais para criação e partilha de conhecimento, e uma ampla gama de habilidades informáticas profissionais (UNESCO[1]).

Ainda de acordo com a UNESCO[1], a Literacia Digital apresenta efeitos positivos sobre capacidades importantes para uma aprendizagem de sucesso. Os alunos podem aceder a informações com mais facilidade, pela crescente quantidade de dados disponível em repositórios digitais.

A utilização das tecnologias digitais implica um novo conjunto de competências e estratégias de pesquisa e avaliação da informação, bem como de comportamentos seguros na partilha e comunicação. É uma abordagem transversal a diversas áreas: segurança e privacidade da informação; partilha e comunicação; pesquisa e fontes fidedignas; direitos de autor.

A literacia digital[2] providencia uma compreensão crítica do impacto da tecnologia na sociedade e nos indivíduos, incluindo a privacidade, uso responsável, questões legais e éticas.

No âmbito da iniciativa de Programação e Robótica no Ensino Básico, define-se os seguintes objetivos para a promoção da Literacia Digital:

- usar a tecnologia de forma segura, respeitosa e responsável, mantendo as informações pessoais privadas;
- conhecer métodos de proteção da sua identidade e como manter a sua privacidade online;
- compreender de que modo as mudanças na tecnologia afetam a segurança;
- usar a tecnologia propositadamente para criar, organizar, armazenar, manipular e recuperar informação digital;
- avaliar a veracidade da informação pesquisada e a fidedignidade das suas fontes;
- compreender as oportunidades oferecidas pela internet para comunicar, colaborar e partilhar informação;
- demostrar comportamentos adequados na colaboração e comunicação online;
- respeitar os direitos de autor na utilização de materiais que não sejam de sua autoria;
- identificar de que modo as tecnologias digitais influenciam o quotidiano e o relacionamento com os outros.

[1] Digital literacy in education; Policy brief. Disponível em: http://unesdoc.unesco.org/images/0021/002144/214485e.pdf

[2] British Computer Society/Royal Academy of Engineering, (2012). Disponível em: www.bcs.org

ÁREAS DAS CIÊNCIAS DA COMPUTAÇÃO

As Ciências da Computação, como campo de estudo de computadores e processos algorítmicos, incluindo os seus princípios, o desenho do seu hardware e software, as suas aplicações e o seu impacto na sociedade (Tucker, 2006), abrange um vasto leque de áreas e fundamentos. É, segundo Berry (2013), onde os alunos aprendem como os sistemas digitais funcionam, como são projetados e programados, e quais os princípios fundamentais da informação e computação.

De acordo com o documento *ICT and Computer Science in UK schools*[3], as Ciências da Computação estudam os princípios e práticas fundamentais da computação e pensamento computacional bem como a sua aplicação no design e desenvolvimento de sistemas informáticos. É uma disciplina curricular, a par com a Matemática ou a Física. Já as TIC (Tecnologias de Informação e Comunicação) centram-se na utilização criativa e produtiva, bem como na aplicação, de tecnologia e sistemas informáticos, especialmente em organizações. Ainda segundo os mesmos autores, as duas áreas apresentam sobreposições, uma vez que as Ciências da Computação incluem aspetos do uso e aplicação de computadores e que as TIC abrangem aspetos de programação e compreensão de dispositivos computacionais. Assim, não sendo entendidas como o mesmo, complementam-se.

É comummente aceite que as TIC são transversais a qualquer currículo e devem considerar áreas como a literacia e segurança digital. Do mesmo modo, pelos fundamentos e áreas de pesquisa que abrangem, as Ciências da Computação podem ser aplicadas a qualquer área do conhecimento. Com a atual presença do digital na sociedade, importa tornar os cidadãos capazes de lidar e compreender o mundo digital, não se configurando apenas como consumidores passivos. A compreensão dos conceitos relacionados com a computação contribui, deste modo, para um melhor conhecimento do funcionamento da tecnologia, sistemas de informação e como detetar e resolver problemas.

[3] Naace, ITTE, and the Computing at School Working Group, June 2012. Disponível em: http://www.computingatschool.org.uk/data/uploads/ICT and CS joint statement.pdf

Miles Berry (2013), propõe a seguinte correspondência entre termos usados em TIC e os usados em Ciências da Computação[4]:

TIC	Ciências da Computação
Users	*Makers*
Consumers	*Creators*
Communicators	*Collaborators*
Digitally literate	*Digitally critical*
Safe	*Responsible*
Skills	*Understanding*

Relativamente à aprendizagem da programação, há evidências de que esta melhora a capacidade de resolução de problemas e superação de obstáculos, envolvendo diversas áreas disciplinares. Trata-se de aptidões fundamentais para lidar no quotidiano de uma sociedade fortemente digital.

Na perspetiva de Mitchel Resnick[5], "no processo de aprender a codificar, as pessoas aprendem muitas outras coisas. Não estão apenas a aprender a codificar, estão a codificar para aprender. Para além de aprender ideias matemáticas e computacionais (como variáveis e condicionais), também estão a aprender estratégias para resolver problemas, projetar projetos e comunicar ideias. Essas habilidades são úteis não apenas para cientistas da computação, mas para todos, independentemente da idade, antecedentes, interesses ou ocupação." Mais do que aprender a programar os alunos aprendem a "dividir problemas complexos em partes mais simples, com refazer projetos, como identificar e corrigir erros, como partilhar e colaborar com outros, como perseverar diante desafios."

Se a programação permite a materialização em aplicações (*software*) de algoritmos concebidos para a resolução de problemas ou situações, a robótica proporciona a execução tangível de soluções concretas para problemas em interação com o mundo físico.

A robótica abrange assim todas as áreas atrás referidas - o pensamento computacional, a algoritmia, a programação e ainda os robôs e outros objetos tangíveis programáveis. É entendida como um sistema complexo, composto por componentes mecânicos, eletrónicos e outras estruturas que lhes dão forma - podendo ser utilizados na sua construção os mais variados materiais, recicláveis, impressos em tecnologias 3D, entre outros. Controlados por microprocessadores programáveis utilizam atuadores (motores) e sensores para interagirem com o ambiente que os rodeia.

Quando enquadrada num ambiente tecnológico, a robótica estimula o desenvolvimento da criatividade e a construção do conhecimento pelo próprio aluno, contribuindo para a definição de estratégias de resolução de problemas e envolvendo-o simultaneamente em soluções complexas que podem requerer pensamento de alto nível. Deste modo, a

[4] Iniciação à Programação no 1º Ciclo do Ensino Básico - Linhas Orientadoras Gerais: http://www.erte.dge.mec.pt/sites/default/files/Projetos/Programacao/IP1CEB/linhas_orientadoras.pdf

[5] Learn to Code, Code to Learn: https://www.edsurge.com/news/2013-05-08-learn-to-code-code-to-learn

resolução de problemas complexos pode ser introduzida de forma parcelar e gradual, levando o aluno a novos desafios numa progressão contínua que envolve o desenho físico (estrutura, mecanismo e sensores), o desenho computacional (programação) e a análise e avaliação dos resultados.

Em atividades que configuram desafios contextualizados, a robótica apresenta-se como um extraordinário potencial pedagógico para a abordagem de temas e conceitos multidisciplinares de uma forma prática, tangível e motivadora.

No sentido de contribuir para i) o desenvolvimento de capacidades associadas à computação, ii) aumentar os níveis de literacia digital dos alunos e iii) fomentar competências transversais ao currículo, e considerando a operacionalização da iniciativa Probótica nos três ciclos do ensino básico, sugere-se as seguintes áreas das ciências da computação a abordar: Pensamento Computacional e Algoritmia, Programação e Robótica.

Apresenta-se de seguida as áreas das ciências da computação sugeridas na iniciativa Probótica, bem como os objetivos de aprendizagem e padrões de desempenho.

Pensamento Computacional e Algoritmia

Quando uma pessoa usa um computador com o objetivo de resolver um dado problema, o tipo de pensamento envolvido no mapeamento do problema com vista à sua implementação computacional é designado de Pensamento Computacional. A expressão *Computational Thinking* foi utilizada por Jeannette Wing para designar processos de resolução de problemas de forma computacional. Estes processos incluem características tais como i) a formulação de um problema de tal modo que permita e facilite a utilização de meios computacionais para implementar a sua resolução; ii) a organização lógica e a análise de dados; iii) a representação de variáveis, e das suas relações, através de formas abstratas tais como modelos matemáticos e simulações; iv) a criação de soluções automatizadas por meio de um processo algorítmico (entendido como uma série ordenada de passos); v) a identificação, análise e implementação de possíveis soluções para o problema com o objetivo de criar a forma mais eficiente de o resolver com ajuda de recursos computacionais; vi) a generalização do processo de resolução e a sua transferência para uma classe mais alargada de problemas também ela passível de ser implementada em meios computacionais.

Progredir no domínio do Pensamento Computacional implica (mas simultaneamente desenvolve) a persistência ao trabalhar em problemas difíceis, a tolerância pela ambiguidade, a confiança ao lidar com a complexidade, a capacidade de lidar com problemas abertos e a capacidade de lidar e trabalhar com outras pessoas para um objetivo comum. De acordo com Wing (2006) o desenvolvimento do Pensamento Computacional justifica-se porque:

- trata-se de conceptualizar problemas (e não de programar)
- desenvolve *skills* fundamentais para pensar e resolver problemas (e não para decorar factos)
- trata-se de analisar formas como os humanos (não os computadores) pensam
- tem uma natureza complementar e combinatória de pensamento matemático e de engenharia
- trata de ideias (e não de artefactos tecnológicos).

Do ponto de vista das aprendizagens dos alunos, o contributo do Pensamento Computacional e Algoritmia passa por elementos tais como i) tornarem-se capazes de identificar aspetos de um problema que sejam suscetíveis de ser implementados através de meios computacionais, ii) tornarem-se aptos a avaliar as limitações e o poder das ferramentas computacionais que utilizam, iii) serem capazes de reutilizar, adaptar e combinar técnicas e estratégias de resolução a novos problemas, iv) saber reconhecer oportunidades para usar ferramentas computacionais em novas formas e com novos objetivos.

Informalmente, pensamento computacional e algoritmia descrevem a atividade de formulação de um problema e a sua operacionalização algorítmica de forma que admita uma solução computacional. Essa solução pode ser encontrada através da utilização de uma máquina ou apenas através da intervenção humana. Mas muitas vezes a resolução de

problemas envolve a combinação de meios humanos e meios computacionais. No pensamento computacional incluem-se diversas dimensões tais como a decomposição do problema, o reconhecimento de padrões e regularidades, processos de generalização, construção de modelos, design de algoritmos, recolha e análise de dados e a sua visualização.

Pensamento Computacional

Objetivos

Assume-se os seguintes objetivos no âmbito do **Pensamento Computacional**:

- compreender as dimensões envolvidas no pensamento computacional;
- identificar estratégias de abordagem de problemas (redução da complexidade, decomposição, abstração, adaptação ou adoção de modelos e algoritmos conhecidos, recolha e análise de dados, etc);
- problematizar situações do quotidiano e formular problemas;
- descrever e representar simbolicamente sequências de ações de atividades do quotidiano em diferentes graus de complexidade;
- resolver problemas pela sua decomposição em partes menores, por semelhança ou redução de complexidade.

Padrões de desempenho

Sugerem-se os seguintes padrões de desempenho no âmbito do **Pensamento Computacional**:

Iniciais

- Compreende que os computadores precisam de instruções precisas para resolver problemas.
- Demonstra cuidado e precisão na formulação de problemas para antecipar erros futuros.
- Utiliza dados numéricos e textuais para organizar informação e produzir conclusões.
- Decompõe problemas complexos em problemas mais simples.
- Reconhece a importância de recolher diferentes tipos de dados (texto, numérico, gráfico).
- Utiliza métodos numéricos e geométricos simples para análise de dados com vista a detetar padrões e regularidades.

Intermédios

- Elabora mapas conceptuais simples e descreve as relações entre conceitos através de exemplos.
- Desenha esquemas operativos para futura implementação de algoritmos.
- Compreende e utiliza etapas básicas na resolução de problemas.
- Identifica tipos de dados necessários para resolver um problema.
- Reconhece que os dados podem ser organizados e estruturados em tabelas úteis para a sua análise.
- Analisa conjuntos de dados com vista a gerar informação que contribui para a resolução de um problema.

Avançados

- Identifica categorias a partir de dados de natureza qualitativa agrupando esses dados segundo critérios explícitos.
- Identifica variáveis presentes num problema e avalia a necessidade e viabilidade de as considerar na construção de um algoritmo para a resolução de um problema.
- Compreende as limitações dos modelos matemáticos que cria e avalia a sua plausibilidade face a situações reais.
- Identifica estratégias de resolução de problemas que utilizam auto-semelhança na criação de soluções por métodos recursivos.
- Generaliza relações entre variáveis criando modelos matemáticos parametrizados.
- Estabelece domínios de validade de modelos matemáticos.
- Identifica isomorfismos entre problemas.
- Importa e adapta modelos matemáticos para problemas isomorfos.

Algoritmia

Objetivos

Assume-se os seguintes objetivos no âmbito da **Algoritmia**:

- compreender o que são algoritmos, como funcionam e sua aplicação prática;
- descrever e representar simbolicamente sequências de ações de atividades do quotidiano;
- reconhecer a importância do desenho de algoritmos como método de resolução de problemas;
- resolver problemas pela sua decomposição em partes menores;
- compreender que diferentes algoritmos podem atingir o mesmo resultado e que um mesmo algoritmo pode ser reutilizado em diferentes situações;
- reconhecer que alguns algoritmos são mais apropriados para um contexto específico do que outros;
- reutilizar um mesmo algoritmo em diferentes situações.

Padrões de desempenho

Sugere-se os seguintes padrões de desempenho no âmbito da **Algoritmia**:

Iniciais

- Usa etapas básicas na construção de algoritmos simples para conceber soluções de problemas.
- Reconhece utilidade na reutilização de algoritmos simples existentes na construção de algoritmos mais complexos.
- Traduz relações entre conceitos em termos das variáveis que os definem.
- Estabelece relações entre ações com sequencialidade lógica na construção de um algoritmo.
- Combina e articula algoritmos simples.

Intermédios

- Representa simbolicamente sequências de ações na resolução de problemas.
- Utiliza variáveis bem identificadas presentes num problema para definir relações funcionais na construção de um algoritmo.
- Demonstra a compreensão de algoritmos, sua aplicação prática e os resultados produzidos.
- Transforma algoritmos extensos em algoritmos articulados mais simples.
- Estabelece sequências lógicas de algoritmos para executar uma dada tarefa.
- Demonstra capacidade crítica na avaliação da plausibilidade de operacionalizar computacionalmente algoritmos simples.

Avançados

- Cria algoritmos pela decomposição de problemas complexos em componentes mais simples e articuladas.
- Compreende que diferentes algoritmos podem produzir os mesmos resultados, mas que em determinados contextos uns são mais adequados e eficientes do que outros.
- Adapta algoritmos conhecidos articulando-os em novos algoritmos para resolver problemas específicos.
- Revê criticamente algoritmos complexos identificando e isolando erros com vista à sua correção.
- Aplica métodos numéricos na construção de algoritmos.

Programação e Robótica

No domínio da programação pretende-se desenvolver a capacidade de codificação de algoritmos em ambientes e linguagens específicos concebidos para solucionar um dado problema ou situação. Se o conhecimento de algoritmia é fundamental para o desenvolvimento de competências em diversas áreas do conhecimento, a programação permite assim colocar em prática esse conhecimento.

A aprendizagem da codificação de algoritmos em programas de computador através de uma determinada linguagem, ou ambiente, tem como principal objetivo dar corpo a soluções tanto de problemas específicos, ou a desafios como por exemplo a criação de histórias e animações, jogos interativos ou aplicações para dispositivos móveis.

Neste sentido, importa abordar a aprendizagem dos fundamentos da programação e suas especificidades, bem como de diferentes linguagens ou ambientes de programação por blocos ou textuais, de acordo com a maturidade, faixa etária, motivação e conhecimentos dos alunos. São diversos os ambientes de programação adequados à aprendizagem da programação por crianças, desde a programação por blocos simbólicos, direcionada a crianças que ainda não sabem ler, blocos textuais, mais simplificadas pela facilidade de uso, a linguagens de programação propriamente ditas, com maior liberdade e flexibilidade, mas adequadas a uma faixa etária a partir do 3º ciclo.

De acordo com a faixa etária, a maturidade, a motivação, os conhecimentos e padrões de desempenho apresentados pelos alunos, poderá utilizar-se o ambiente ou linguagem de programação que o professor considere mais adequado. As atividades a desenvolver deverão ter em consideração a adequação à faixa etária dos alunos, padrões de desempenho e contexto em que serão integradas. Salienta-se igualmente que existem outros ambientes e aplicações, algumas dependentes de determinada tecnologia, e que, a qualquer momento, poderão deixar de se encontrar disponíveis.

A **robótica**[6] permite tornar tangíveis os conceitos ligados à programação e ao pensamento computacional, ou seja, fora do espaço dum ecrã de computador. Deste modo, é possível aprender a criar, planear, resolver problemas, programando artefactos tangíveis, e, consequentemente, construindo algo com uma finalidade e que proporciona a articulação com conteúdos das diferentes áreas do saber. A robótica propicia uma aprendizagem mais profunda da tecnologia, promovendo momentos para "aprender fazendo", de forma táctil, na relação que o aluno estabelece ao relacionar as suas ideias com os artefactos, processo durante o qual os alunos obtêm e visualizam resultados imediatos.

A integração da robótica em contexto educativo permite criar cenários de aprendizagem diversificados, que reúnem tecnologia, linguagens de programação e objetos tangíveis; promovendo-se assim a articulação com as áreas curriculares e/ou transversais, onde se realizam projetos contextualizados que no seu conjunto proporcionam aos alunos a

[6] Iniciação à Programação no 1º Ciclo do Ensino Básico - Linhas Orientadoras para a Robótica: http://www.erte.dge.mec.pt/sites/default/files/linhas_orientadoras_para_a_robotica.pdf

oportunidade de desenvolver a sua criatividade e ter um papel ativo na construção do seu próprio conhecimento.

Considerando o universo dos artefactos programáveis, o termo robótica evidencia uma abrangência um pouco limitada no domínio da programação tangível. Deste modo assume-se a designação de Objetos Tangíveis Programáveis [OT], para uma maior abrangência que inclua robots, drones, plataformas de prototipagem baseadas em microcontroladores, entre outros.

Pela multiplicidade de projetos e atividades a desenvolver, diversidade de artefactos, tipologia e níveis de dificuldade de programação, níveis de autonomia e estrutura física, no âmbito da iniciativa Probótica consideram-se três grupos de objetos tangíveis programáveis, e que se encontram articulados com os padrões de desempenho definidos.

Nos padrões de desempenho iniciais consideram-se OT simples, com estrutura física inalterável ou de reduzida flexibilidade; sem sensores ou de diversidade muito limitada; com predominância de funções pré-definidas; e reduzida autonomia na interação com o ambiente; a título de exemplo referem-se o BeeBot, Dash&Dot, Lego WeDo, Osmo, Ozobot, Zowi, Drones, entre outros.

Nos padrões de desempenho intermédios consideram-se OT modulares, com estrutura física alterável e com grande flexibilidade e personalização; com diversidade de sensores e possibilidade de interação entre eles; uma grande autonomia na interação com o ambiente, utilizando sensores; como exemplo referem-se o Anprino, Bot'n'Roll, BQ PrintBot, Lego EV3, mBot, Picaxe, entre outros.

Nos padrões de desempenho avançados consideram-se OT prototipados, baseados em microcontroladores e microprocessadores com estrutura física totalmente flexível e personalizável; propiciando a integração de materiais externos, por exemplo recicláveis; vasta diversidade de sensores e possibilidade de interação entre eles; e, uma grande autonomia na interação com o ambiente; tais como o Arduino, Bitalino, Raspberry Pi, entre outros.

Se o professor considerar vantajoso, poderá considerar a utilização de outras tipologias de artefactos programáveis e com potencial pedagógico. Os OT nos exemplos apresentados anteriormente são meramente indicativos e poderão ser utilizados em níveis diferentes de aprofundamento dos conceitos.

De salientar que o nível de aprofundamento dos conceitos e dificuldade de programação não dependem apenas dos tipos de OT a utilizar, mas também dos contextos e desafios que se pretendem superar.

Programação

Objetivos

Assume-se os seguintes objetivos no âmbito da **Programação**:

- compreender e aplicar os princípios e conceitos fundamentais da programação (lógica, tipos de dados, variáveis, estruturas condicionais e repetitivas, entre outros);
- analisar programas, identificando o seu resultado, erros e respetiva correção;
- otimizar a programação da solução encontrada para determinado problema;
- desenhar programas com diversos níveis de complexidade na resolução de problemas específicos;
- criar programas para resolver problemas, animar histórias ou jogos utilizando uma linguagem de programação textual ou ambiente de programação por blocos.

Padrões de desempenho

Sugerem-se os seguintes padrões de desempenho no âmbito da **Programação**:

Iniciais

- Compreende e utiliza etapas básicas na programação para resolução de problemas.
- Utiliza adequadamente diferentes tipos de dados.
- Conhece a necessidade de utilização de variáveis para armazenamento de dados de diversos tipos.
- Compreende os diferentes operadores aritméticos, relacionais e lógicos.
- Conhece diferentes estruturas de controlo (seleção e repetição).
- Conhece e utiliza eventos simples e pré-definidos.
- Identifica a necessidade de várias instruções serem executadas simultaneamente.
- Associa eventos gerados por periféricos (rato, teclado, etc.) a ações concretas do programa.
- Conhece a necessidade de testar programas e analisar os seus resultados

Intermédios

- Aplica as operações adequadas a cada tipo de dados.
- Utiliza os tipos de variáveis apropriados para manipular dados de entrada, de processamento e de saída.
- Utiliza operadores e expressões aplicando-os em estruturas de controlo.
- Utiliza adequadamente diferentes estruturas de controlo.
- Utiliza estruturas modulares (funções) pré-definidas na linguagem utilizada.
- Compreende os conceitos de objeto, propriedade, método e evento.
- Cria blocos de instruções prevendo a sua execução em paralelo.
- Trabalha com diversas formas de entrada e saída de informação.
- Usa o raciocínio lógico para detetar e corrigir erros em programas e prever o seu comportamento.
- Testa passo a passo (*tracing*) a execução de programas analisando os resultados parcelares.

Avançados

- Demonstra a compreensão de programas, sua aplicação prática e resultados produzidos.
- Cria programas pela decomposição de problemas complexos em partes menores.
- Deteta e corrige erros em programas, acompanhando a sua execução passo-a-passo.
- Analisa, manipula e converte dados de diferentes tipos.
- Analisa e cria programas de pesquisa e ordenação na resolução de problemas concretos.
- Seleciona e utiliza apropriadamente diferentes estruturas de dados.
- Aplica operadores booleanos em estruturas de controlo.
- Utiliza adequadamente estruturas de controlo simples, compostas e aninhadas.
- Cria estruturas modulares para melhor gestão de tarefas complexas.
- Define propriedades, métodos e eventos em objetos.
- Cria eventos a serem desencadeados por outros eventos.
- Explica e otimiza a execução de blocos de instruções em paralelo.
- Trabalha com diferentes formas de entrada e saída de informação.
- Reconhece que diferentes problemas partilham as mesmas caraterísticas e utiliza o mesmo programa para solucioná-los.
- Analisa se os programas desenvolvidos solucionam problemas específicos.
- Deteta se os resultados gerados pelo programa são os expectáveis e analisa formas para a sua melhoria e otimização.

Robótica: objetos tangíveis programáveis (OT)

Objetivos

Assume-se os seguintes objetivos no âmbito da **Robótica**:

- compreender o que é suposto os OT fazerem;
- caraterizar robots, drones e computação física;
- distinguir OT nas suas caraterísticas, funcionalidades e aplicabilidade;
- adequar atuadores e sensores à resolução de situações específicas;
- programar OT que façam uso de atuadores e sensores para interagir com o ambiente em que se integram;
- manipular dados de entrada e de saída;
- adequar a estrutura de OT a contextos específicos;
- criar OT que interajam com o mundo físico;
- programar OT para resolução de desafios simples e desafios complexos;
- detetar e corrigir erros de programação e desadequação de estruturas físicas a situações específicas.

Padrões de desempenho

Sugere-se os seguintes padrões de desempenho no âmbito da **Robótica**:

Iniciais

- Carateriza os conceitos dos diversos OT.
- Compreende a importância dos OT no quotidiano.
- Conhece a estrutura física de OT simples e avalia as suas limitações.
- Compreende as diferentes funcionalidades dos atuadores (motores) na locomoção.
- Compreende conceitos associados à locomoção de OT: potência, velocidade, distância, direção.
- Programa OT para se movimentarem de forma simples em cenários específicos.
- Distingue diversos sensores e suas funcionalidades.
- Compreende como diferentes sensores ajudam os OT a interagir com o ambiente onde se encontram.
- Programa OT para produção de informação de output em formatos simples ou pré-definidos.
- Programa OT, através de instruções pré-definidas ou ambiente de programação específico, para resolução de problemas simples.
- Programa OT, através do ambiente de programação selecionado, para resolução de problemas simples que necessitem da utilização de sensores.
- Utiliza o raciocínio lógico para prever os resultados.
- Avalia as soluções encontradas e procede a correções e melhorias.

Intermédios

- Analisa o impacto da robótica na sociedade e os normativos de segurança associados.
- Distingue as diversas tipologias de OT.
- Distingue os vários componentes mecânicos e eletrónicos.
- Aplica conceitos associados à locomoção de OT: potência, velocidade, distância, direção.
- Utiliza atuadores noutras funcionalidades para além da locomoção.
- Adequa a estrutura física do OT, a aplicação de atuadores, e a sua forma de locomoção a diferentes situações.
- Programa OT para se movimentarem em ambientes complexos.
- Utiliza adequadamente diferentes sensores em situações específicas.
- Articula a funcionalidade de diferentes sensores na interação com o ambiente para resolução de problemas específicos.
- Analisa e incorpora os dados recolhidos por sensores em ações específicas de interação com o ambiente onde os OT se integram.
- Programa OT para produção de informação de output em diferentes formatos (som, luz, entre outros).
- Programa OT, através do ambiente de programação específico, para resolução de problemas complexos que necessitem de interação com o ambiente onde se integram.
- Avalia as soluções encontradas e procede a correções, melhorias e otimizações.

Avançados

- Avalia perspetivas futuras da robótica e inteligência artificial na sociedade.
- Compreende o conceito de Computação Física.
- Descreve caraterísticas de voltagem, corrente e resistência.
- Compreende o que é um circuito e o seu funcionamento.
- Distingue as diferentes componentes de placas controladoras.
- Descreve a utilização dos componentes eletrónicos fundamentais.
- Distingue sinais analógicos de digitais.
- Compreende a utilização de operadores lógicos, estruturas de dados, estruturas condicionais e ciclos.
- Cria e utiliza estruturas modulares (funções).
- Compreende a função dos sensores em Computação Física.
- Identifica sensores analógicos e digitais e compreende as suas funcionalidades.
- Cria circuitos simples utilizando placas de ensaio (breadboards).
- Distingue os vários componentes mecânicos e eletrónicos.

Avançados

- Compreende a necessidade e o processo de calibração.
- Programa microprocessadores que manipulem valores de sensores e dados de saída.
- Programa OT que operem atuadores e sensores.
- Constrói OT com estrutura física adequada à resolução de situações específicas e em que a interação com o ambiente seja necessária.
- Avalia as soluções encontradas e procede a correções, melhorias e otimizações.

ORIENTAÇÕES METODOLÓGICAS E ESTRATÉGIAS DE OPERACIONALIZAÇÃO

A iniciativa Programação e Robótica no ensino básico assume-se como uma iniciativa multidisciplinar de caráter prático, investigativo, experimental e sobretudo criativo. Considerando esta premissa, o professor deve adotar metodologias e estratégias de trabalho que proporcionem aos alunos a oportunidade de analisarem, investigarem, experimentarem e proporem soluções para problemas concretos. Sugere-se, deste modo, a adoção de metodologias de aprendizagem ativas e colaborativas que privilegiam a participação dos alunos, a articulação de saberes, a colaboração, o pensamento crítico, a resolução de problemas, o raciocínio lógico, a partilha e a comunicação. Todas estas competências têm vindo a ser apresentadas como competências essenciais para os cidadãos do séc. XXI.

A articulação de saberes das várias disciplinas ou áreas disciplinares deverá ser colocada em prática através da realização de projetos ou da resolução de problemas que permitam aos alunos encarar a programação e a robótica não como um fim em si mesmas, mas como ferramentas transversais promotoras da aprendizagem de diversos saberes. Assim, é importante que o professor promova a articulação com outros professores da turma (da turma ou da escola), privilegiando as áreas onde possam vir a desenvolver projetos de forma articulada e contextualizada.

O professor deverá, ainda, privilegiar um conjunto de estratégias que motivem o aluno a envolver-se na sua e na aprendizagem dos colegas e lhe permitam desenvolver a sua criatividade, autonomia e iniciativa.

Resnick (2016) pensando no desenvolvimento do pensamento criativo e da criatividade propõe uma abordagem iterativa da aprendizagem assumindo organização de atividades em espiral representada na figura 1. De acordo com esta proposta, o aluno, imagina, cria, brinca, partilha, reflete e volta ao ponto de partida para iniciar outro ciclo.

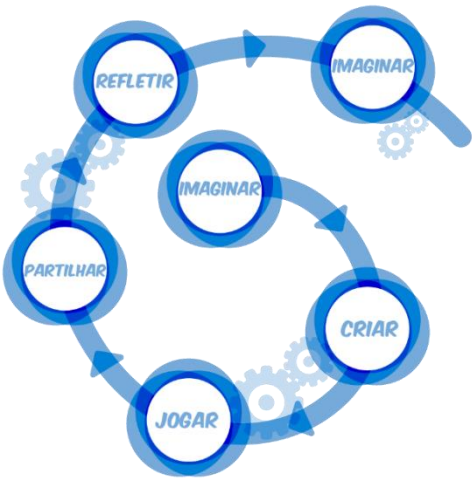


Figura 1 - Espiral de pensamento criativo proposta por Resnick (2007)

Metodologias ativas de aprendizagem

Em modo de sugestão apresentamos três exemplos de metodologias de aprendizagem/estratégias ativas que podem ser mobilizadas para dinamizar atividades de programação e robótica na construção de Cenários de Aprendizagem, a saber: a) Aprendizagem Baseada em Projetos; b) Aprendizagem Baseada em Problemas; e c) Pair Programming

Aprendizagem Baseada em Projetos

A Aprendizagem Baseada em Projetos é uma metodologia de aprendizagem ativa que aposta no envolvimento dos alunos no desenvolvimento de projetos que pretendem solucionar problemas concretos. Envolve a pesquisa, a investigação, a colaboração, a multidisciplinaridade e **no final resulta na definição e apresentação de um produto** que procura solucionar um determinado problema.

Esta metodologia de aprendizagem procura integrar conhecimentos de diferentes áreas e estimula o desenvolvimento de competências, como trabalho em equipa, o pensamento crítico, a criatividade, a resolução de problemas, entre outras. Deste modo, procura que o aluno tenha um papel ativo na sua aprendizagem, bem como no desenvolvimento do projeto, tornando a sua aprendizagem adequadamente contextualizada e significativa.

A aprendizagem e o desenvolvimento de competências para o séc. XXI deverão estar no centro das preocupações do projeto a desenhar e a desenvolver pelos alunos. O desenvolvimento de atividades suportadas por esta metodologia organiza-se em torno de 7 elementos estruturantes representados na figura 2.

Figura 2 - Elementos estruturantes da metodologia de aprendizagem baseada em projetos (adaptado do Buck Institute of Education)

Esta abordagem metodológica inicia-se com a identificação de um problema a resolver ou de um desafio e com definição de uma questão orientadora que, não sendo de resposta fácil, deverá ser estimuladora e desafiadora para os alunos. De seguida, os alunos, de forma colaborativa, aplicam conhecimentos prévios de várias áreas, investigam e procuram informação sobre os temas, idealizam e refletem sobre propostas de solução autênticas e apresentam as suas ideias aos restantes colegas e professores, na procura de críticas e sugestões de melhoria. No final, deverá existir uma apresentação pública (por exemplo, na escola ou em eventos) dos produtos finais desenvolvidos.

Deste modo, esta metodologia pode permitir: o desenvolvimento de competências para o séc. XXI e de aprendizagem ao longo da vida, melhorar a motivação dos alunos, melhor a aprendizagem, criar oportunidades de pensar o futuro, entre outras.

Aprendizagem Baseada em Problemas

A aprendizagem baseada em problemas foi apresentada por Howard Barrows, nos anos 60, associada às ciências médicas, com o objetivo de possibilitar aos alunos de medicina a aplicação e integração dos conceitos apreendidos, em problemas concretos nos contextos clínicos.

A aprendizagem baseada em problemas é uma abordagem ativa, centrada no aluno, em que estes adquirem novos conhecimentos e desenvolvem novas competências trabalhando e colaborando em grupo na resolução de problemas abertos e semiabertos. Assenta na utilização de problemas como ponto de partida para a aquisição e desenvolvimento de novas aprendizagens. Deste modo, os problemas devem funcionar como um estímulo para a aprendizagem dos conceitos necessários à sua resolução. A aprendizagem dos alunos é centrada em temáticas interdisciplinares e não em temáticas individualizadas das várias disciplinas.

Tal como na aprendizagem baseada em projetos, esta metodologia promove o trabalho em equipa, a colaboração, o pensamento crítico e criativo, a resolução de problemas, o desenvolvimento de competências de comunicação, entre outras competências para o séc. XXI. A principal diferença entre estas duas opções metodológicas reside no fator tempo, normalmente o desenvolvimento de um projeto pressupõe uma necessidade de tempo superior comparativamente à resolução de problemas, tendencialmente mais curtos. Outro aspeto relevante prende-se com as propostas de soluções finais, em que a metodologia baseada em projetos aponta para a criação de um produto concreto enquanto que na aprendizagem baseada em problemas o resultado final pode ser apenas uma ou várias propostas de solução para o problema.

Na figura 3 apresenta-se uma proposta de organização de atividades suportadas na aprendizagem baseada em problemas.

Figura 3 – Atividades suportadas na aprendizagem baseada em problemas

Pair Programming

Não sendo claramente uma metodologia de aprendizagem, o *Pair Programming* é uma estratégia pedagógica utilizada na aprendizagem da programação, que pode ser utilizada de forma autónoma ou articulada, com estratégia noutras metodologias de aprendizagem.

A programação em pares (*pair programming*) organiza-se em torno de dois programadores que programam partilhando um único computador ou dispositivo. Um dos elementos do par assume a função de "*driver*" e é responsável pela tarefa de programação, o outro assume a função de "*navigator*" e é responsável por analisar o programa, procurando eventuais falhas e pensando nos próximos passos. A figura 4 representa a organização das atividades usando esta estratégia. Uma das características desta estratégia de programação é a troca de papéis regular, ou seja, o elemento com a função de "*driver*" assume por vezes a função de "*navigator*" e vice-versa. Ambos os programadores (alunos) devem estar fortemente envolvidos nas atividades de programação e trabalhar de forma colaborativa na resolução dos problemas.

A programação em pares enquanto estratégia pedagógica vem sendo referida por alguns autores como um método eficiente na aprendizagem da programação, na medida em que privilegia a aprendizagem ativa e colaborativa, ajuda os alunos a ultrapassar a frustração quando estão perante problemas desafiadores e aumenta a autoconfiança e interesse pela programação. A atividade de programação realizada entre pares apresenta benefícios ao nível académico destacando que as soluções encontradas para os problemas podem ser elaboradas e discutidas com maior detalhe. Ao trabalhar em pares, os alunos argumentam e defendem as suas opiniões, sendo que desta forma as aprendizagens tornam-se mais efetivas.

A utilização desta estratégia pedagógica permite: i) melhorar a qualidade dos programas e soluções, reduzindo os erros e melhorando a sua eficiência; ii) incrementar a partilha de conhecimento entre pares; iii) promover o desenvolvimento de competências; iv) promover a capacidade de resiliência; e v) facilitar o acompanhamento por parte do professor.

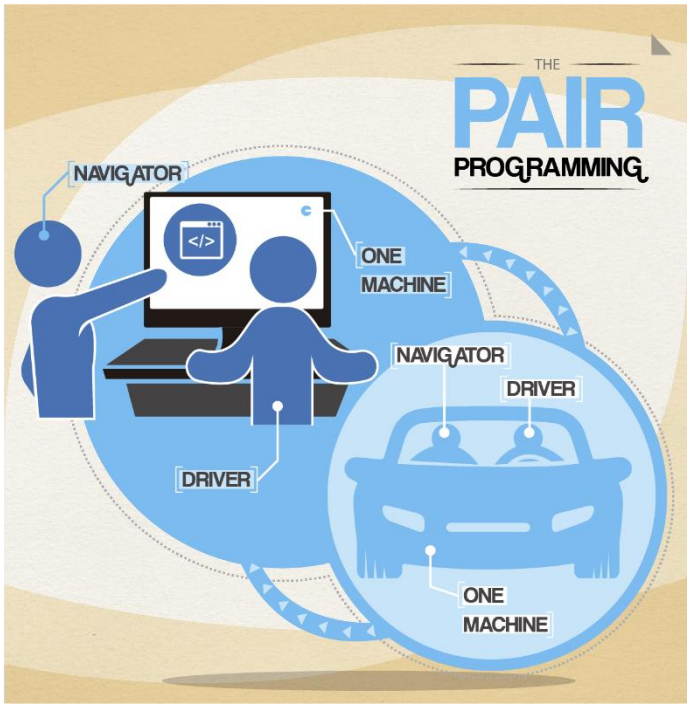


Figura 4 - Operacionalização da estratégia pedagógica pair programming
(adaptado de: https://developer.atlassian.com/blog/2015/05/try-pair-programming)

Cenários de Aprendizagem

Pensar e idealizar cenários de aprendizagem é algo que o professor faz, de forma mais ou menos consciente e organizada, na sua prática profissional aquando do processo de planificação das suas atividades pedagógicas. Procura desenhar e antecipar várias situações de aprendizagem a implementar junto dos seus alunos na sala de aula.

Um cenário de aprendizagem pode ser uma forma criativa de prever e antecipar o futuro, estimulando o pensamento crítico, que permita sair das formas pré-estabelecidas de pensar e planear as ações.

Assim, entende-se por cenário de aprendizagem uma situação hipotética de ensino-aprendizagem (puramente imaginada ou com substrato real, amplamente mutável) composta por um conjunto de elementos que descreve o contexto no qual a aprendizagem tem lugar, o ambiente em que a mesma se desenrola e que é condicionado por fatores relacionados com a área/domínio de conhecimento, pelos papéis desempenhados pelos diferentes agentes ou atores (e pelos seus objetivos), que se estabelece com um dado enredo, incluindo sequências de eventos, criando uma determinada estrutura coordenada numa dada tipologia de atividades.

A figura seguinte apresenta de forma resumida as principais características de um cenário de aprendizagem.

Figura 5 - Características de um Cenário de Aprendizagem

Genericamente, um cenário de aprendizagem assume um conjunto de características, em particular:

- **Inovação** – um cenário deve ser desenhado para demonstrar possíveis atividades inovadoras e não para fornecer planos prescritivos aos professores.
- **Transformação** – um cenário deve encorajar os professores a experimentar mudanças nas suas praticas pedagógicas e métodos de ensino e de avaliação, e fazer surgir experiencias educativas inovadoras com sucesso.
- **Previsão / antevisão** – um cenário deve ser considerado como uma ferramenta de planeamento utilizada para pensar em novas maneiras de perspetivar o futuro e tomar decisões apropriadas relativamente a condições incertas.
- **Imaginação** – um cenário deve ser sempre uma fonte de inspiração e de alimentação da criatividade do professor e alunos. Deve conduzir à aprendizagem do que ainda não é conhecido.

- **Adaptabilidade** – um cenário não deve ser apresentado de forma rígida. Cabe ao professor adaptá-lo aos seus objectivos e às características dos seus alunos. A profundidade da exploração dos temas, assim como o tempo necessário para a concretização das atividades, deverão ficar ao critério de cada professor. Um cenário pode sugerir o nível de escolaridade para o qual os temas e as atividades propostas são mais indicadas. No entanto, as ideias para um determinado nível de ensino podem ser adaptadas pelo professor para alunos mais novos ou mais velhos.
- **Flexibilidade** – um cenário deve fornecer opções dirigidas a diferentes estilos de aprendizagem e a estilos individuais de ensino. Os professores podem escolher usar parte de um determinado cenário ou apenas uma ideia inspirada nele. Pode também escolher a escala em que quer aplicar o cenário. Podem usá-lo a um nível elementar ou torná-lo mais complexo.
- **Amplitude / abrangência** – um cenário pode ser construído de modo a possuir uma maior ou menor abrangência. O papel dos atores pode estar confinado apenas ao nível das operações e das ações ou pretender-se que sejam participantes ativos do sistema de atividade completo. Os cenários podem incluir projetos multidisciplinares para serem trabalhados pelos alunos durante extensos períodos de tempo.
- **Colaboração / partilha** – um cenário pode conter elementos conducentes à realização de atividades colaborativas (síncronas e assíncronas), incluindo ferramentas tecnológicas propiciadoras de partilha e de construção colaborativa de artefactos.

Constituem elementos estruturantes de um cenário de aprendizagem:

a) o desenho organizacional do ambiente - organização dos elementos contextuais de um cenário, requisitos (incluindo convicções e conceções), artefactos materiais;

b) os papéis e atores - posturas e responsabilidades, formas de estar, organização do coletivo, modos de interação e comunicação;

c) o enredo, estratégias de trabalho, atuações e propostas - arquitetura da atuação, estrutura de atividade, sentido teleológico da construção;

d) a reflexão e regulação - processos de reificação do aprendido e da ação, monitorização do desenvolvimento próprio dos atores e do contexto, e avaliação crítica dos produtos.

A figura 6 ilustra alguns princípios orientadores para o desenho de cenários de aprendizagem.

Figura 6 - Princípios orientadores para o design de cenários de aprendizagem

AVALIAÇÃO

Embora a iniciativa de Programação e Robótica no Ensino Básico possa ser implementada numa grande diversidade de cenários, é importante que sejam pensados à partida os mecanismos de avaliação das aprendizagens dos alunos de modo a respeitar as opções globais definidas para cada escola ou agrupamento em que seja implementado. Sugere-se que se privilegie uma avaliação baseada em evidências recolhidas ao longo das sessões em que os alunos desenvolvem as suas atividades e projetos.

A avaliação deverá ser sempre formativa, implicando o retorno qualitativo aos alunos dos resultados obtidos e respetivo processo.

Para tal podem ser utilizados os instrumentos que se adequem em cada momento, como grelhas de observação, listas de verificação, registo de notas, rubricas, questionários, entre outros.

Outros elementos a ter em consideração serão os produtos elaborados pelos alunos e o modo como os apresentam e documentam. Deverá ficar claro, no entanto, que, mais que o produto, interessa o processo e o modo como cada grupo trabalhou para chegar a esse produto.

A avaliação baseada na aplicação de fichas, testes ou provas, teóricas ou práticas, não está em momento algum contemplada na iniciativa Probótica.

Os registos das atividades dos alunos, feitos em áudio, foto ou vídeo deverão ser, obrigatoriamente, autorizados pelos encarregados de educação e pela escola, a quem deve ser comunicada a finalidade e o âmbito em que serão utilizados.

Sugere-se a aplicação de processos de autoavaliação, heteroavaliação e de avaliação por grupos (nos grupos e pair programming). Para a avaliação da programação por pares ou grupos recomenda-se a consulta dos recursos disponíveis em "*Pair Programing-in-a-Box: The power of collaborative learning*[7]".

No Anexo I apresenta-se, a título de exemplo, as rubricas de desempenho[8] propostas por Brenann, Balsch e Chung (2014), traduzidas por Ramos e Espadeiro (2015), salvaguardando a necessidade de serem adaptadas a cada contexto.

[7] Pair Programing-in-a-Box: The power of collaborative learning. National Center for Women & information technology (2009). Disponível em: www.ncwit.org/pairprogramming

[8] Disponíveis em: http://dspace.uevora.pt/rdpc/bitstream/10174/14227/1/challenges%202015br.pdf

Anexo I – Rubricas para avaliação das aprendizagens e processos de desenvolvimento de projetos

Experimentar e interagir	Baixo	Médio	Alto - Elevado
Descreve como construíste o teu projeto, passo a passo.	Fornece uma descrição elementar da construção do projeto, mas não detalha aspetos específicos do mesmo.	Faz uma descrição genérica do projeto, de forma ordenada.	Fornece detalhes sobre as diferentes componentes dum projeto específico e descreve o modo como foram desenvolvidos, de forma ordenada.
Que outras coisas foste experimentando ao longo da elaboração do projeto?	Não apresenta exemplos específicos do que experimentou.	Deixa transparecer de forma genérica que experimentou outras coisas no projeto.	Fornece exemplos específicos de outras coisas que foi experimentando no projeto.
Que revisões fizeste e porque é que as fizeste?	Afirma não ter feito revisões, ou afirma ter feito algumas, mas não exemplifica.	Descreve uma revisão específica que fez ao projeto.	Descreve aspetos específicos de coisas que acrescentou ao projeto e justifica.
Descreve as diferentes abordagens que experimentaste no teu projeto, ou quando tentaste fazer algo novo.	Não revela evidências de ter experimentado algo novo.	Fornece um exemplo de algo novo que experimentou no projeto.	Descreve com detalhe coisas novas que experimentou no projeto.

Testar e corrigir	Baixo	Médio	Alto - Elevado
Descreve o que aconteceu com o teu projeto de diferente em relação ao pretendido.	Não descreve o que resultou diferente em relação ao pretendido.	Descreve o que correu mal no projeto, mas não o que pretendia fazer.	Dá um exemplo detalhado do que aconteceu e o que pretendia, quando executa o programa.
Descreve de que forma fazes a leitura do código para encontrar a causa do problema.	Não descreve um problema.	Descreve como faz a leitura, mas não apresenta um exemplo específico de encontrar um problema no código.	Descreve como faz a leitura e apresenta um exemplo específico de encontrar um problema no código.
Descreve como fizeste alterações e testaste para verificar os resultados.	Não descreve que problemas teve ou a solução.	Fornece um exemplo genérico sobre as alterações feitas e os testes feitos para verificar o funcionamento.	Fornece um exemplo específico sobre as alterações feitas e os testes feitos para verificar o funcionamento.
Descreve como considerarias outras formas de resolver o problema.	Não apresenta uma forma para encontrar uma solução para o problema.	Apresenta uma forma genérica para encontrar uma solução para o problema.	Apresenta um exemplo específico de como encontrar uma solução para o problema.

Reutilizar e recombinar	Baixo	Médio	Alto - Elevado
Descreve se encontrou inspiração em outros projetos e na leitura do código disponível.	Descreve como desenvolveu as ideias ou em que projetos se inspirou.	Fornece uma descrição geral de um projeto que o inspirou.	Dá um exemplo específico do projeto que o/a inspirou.
Descreve como selecionou uma parte de outro projeto e a adaptou ao seu projeto	Não descreve como adaptou as ideias, scripts ou recursos de outros projetos.	Identifica scripts, ideias ou recursos que adaptou de outros projetos.	Fornece exemplos específicos de scripts, ideias ou recursos é/ele adaptou de outros projetos e como.
Como refere/cita as pessoas cujo trabalho inspirou o seu próprio.	Não identifica as fontes e os autores em que se inspirou no projeto.	Identifica as fontes e os autores em que se inspirou no projeto.	Documenta no projeto as fontes e os autores que inspiraram o projeto.

Abstrair e modularizar	Baixo	Médio	Alto
Como foi decidido que *sprites* eram necessários para o projeto e onde eram utilizados.	Não descreve que *sprites* foram selecionados.	Fornece uma descrição geral da decisão de escolher certos *sprites*.	Dá uma explicação detalhada acerca de como selecionou os *sprites* em função do objetivo do projeto.
Como foi decidido que *scripts* eram necessários para o projeto e onde eram utilizados.	Não descreve que *scripts* foram criados.	Fornece uma descrição geral da decisão de criar certos *scripts*.	Dá uma explicação detalhada acerca de como criou os *scripts* em função do objetivo do projeto.
Como foram organizados os *scripts* de forma a terem significado para o estudante e para os outros.	Não descreve como os *scripts* foram organizados.	Fornece uma descrição geral da forma como foram organizados *scripts*.	Dá uma explicação detalhada acerca de como organizou os *scripts* e porquê.

REFERÊNCIAS

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge. Educational Technology & Society, 19 (3), 47-57.

Berry, M. (2013). The computing curriculum beyond 2014. Retirado de www.slideshare.net/mgberry/the-computing-curriculum-beyond-2014

Bryant, S., Romero, P., & Du Boulay, B. (2008). Pair programming and the mysterious role of the navigator. International Journal of Human-Computer Studies, 66 (7), 519-529. Elsevier Ltd. Retirado de http://linkinghub.elsevier.com/retrieve/pii/S1071581907000456

Buck Institute for Education, 2017. Gold Standard PBL: Essential Project Design Elements. Retirado de www.bie.org/blog/gold_standard_pbl_essential_project_design_elements

Curzon, P., Dorling, M., Ng, T., Selby, C. & Woollard, J. (2014). Developing computational thinking in the classroom: a framework. Retirado de https://pdfs.semanticscholar.org/e4f3/c24924c5a707a196b2015494c829c15618d1.pdf

Dreamfeel - Technology, Digital World, Design, Interactive Marketing and Multimedia Trends (2009). Computação Física. Retirado de https://dreamfeel.wordpress.com/2009/03/07/computacao-fisica

DGE. (2015). Iniciação à Programação no 1º Ciclo do Ensino Básico - Linhas Orientadoras Gerais. Retirado de http://www.erte.dge.mec.pt/sites/default/files/Projetos/Programacao/IP1CEB/linhas_orientadoras.pdf

DGE. (2016). Iniciação à Programação no 1º Ciclo do Ensino Básico - Linhas Orientadoras para a Robótica. Retirado de http://www.erte.dge.mec.pt/sites/default/files/linhas_orientadoras_para_a_robotica.pdf

Fernandes, S. (2014). Aprendizagem Baseada em Projetos na Consolidação de Conceitos de Programação de Linguagens Script (Relatório da Prática de Ensino Supervisionada de Mestrado em Ensino da Informática, apresentado ao Instituto de Educação da Universidade de Lisboa). Lisboa: Universidade de Lisboa.

Fluck, A., Webb, M., Cox, M., Angely, C., Malyn-Smith, J., Voogt, J., & Zagami, J. (2016). Arguing for Computer Science in the School Curriculum. Educational Technology & Society, 19 (3), 38-46.

Freeman, A., Adams Becker, S., Cummins, M., Davis, A., and Hall Giesinger, C. (2017). NMC/CoSN Horizon Report: 2017 K–12 Edition. Austin, Texas: The New Media Consortium

Graça, A. (2011). Aprendizagem baseada em problemas no ensino profissional da informática: aplicação a uma turma do curso técnico de multimédia - 10º ano na unidade de noções de layout (Relatório da Prática de Ensino Supervisionada de Mestrado em Ensino da Informática, apresentado ao Instituto de Educação da Universidade de Lisboa). Lisboa: Universidade de Lisboa.

ISTE (2011). Computational Thinking - teacher resources. Retirado de http://www.iste.org/docs/ct-documents/ct-teacher-resources_2ed-pdf.pdf?sfvrsn=2

Matos, J.F. (2010). Princípios Orientadores para o Desenho de Cenários de Aprendizagem. Lisboa: Projeto LEARN.

Markham, T. (2015). Project Based Learning Handbook. Blurb, Incorporated.

McDowell, C., Werner, L., Bullock, H., & Fernald, J. (2006). Pair Programming Improves Student Retention, Confidence, and Program Quality. Communications of the ACM, 49, (8), 90-95. Retirado de https://users.soe.ucsc.edu/~charlie/pubs/cacm.pdf

Naace, ITTE, and the Computing at School Working Group. (2012). ICT and Computer Science in UK schools. Retirado de http://www.computingatschool.org.uk/data/uploads/ICT and CS joint statement.pdf

National Center for Women & information technology (2009). Pair Programing-in-a-Box: The power of collaborative learning. Retirado de www.ncwit.org/pairprogramming

Olabe, X., Basogain, M. & Basogain, J. (2015). Pensamiento Computacional a través de la Programación: Paradigma de Aprendizaje. RED-Revista de Educación a Distancia, 46(6). Retirado de http://www.um.es/ead/red/46/Basogain.pdf

Partnership for 21-st century skills (2009). Framework for 21st Century Learning. Retirado de http://www.p21.org/index.php?option=com_content&task=view&id=254&Itemid=120

Tucker, A., McCowan, D., Deek, F., Stephenson, C., Jones, J., & Verno, A. (2006). A model curriculum for K–12 computer science: Report of the ACM K–12 task force curriculum committee (2nd ed.). New York, NY: Association for Computing Machinery.

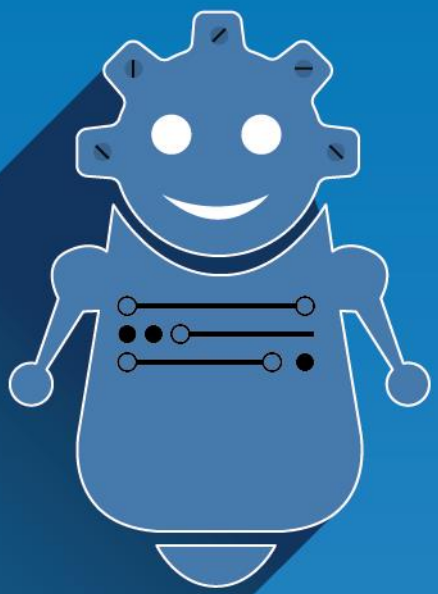
Ramos, J.L. & Espadeiro, R.G. (2014). Os futuros professores e os professores do futuro. Os desafios da introdução ao pensamento computacional na escola, no currículo e na aprendizagem. *Educação, Formação & Tecnologias*, 7 (2), 4-25. Retirado de https://eft.educom.pt/index.php/eft/article/download/ 462/208

Ramos, J.L. & Espadeiro, R.G. (2015) Pensamento computacional na escola e práticas de avaliação das aprendizagens. Uma revisão sistemática da literatura. Retirado de http://dspace.uevora.pt/rdpc/bitstream/10174/ 14227/1/challenges%202015br.pdf

Sousa, S. (2016). Utilização da estratégia Pair Programming no ensino da programação de sistemas de comunicação com recurso a sockets (Relatório da Prática de Ensino Supervisionada de Mestrado em Ensino da Informática, apresentado ao Instituto de Educação da Universidade de Lisboa). Lisboa: Universidade de Lisboa.

UNESCO. (2011). Digital literacy in education; Policy brief. Retirado de http://unesdoc.unesco.org/images/0021/002144/214485e.pdf

Wing, J. (2006). Computational Thinking. Communications of the ACM. Retirado de https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf



REPÚBLICA PORTUGUESA | EDUCAÇÃO
direção-geral da educação
PORTUGAL INCoDe.2030