DOI: 10.36558/rsc.v13i3.8547

Aplicação de metodologias ativas em disciplina de Lógica de Programação no IFPE Jaboatão dos Guararapes

Havana Diogo Alves
Instituto Federal de
Pernambuco - Campus
Jaboatão dos Guararapes
Recife - Pernambuco - Brasil
havana.alves@gmail.com

ABSTRACT

This article reports on the application of active methodologies as a teaching-learning strategy in the Programming Logic subject, aiming to improve failure and dropout indicators in this subject. After presenting the scenario that motivated the application of this type of methodology, we describe the ways in which Problem-Based Learning, Team-Based Learning and the Flipped Classroom were inserted. As a result, after six semesters of using these active methodologies, the result was satisfactory, reducing the failure rate due to absence and grade.

RESUMO

Este artigo traz o relato da aplicação de metodologias ativas como estratégia de ensino-aprendizagem em disciplina de Lógica de Programação, visando melhorar indicadores de reprovação e evasão desta disciplina. Após apresentarmos o cenário que motivou a aplicação desse tipo de metodologia, descrevemos as formas em que foram inseridas o Aprendizado Baseado em Problemas, Aprendizado Baseado em Equipes e a Sala de Aula Invertida. Como resultado, após seis semestres de utilização destas metodologias ativas, o resultado mostrou-se satisfatório, diminuindo o índice de reprovações por falta e por nota.

CCS Concepts

•**Applied computing** → **Collaborative learning;**

Keywords

metodologias ativas; ensino de programação

1. INTRODUÇÃO

O desafio de ensinar e aprender lógica de programação faz parte do dia a dia de professores e alunos dos cursos de Computação dos níveis técnico e superior, dificuldade que vem mantendo os índices de retenção nos períodos iniciais desses cursos em níveis alarmantes. 43% dos alunos de computação, brasileiros, falharam em disciplinas introdutórias de programação, já incluindo um número alarmante de 25% de abandono da disciplina [24]. O estudo de [14] mostra que os alunos desistem dos cursos logo no primeiro ano do ensino superior de computação, e as disciplinas causadoras desta desistência são aquelas associadas ao ensino de Cálculo e de Programação.

De fato, escrever um programa de computador não é uma tarefa simples. Segundo [21] e [3], para programar é necessário ao estudante, entre outras habilidades, compreender o problema interpretando os requisitos em um algoritmo, formular a solução utilizando técnicas padrão de resolução de problemas, compreender a sintaxe da linguagem de programação, de tal forma que um computador possa seguir as instruções, encontrar erros (*bugs*) e usar o ambiente de desenvolvimento do programa.

A revisão sistemática de [16] identificou as seguintes dificuldades dos estudantes em relação ao uso de linguagens de programação:

- Falta de habilidade para relacionar a sintaxe da linguagem aos conceitos de lógica;
- Falta de habilidade em compreender os problemas e traduzi-los em uma sequência lógica de instruções;
- Alguns estudantes chegam a saber a sintaxe e até a semântica dos tipos de blocos de código, individualmente, porém não conseguem combiná-los de forma a criar um código de programa válido para resolver o problema em questão;
- Dificuldades em utilizar bibliotecas e em encontrar e utilizar função ou propriedade apropriadas ao contexto que se está desenvolvendo;
- Ponteiros, listas e estrutura de dados são os conceitos mais difíceis de serem aprendidos;
- Dificuldade com o conceito de modularização, no que consiste na criação e utilização de funções, com ou sem passagem de parâmetros e com ou sem retorno;
- Impossibilidade de uso das ferramentas de auxílio ao desenvolvimento como compiladores e debugadores, aliada a falta de entendimento do significado das mensagens de alerta e de erro.

Conhecimentos prévios desestruturados, principalmente nos domínios matemático e lógico, resultam em não compreensão dos enunciados dos problemas e de conceitos abstratos [7, 14].

Em relação às formas de ensino e de aprendizagem de programação, [7, 14] identificam abordagens pouco motivadoras e conteúdos pouco relacionados ao cotidiano dos sujeitos, hábitos de estudo pouco disciplinados e centrados em memorização.

Os problemas elencados nos parágrafos anteriores foram também identificados durante os primeiros três anos de ensino da disciplina de Lógica de Programação e Estrutura de Dados (LPED) do curso Técnico em Informática para Internet do Instituto Federal de Pernambuco, Campus Jaboatão dos Guararapes. O percentual médio de falhas, as quais incluem reprovações e evasões desde as primeiras turmas do curso iniciado no 2º semestre de 2014 (entrada 2014.2) até as turmas do 2º semestre de 2017 (entrada 2017.2), passava dos 66% dos matriculados.

Por ser disciplina do primeiro período do curso e pré-requisito para várias outras dos períodos subsequentes, o alto índice de reprovação neste componente curricular resultava em altas taxas de retenção e de evasão. Desta forma, várias foram as tentativas de diminuir o alto índice de reprovação em LPED, iniciando de simples mudanças nas linguagens de programação utilizadas nas aulas até a culminância da aplicação de metodologias ativas como *Aprendizado Baseado em Problemas* (ABP), *Aprendizado Baseado em Equipes* (ABE)e Sala de Aula Invertida.

2. CENÁRIOS INICIAIS

Desde a implantação do curso, em 2014.2 até o semestre letivo de 2017.2, a metodologia de ensino da disciplina de LPED seguia o modelo tradicional expositivo, geralmente com o seguinte roteiro: (1) explanação do professor, (2) execução de exercícios em sala, (3) listas de exercícios para a prática em casa; (4) atividade avaliativa escrita ou utilizando computador ao final de cada bimestre. Era possível perceber alguns problemas resultantes deste tipo de metodologia:

- O tipo de aula expositiva não estimulava os estudantes a se manterem atentos à explanação do conteúdo da aula, causando a dispersão de boa parte dos alunos durante as aulas;
- Apenas pequena parte dos alunos se sentia motivada a resolver os exercícios em sala de aula;
- As listas de exercícios, que têm como objetivo estimular os estudantes a praticarem a resolução de problemas fora da sala de aula eram tratadas de duas formas: (1) se a lista não tivesse alguma pontuação atribuída, poucos eram os alunos a tentarem resolvê-la ou (2) se houvesse uma pontuação atrelada, era identificado um alto índice de plágio nas resoluções.

Durante esse tempo, foram realizadas várias tentativas para mitigar as falhas no aprendizado, indo desde a mudança de professores até a escolha de diferentes linguagens de programação.

De 2014.2 a 2015.2, utilizou-se a linguagem Java, a qual trazia algumas dificuldades como abstrair o conceito de classe de objetos, visto que é necessária a criação de uma classe para o "programa rodar"; o ambiente de desenvolvimento utilizado, como o Eclipse [1] ou o Visual Studio Code [2], necessita da realização de algumas configurações por parte do usuário, bem como da intervenção do professor em alguns momentos de falha da IDE. Bom tempo da aula presencial era despendido para a resolução de problemas com a IDE, desviando também a atenção do aluno do processo de ensino-aprendizagem em relação à compreensão do algoritmo. Os trabalhos [23] e [15] indicam que linguagens de programação com sintaxes inadequadas e ambientes de desenvolvimento complexos são fatores que dificultam o aprendizado de estudantes principiantes.

Na tentativa de contornar estas dificuldades, em 2016.1, a linguagem Java foi substituída por *JavaScript*, por ter sintaxe simples e necessitar apenas de um editor de texto simples e um navegador para ser implementada. As aulas continuaram seguindo a mesma metodologia: expositiva, exercícios realizados em sala, listas de exercícios valendo pontuação e provas individuais. Essa mudança de linguagem de programação mostrou-se irrelevante, o que pode ser corroborado nos dados a seguir.

Na turma vespertina apenas 1 aluno, dos 39 matriculados foi aprovado. Dos 38 estudantes reprovados, 23 não obtiveram a média mínima (que é 6,0) para aprovação e quinze reprovaram por falta, onde oito evadiram-se após receber a nota da primeira prova (as quais tiveram variação de 0 a 3,0) e sete nunca frequentaram as aulas.

Na turma noturna houve a aprovação de apenas 7 alunos, dos 47 matriculados. 20 dos alunos matriculados e que frequentaram a primeira unidade evadiram-se da disciplina após o recebimento da nota da primeira prova. Este valor corresponde a 42% dos matriculados na turma. 12 estudantes reprovaram por não terem atingido a média mínima (que é 6,0) para passarem na disciplina, onde o *range* de média final foi de 0 a 5,2. Ainda em relação aos 47 matriculados, 8 evadiram-se ainda na primeira unidade.

Cabe aqui comentar sobre a quantidade de matriculados na turma noturna de 2016.1. Cada turma deve ter no máximo 40 estudantes matriculados, porém, em alguns casos, é possível que se permita matricular uma pequena quantidade a mais de estudantes que ficaram retidos em semestres anteriores. Este cenário de super lotação de sala de aula, obviamente, é um cenário nada favorável para professores e alunos.

A alta quantidade de reprovações ocorridas em 2016.1 resultou em uma alta taxa de retenção, o que gerou a necessidade de se oferecer uma turma extra de LPED para matricular os alunos retidos que desejavam cursar a disciplina novamente.

2.1 Turmas Extras

Em 2016.2, além das turmas regulares nos turnos vespertino e noturno, uma turma extra da disciplina de LPED foi criada, utilizando também o horário da tarde. Esta turma foi composta apenas por alunos retidos nos semestres anteriores. O desafio passou a ser a de como ensinar os mesmos assuntos para alunos que não conseguiram obter êxito anteriormente, sem que estes esmorecessem e desistissem do curso.

É necessário que a metodologia utilizada em turmas formadas em grande parte por repetentes seja diferenciada e

[1] https://www.eclipse.org/

[2] https://code.visualstudio.com/

pedagogicamente adequada para mantê-los motivados [28]. Segundo [25], o aluno torna-se mais interessado em um ambiente descontraído e colaborativo do que em um ambiente instrucionista em que o aluno é sujeito passivo e apenas recebe o conhecimento vindo do professor.

Após algumas pesquisas, decidiu-se basear o primeiro bimestre no ambiente de programação visual *Scratch*[3], com foco prático na resolução de problemas disponibilizados em listas de exercícios. A ideia de utilizar o Scratch é a de poder forcar na semântica dos blocos de código, não tendo que se preocupar com erros de sintaxe. Foram abordados os assuntos de variáveis, operadores, cláusulas condicionais e blocos de repetição. Os assuntos de listas e funções ficaram para o segundo bimestre.

No segundo bimestre, os códigos que antes foram implementados no Scratch passaram a ser escritos em JavaScript.

24 alunos matricularam-se nessa turma, onde 16 deles foram aprovados. Dos oito reprovados, apenas dois evadiram-se após a nota da primeira unidade.

Em 2017.1, houve a necessidade de abrir-se uma nova turma extra, onde 16 estudantes resolveram matricular-se. Nesta turma, o índice de aprovação foi bem menor. Apenas 6 foram aprovados, correspondendo a 37,5% dos matriculados.

A experiência das turmas extras nos permitiu identificar alguns pontos interessantes. Apesar de maior foco na prática, as aulas continuaram seguindo o modelo expositivo, porém, nos momentos em que as aulas eram iniciadas com resolução de exercícios (baseando-se em problemas), o índice de entrega das listas mostrava-se maior e com mais qualidade. O fato de os alunos estarem sendo observados e estimulados pelo professor favoreceu a tentativa de resolução das atividades e, consequentemente, o surgimento de perguntas para sanar dúvidas. Desta forma, os estudantes tiveram que participar mais ativamente das aulas e não apenas como meros ouvintes, como era anteriormente praticado.

A disponibilização de listas para serem iniciadas e conduzidas fora da sala de aula continuou sendo nada eficaz, pois a negligência para resolver e a tendência a copiar resoluções alheias ainda se fez presente.

Em relação ao *Scratch*, a reação inicial dos alunos - que tinham idades entre 18 e 50 anos - era de subestimação do ambiente, por conta de sua apresentação "infantilizada", porém no decorrer das aulas, os estudantes perceberam que a ferramenta não era tão "boba" como eles imaginavam. É importante frisar que não fazia parte da estratégia pedagógica utilizar todo o potencial da ferramenta, como a criação de animações. O uso foi concentrado na resolução de problemas comuns de ensino de programação, tais como, criar um algoritmo para calcular a área de um triângulo, ou criar uma calculadora das quatro operações básicas.

Em paralelo às turmas extras, as turmas regulares seguiam suas aulas utilizando Portugol[4] na primeira unidade letiva e JavaScript na segunda.

Um ponto importante a ressaltar é que desde o início do curso, houve certa rotatividade de professores lecionando a disciplina, trazendo consigo a natural adequação e personalização de métodos e instrumentação pedagógica. A Tabela 1 exibe a configuração das aulas nas turmas regulares em cada um dos semestres letivos de 2014.2 a 2017.2. Importante salientar que em todos estes semestres tivemos turmas nos turnos vespertino e noturno. Nas colunas indicadas como **Linguagem UN1** e **Linguagem UN2** estão as linguagens de programação utilizadas, respectivamente, na 1ª e na 2ª unidades de cada semestre letivo. P1 a P4 representam cada um dos quatro professores diferentes que ministraram a disciplina. É possível observar que todas as variáveis mudaram exceto a metodologia de ensino-aprendizagem, que continuou sendo expositiva, com certa atuação passiva dos estudantes. Tendo em mãos esta análise, iniciaram-se pesquisas sobre formas de ensinar que motivassem o aprendizado dos alunos.

[3]https://scratch.mit.edu/

[4]https://univali-lite.github.io/Portugol-Studio/

3. METODOLOGIAS ATIVAS PARA O ENSINO DE PROGRAMAÇÃO

As Metodologias Ativas (MAs) são estratégias de ensino centradas na participação ativa dos estudantes, auxiliando na construção do processo de aprendizagem. Estas vêm sendo estudadas e indicadas como formas de proporcionar maior interesse do aluno sobre o assunto a ser aprendido, reduzindo as taxas de reprovação, além de promover o pensamento crítico e uma compreensão mais profunda [5, 2, 13].

Existem várias metodologias de ensino e aprendizado ativo, que têm, em geral, como premissas:

- motivar o porquê de se estar estudando determinado conteúdo;
- tornar o estudante protagonista de seu aprendizado;
- Estimular a colaboração entre colegas de grupo.

Há várias publicações que relatam o uso de algum tipo de MA para ensino e aprendizagem de programação. A revisão sistemática de Calderon e colegas [8] traz a análise de 21 publicações, de 2010 a 2021, que descrevem a aplicação de ao menos um tipo de MA durante o ensino de programação em instituições brasileiras. O trabalho de Berssanete e colegas [5] mapeou 38 publicações, de 2014 a 2019, relatando o uso de alguma técnica de aprendizagem ativa em vários países. Em conjunto, os dois trabalhos identificaram, entre outras técnicas, a utilização das seguintes propostas de MAs: Aprendizado Baseado em Problemas (ABP), Aprendizado Baseado em Projetos (ABPj), Sala de Aula Invertida, Gameficação, Instrução por pares, Aprendizado Baseado em Equipes (ABE), Método Baseado em Tutoriais.

A importância da utilização de MAs para o ensino de programação cresceu tanto que o trabalho de Calderon [26] descreve a criação de um repositório colaborativo de apoio à adoção de MAs no ensino de programação.

Assim, após algumas pesquisas, a partir do semestre 2018.1 foram utilizados de forma isolada ou combinada três das várias MAs existentes: o Aprendizado Baseado em Problemas (ABP), a Sala de Aula Invertida (SAI) e o Aprendizado Baseado em Equipes (ABE).

O Aprendizado Baseado em Problemas foi inicialmente implantado em 1964, na Universidade McMasters, localizada no Canadá [1]. O ABP tem como base a teoria racionalista de conhecimento, segundo a qual o conhecimento dos fatos é primariamente um produto da nossa capacidade de pensamento e, portanto, de dedução. Esta metodologia vem sendo bastante utilizada por professores de computação, devido à capacidade de desenvolver as competências necessárias a um profissional da área, como resolução de problemas, trabalho colaborativo e busca auto-gerida do conhecimento necessário

Table 1: Mudanças realizadas no decorrer dos sete primeiros semestres letivos

Semestre	Professor	Linguagem UN1	Linguagem UN2	Aula
2014.2	P1	Java	Java	Expositiva
2015.1	P2	Java	Java	Expositiva
2015.2	P3	Java	Java	Expositiva
2016.1	P4	Java Script	Java Script	Expositiva
2016.2	P4	Portugol	Java Script	Expositiva
2017.1	P4	Portugol	Java Script	Expositiva
2017.2	P4	Portugol	Java Script	Expositiva

para se alcançar os objetivos [30, 22, 9, 12, 17]. O trabalho *A review of project-base learning (pbl) and computational thinking (ct) in teaching and learning* de Saad e Zainudin [27] encontrou diversas publicações, de 2017 a 2021, descrevendo implementações de ABP como forma eficaz de ensinar e aprender pensamento computacional. A revisão sistemática realizada por Coelho e Guedes [10] encontrou 35 trabalhos relatando a utilização de ABP para o ensino de programação de computadores.

Algumas pesquisas, como as de Cohen [11], McNeil[18] e de Zhang [31], indicam que aulas em vídeo são tão efetivas quanto aquelas realizadas presencialmente na transmissão de informações básicas. Assim, aulas pré-gravadas podem ser atribuídas a alunos como lição de casa, a serem vistas antes da abordagem do assunto em sala de aula. O ideal é que a aula, de fato, seja utilizada para realização de atividades de aprendizagem interativas, debates e retirada de dúvidas.

O rótulo de Sala de Aula Invertida é mais frequentemente atribuído a cursos que utilizam atividades compostas de videoaulas assíncronas baseadas na web e a posterior resolução de listas de exercícios, problemas fechados ou questionários. O trabalho de Bishop e colegas [6] definem a SAI como uma técnica educacional que consiste em duas partes: atividades interativas em grupo em sala de aula e estudo individual realizado por tutoriais pré-gravados fora de sala de aula. Berrett [4] escreve que a SAI pode ser conduzida de várias formas e utilizar combinações de diferentes técnicas de ensino-arpendizagem, como instrução por pares, atividades interativas e o chamado ensino "just-in-time", onde os estudantes respondem a questões online antes da aula de determinado assunto, o que permite ao professor ter um *feedback* sobre o conhecimento prévio dos alunos e assim conduzir suas aulas da melhor forma possível, porém, todas as técnicas compartilham da mesma premissa: Os estudantes não podem receber o material em classe, de forma passiva, estes devem obter a maior parte do conhecimento fora da sala de aula através de leituras, assitindo a vídeo-aulas ou ouvindo podcasts.

O Aprendizado Baseado em Equipes (ABE) é uma estratégia de ensino desenvolvida Larry Michaelsen, nos anos 1970 [20]. Esta metodologia busca obter os benefícios da realização de trabalho em pequenos grupos de aprendizagem. Faz parte da proposta estimular os estudantes ao estudo prévio para as atividades em classe, ou seja, pode ser tranquilamente combinada com Sala de Aula Invertida. Assim como as duas metodologias apresentadas anteriormente, baseia-se no construtivismo, em que o professor se torna um facilitador para a aprendizagem em um ambiente despido de autoritarismo.

4. APLICAÇÃO DAS METODOLOGIAS ATIVAS

Nos semestres letivos 2018.1 e 2018.2, a metodologia escolhida foi o Aprendizado Baseado em Problemas.

Em geral, a ABP possui 7 fases, as quais podem ser adaptadas de acordo com a necessidade da disciplina. O processo de aprendizado é iniciado quando um problema é proposto pelo professor, que também disponibiliza os materiais instrucionais que irão servir como base teórica inicial para o desenvolvimento da solução. Neste processo, o professor atua como facilitador, dirimindo dúvidas, guiando e estimulando as equipes. É fundamental que os problemas criados para se trabalhar com ABP sejam bem estruturados, estimulando o debate de ideias e formulação do problema [29].

4.1 Aplicando ABP aliado ao Scratch

Aliada à metodologia ABP, a linguagem visual *Scratch* mais uma vez foi escolhida para o desenvolvimento das soluções propostas na primeira unidade de cada semestre letivo. Desta vez, o objetivo foi utilizar o recurso de criar animações completas mesmo utilizando poucos blocos de código *baby steps*. O objetivo de cada problema proposto era o de proporcionar ao estudante um caminho de aprendizado gradual, trabalhando de dois a três conceitos de progamação em cada projeto. A aplicação de ABP-Scratch foi dividida em 6 etapas, conforme descritas a seguir:

1. **Elaboração dos problemas:** Os problemas precisam ser elaborados visando uma sequência pedagógica de complexidade, e de acordo com os itens do conteúdo programático. Além disso, é importante observar a viabilidade de resolução colaborativamente e estimar o tempo para se adequar ao tempo disponível de aula.

2. **Organização dos grupos:** Os alunos são divididos em grupos de no máximo quatro componentes, e cada membro recebe um dos papéis: líder, relator, desenvolvedor ou pesquisador. Estes papéis eram sempre atribuídos de forma aleatória.

3. **Disponibilização do problema e dos materiais instrucionais:** O professor deve explicar o problema a ser trabalhado durante a aula. Neste momento, é fundamental apresentar os objetivos de aprendizagem e mostrar o problema como motivador dessa aprendizagem. Os materiais instrucionais são, então, disponibilizados para serem utilizados já em sala de aula. Estes recursos de estudo podem e devem ser os mais variados possível como por exemplo, vídeos de como utilizar os blocos do Scratch, apostilas, links de tutorais, podcasts e demais fontes de conhecimento. Neste momento é importante deixar claro que o aluno deve buscar materi-

ais complementares. Não obstante, é essencial deixar claro quais são os entregáveis, os prazos e os meios de submissão.

4. **Apresentação das animações:** Em data pré-agendada, cada grupo apresenta o resultado de seu trabalho a toda a turma, explicando como se deu o desenvolvimento do código, e a utilização dos conceitos que tiveram de ser aprendidos para solucionar o problema proposto.

5. ***Feedback* para turma:** Ao término do prazo, o professor deve retomar o problema apresentado, fazendo uma retrospectiva sobre os conceitos trabalhados e o desenvolvimento destes.

6. **Registro da prática:** Fazendo parte do conjunto de entregáveis, havia a criação de um relatório sobre o decorrer da criação da solução em equipe.

Em 2018.1, os grupos foram formados no primeiro dia de aula, onde os membros foram escolhidos pelos próprios alunos a partir de possíveis afinidades, ou até por proximidade em relação à moradia. Este tipo de formação, em geral agrada mais aos alunos, porém resulta em pouca diversidade intragrupo, o que pode não ser interessante para estudantes que estão sendo formados para atuarem em um mercado de trabalho, onde em grande parte das vezes, os colegas de trabalho não poderão ser escolhidos.

Já em 2018.2, os grupos foram formados aleatoriamente para cada projeto, o que não agradava muito aos alunos, mas que era necessário para simular ambientes em que os profissionais teriam de trabalhar com pessoas diferentes.

Os problemas propostos foram elaborados de forma a seguirem grau de dificuldade gradual, e que são descritos nas próximas subseções.

4.1.1 Problema Proposto 1 - Diálogos

O primeiro problema proposto teve como objetivos de aprendizagem os conceitos de algoritmo, instruções e interação com o usuário. Estas foram as competências necessárias para a criação de uma animação consistindo de um diálogo entre dois personagens e deveria contemplar os seguintes requisitos:

- Cada personagem deveria ter no mínimo três falas, onde o nome de um deles deveria ser fornecido pelo usuário. Este requisito visa motivar a utilização de instruções de interação com o usuário;
- Os personagens deveriam se movimentar pelo cenário, o que exigia a utilização de mudança de coordenadas no palco da animação.

.

Os alunos podiam escolher seus personagens, logo alguns sentiram-se livres para procurar os seus preferidos, mesmo que não fizessem parte da instalação padrão do *Scratch*. Este tipo de liberdade permite que o estudante utilize recursos que estão ligados ao seu cotidiano. A Figura 1 exibe parte da animação criada por um grupo da turma vespertina de 2018.1, onde foram utilizados *sprites* de personagens da série de desenhos Dragon Ball Z [5]. Na referida figura é possível perceber que os estudantes utilizaram um bloco de repetição, que estava além dos conceitos-alvo deste projeto, mostrando que há certa facilidade na compreensão da semântica proporcionada pelo uso de blocos.

[5]https://pt.wikipedia.org/wiki/Dragon_Ball_Z

4.1.2 Problema Proposto 2 - Jogo de Perguntas e Respostas

Neste problema havia o desafio de criar um jogo do tipo Perguntas e Respostas, onde as perguntas estariam ligadas a temas específicos. As perguntas de cada tema, deveriam ter pesos diferentes no cálculo da pontuação total, necessitando assim, que os alunos utilizassem cálculo de média ponderada.

O jogo deveria ter no mínimo duas fases, sendo necessária uma pontuação mínima para passar de fase ou para ganhar o jogo.

Os objetivos de aprendizagem foram a compreensão dos conceitos de variáveis, operadores aritméticos, lógicos e relacionais e cláusulas condicionais simples.

4.1.3 Problema Proposto 3 - Jogo de Colisão

A proposta consiste na criação de um jogo de colisão, tipo de jogo onde a personagem permanece "vivo" enquanto esta não entrar em colisão com elementos específicos do cenário.

A solução para este problema necessita do conceito de blocos de repetição, sendo este o objetivo de aprendizagem. Infelizmente, os blocos de repetição nativos do Scratch possuem algmas diferenças em relação aos blocos da maioria das linguagens textuais. Como exemplo, o bloco *repita até que...*, o qual pode ser visto na Figura 2, executa as instruções internas ao bloco enquanto a expressão lógica - montada dentro do espaço em formato de hexágono - for falsa. Este comportamento difere do bloco *while* das linguagens de programação textual, onde as instruções são executadas enquanto a expressão lógica resultar em verdade (*true*). Utilizar este bloco nativo do *Scratch* poderia causar confusão no entendimento do bloco de código da próxima linguagem utilizada, no caso o JavaScript. A solução encontrada foi lançar mão de um bloco *while* customizado e disponibilizados pela comunidade que desenvolve recursos para serem utilizados com a ferramenta.

Em todos os projetos, os grupos deveriam submeter o código da animação e um relatório sobre o processo de desenvolvimento.

A avaliação dos alunos foi realizada de forma contínua desde o dia da proposição de cada problema. Atitudes como a não participação nas discussões em grupo, a falta de cumprimento das tarefas atreladas ao papel incumbido e ausências durante as aulas, eram anotadas e um retorno, de forma o mais imediata possível, era dado ao aluno. Para permitir o melhor acompanhamento das equipes em sala, o *layout* do laboratório teve de ser alterado do formato tradicional, em que as bancadas ficam todas voltadas para o quadro branco, para o formato em"U" [19], pois esta forma de dispor as bancadas facilita a movimentação de professores e monitores no ambiente, e permite observar o que os alunos estão vendo ou fazendo nos computadores, permitindo a intervenção quando necessário.

Como os projetos eram iniciados durante as aulas, mas precisavam ser desenvolvidos em outros momentos, tornou-se necessário o acompanhamento do andamento do trabalho de cada grupo, também, de forma remota. Com este intuito foi disponibilizado o Diário de Bordo [6], recurso que integra alguns Ambientes Virtuais de Aprendizagem (AVAs) como o Moodle, para colher relatos individuais dos alunos sobre cada um dos projetos. Assim, tanto o processo de desenvolvimento, bem como possíveis problemas com o grupo,

[6]https://moodle.org/plugins/mod_journal

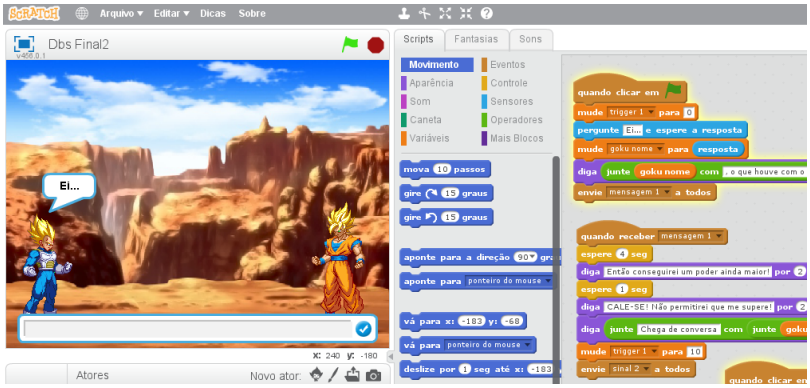

Figure 1: Animação criada por uma das equipes como solução para o Problema Proposto 1

Figure 2: Bloco repita até que... disponível no Scratch

tais quais conflitos de relacionamento e dificuldades na divisão e execução de tarefas, entre outras informações, eram estimulados a serem incluídos no texto.

A tabela 2 exibe relatos de dois alunos em 2018.1 e 2018.2. O monitoramento destes relatos deveria ser o mais célere possível para permitir intervenções necessárias de forma eficaz. Um sistema que detectasse problemas nos relatos, em tempo real, ajudaria bastante, porém o Moodle não permite esse tipo de monitoramento. Esta tarefa apenas tornou-se possível com a atuação de alunos monitores. Destaca-se aqui, que diante da necessidade de mediação de vários grupos de forma simultânea, ter monitores atuantes durante as aulas e também em horários vagos, mostrou-se de grande importância na implantação da metodologia.

Foi possível perceber, em ambos os semestres, que os alunos das turmas do turno vespertino se mostravam mais empolgados em utilizar o *Scratch*, diferentemente do percebido nas turmas noturnas. Isto pode ter acontecido pela característica etária de cada turno: à tarde a média de idade dos alunos é de 23 anos e em sua grande maioria não possuem outra atividade além do referido curso, enquanto que à noite temos alunos com média de idade acima dos 26 e que boa parte possuem outras atividades durante o dia e consequentemente, requerem um nível maior de objetividade e seriedade.

A desvantagem de ter seguido com o *Scratch* durante todo semestre letivo foi a de que os alunos só tiveram contato com a linguagem textual no semestre seguinte.

4.2 Aplicando Aprendizado Baseado em Equipes combinado à Sala de Aula Invertida

Nos semestres letivos de 2019.1, 2019.2, 2022.1 e 2022.2, outra abordagem passou a ser utilizada em ambos os turnos: a combinação de Aprendizado Baseado em Equipes e Sala de Aula Invertida. Nos semestres de 2020.1 a 2021.2 não houve aplicação de qualquer tipo de abordagem presencial, devido à pandemia de COVID-19, momento em que as aulas passaram a ser remotas. Como vimos na seção 3, a ABE e a SAI são totalmente compatíveis para uso em conjunto, o que se deu da seguinte forma:

1. Cada turma foi dividida em grupos de no máximo 4 membros;
2. O conteúdo a ser abordado em cada momento era disponibilizado, no mínimo, uma semana antes em plataforma online, contendo vídeos (exemplo na Figura 3, textos e listas de problemas referentes ao assunto ministrado;
3. As listas de problemas eram compostas por uma lista de exercícios, utilizada para os estudantes praticarem e retirarem dúvidas durante as aulas presencias, e por uma lista avaliativa, a qual deveria ser estudada e resolvida em grupo, para ser executada em data marcada. Na data agendada, a da execução da lista avaliativa, dava-se de acordo com a sequência a seguir:
   (a) Sorteava-se um grupo por vez. O grupo sorteado tinha um dos seus componentes também sorteado. Este componente sorteado, então, deveria resolver, em computador reservado para tal, uma das questões da lista avaliativa sorteada no momento;
   (b) Este procedimento era realizado até que todos os problemas da lista fossem resolvidos.

Nos casos em que o aluno sorteado não conseguisse resolver o problema no tempo estipulado, ou se recusasse a tentar resolver, o grupo todo não pontuava para aquela lista. O aluno que faltava também não pontuava, mesmo que o grupo conseguisse a pontuação por meio de aluno presente que fosse sorteado.

Ao final de cada unidade letiva, os estudantes que, porventura, não obtiveram a pontuação em alguma das listas, tinham a possibilidade de recuperar a nota referente àquele assunto abordado nos problemas. Esta recuperação consistia em prova prática individual.

Table 2: Exemplos de relatos em Diário de Bordo

aluno X em 2018.1	"Sobre o grupo, dessa vez eu fiz tudo sozinho, com apenas uma ajuda de *aluna Z*, *aluno W* e *aluno P* perguntaram sobre o trabalho na greve porem nao fizerem nada, e sumiram novamente(...)"
aluna Y em 2018.2	"(...) O grupo me ajudou muito, por eu ser a desenvolvedora cabia a mim fazer a maior parte do projeto, mas por está em semana de prova na faculdade, pedi ao grupo que entendesse e que dividíssemos o projeto, para não ficar tão pesado para mim, e eles entenderam e me ajudaram bastante.(...)"

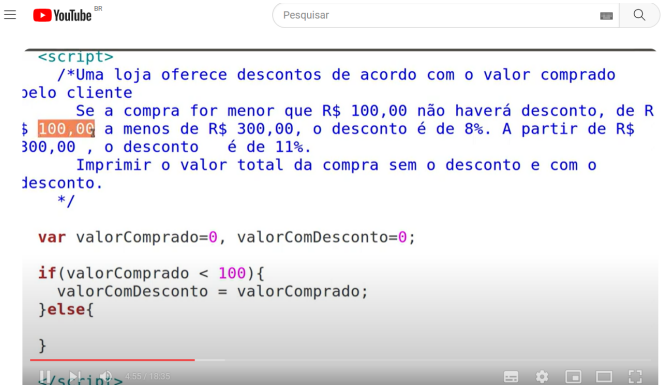

Figure 3: Exemplo de vídeo instrucional disponibilizado para os estudantes.

O que se quis estimular e o que foi percebido durante a execução desta abordagem foi a cooperação inter e intra-grupo. Visto que o plágio era uma constante nas execuções de listas de exercícios no modelo tradicional, o compartilhamento de respostas passou a ser, inclusive, estimulado. O objetivo era que, devido à aleatoriedade dos sorteios de membros dos respectivos grupos e também dos problemas, os estudantes tinham muita necessidade de entenderem a resolução dos problemas e também de ajudar seus colegas neste entendimento. Dessa forma, os alunos com mais habilidade em um grupo ajudavam aqueles que não estavam compreendendo bem a resolução de cada problema. Como também não havia disputa entre os grupos, a cooperação intergrupo era bastante frequente.

De início, os alunos tentavam apenas decorar as soluções, o que poderia até ser possível na primeira lista, devido ao nível de dificuldade do conteúdo abordado ser relativamente mais fácil. Essa estratégia, porém, não funcionava a partir da segunda lista, pois a dificuldade aumentava e, simplestemente, decorar a solução não era mais uma tarefa simples, tendo o estudante que, de fato, entender o que se estava codificando.

Um ou outro aluno mais aplicado reclamou que não achava justa a possibilidade de serem prejudicados por colegas menos interessados. Porém, esses conseguiam recuperar suas notas na prova prática individual.

5. COMPARANDO OS RESULTADOS

Foram analisados os dados de reprovações por nota e de evasões para comparar os resultados das turmas onde se utilizaram aulas expositivas e as turmas onde metodologias ativas foram aplicadas.

Para realizar as análises, retiramos de cada turma os alunos que nunca compareceram às aulas ou que compareceram apenas à primeira semana de aula, na qual costuma-se apenas apresentar a disciplina e realizar cadastro dos estudantes nas plataformas online. Dificilmente, um aluno que apenas frequentou este período de tempo evadiu-se por não se identificar com a metodologia aplicada na disciplina, e logo podemos considerar que a situação de cada um destes está fora da influência do tipo de aula ou do conteúdo. Desta forma, foram levados em conta nas análises, apenas os estudantes que frequentaram pelo menos as duas primeiras semanas de aula, pois na segunda semana a metodologia aplicada já é posta em ação.

Para cada uma das situações que serão apresentadas nas subseções a seguir, foram calculados e plotados os gráficos de caixa (*boxplots*) das proporções de alunos reprovados por nota e por evasão, visto que os números absolutos costumam diferir entre semestres letivos e também entre turnos de um mesmo semestre. Essas diferenças ocorrem, em sua maioria, por conta da quantidade de matrículas que realmente são efetivadas. As turmas noturnas, por exemplo, costumam ter um menor número de matrículas concluídas em relação às vespertinas.

5.1 Reprovações por nota

Nesta seção comparamos as reprovações por nota entre os semestres letivos em que foram utilizadas os tipos de aula expositiva e àqueles em que as Metodologias Ativas foram postas em ação. Assim, os gráficos em verde foram gerados pelos dados dos semestres de 2014.2 a 2017.2, excluindo-se as turmas extras.

Observando os boxplots apresentados na Figura 4, é possível notar a diferença na proporção de reprovados por nota. Nas turmas da tarde onde as aulas seguiam o modelo de aulas expositivas, a mediana de reprovações por nota foi de pouco mais 27% da turma, com picos onde mais da metade (até 57%) dos estudantes não obteve nota suficiente para ser aprovado na disciplina. Nas turmas noturnas a mediana ficou em 20% com pico de 46% de reprovação. Já nas turmas onde foram aplicadas metodologias ativas, em ambos os turnos, a mediana ficou em cerca de 20%, tendo pico de cerca de 25% na turma noturna e um *outlier* de 38% no turno vespertino.

5.2 Reprovações por falta

O impacto nas evasões também foi significativo. A Figura 5 mostra os boxplots referentes à quantidade de alunos reprovados por falta nas turmas onde houve aplicação da metodologia tradicional em contraste às que houve aplicação de metodologias ativas.

A proporção de estudantes que abandonou a disciplina no turno vespertino apresentou leve diminuição na mediana, passando de 20% para cerca de 17%. Porém, nas turmas noturnas a diferença é bastante significativa, onde houve

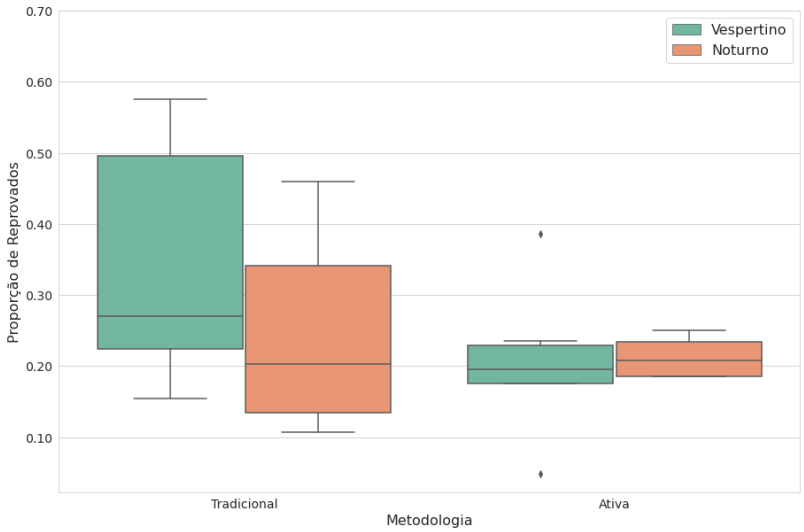

Figure 4: Boxplots das proporções de reprovados por nota nas turmas que utilizaram metodologia tradicional *versus* metodologias ativas

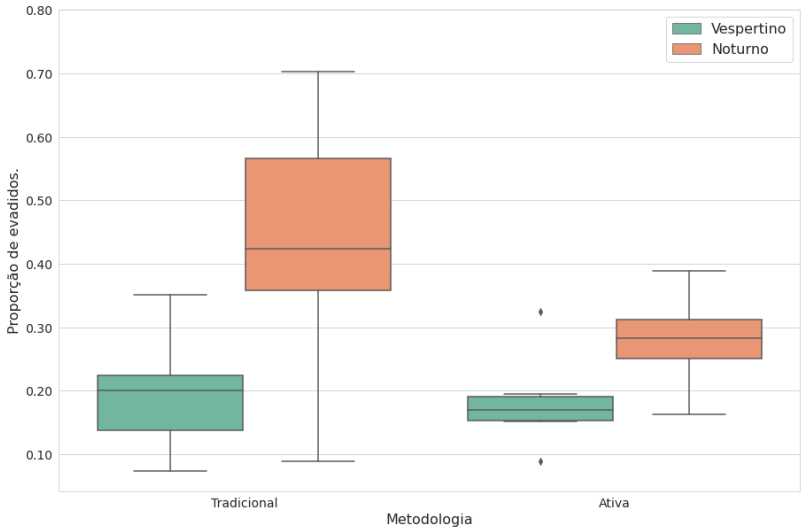

Figure 5: Boxplots das proporções de reprovados por falta nas turmas que utilizaram metodologia tradicional *versus* metodologias ativas

queda da mediana que passou de cerca de 42% para cerca de 28%, também passando de máximo de 70% para um máximo de menos de 40% de reprovações por falta.

6. CONCLUSÃO

As abordagens ativas de ensino têm se mostrado alternativas que podem ajudar a tornar o processo de aprendizagem mais significativo para os estudantes. Apesar de suas vantagens, nem sempre é tarefa fácil aplicar esse tipo de metodologia em sala de aula, principalmente, quando os alunos desconhecem esse formato de aula. Em algumas turmas onde as MAs foram aplicadas, era possível notar o estranhamento inicial por parte dos alunos. A Sala de Aula Invertida, por exemplo, sofre de uma pequena resistência e confusão. Nem todos os alunos entendem a importância do estudo individual de forma adiantada em relação à posterior aula presencial.

Do ponto de vista comportamental, observou-se forte integração dos alunos que trabalharam com ABP e com o ABE combinado à Sala de Aula Invertida, característica não observada em turmas anteriores. Enquanto até 2017.2 constatava-se uma falta de motivação por parte dos estudantes, inclusive para participar de eventos e festividades promovidos pela instituição, de 2018.1 em diante (excetuando-se no período de aulas totalmente remotas), os alunos passaram a participar mais ativamente das palestras, eventos científicos, *workshops* e festividades que eram promovidos pelo campus.

7. REFERÊNCIAS

[1] M. A. Albanese and S. Mitchell. Problem-based

learning: A review of literature on its outcomes and implementation issues. *Academic medicine*, 68(1):52–81, 1993.

[2] L. Bacich and J. Moran. *Metodologias ativas para uma educação inovadora: uma abordagem teórico-prática*. Penso Editora, 2017.

[3] A. Baist and A. S. Pamungkas. Analysis of student difficulties in computer programming. *VOLT: Jurnal Ilmiah Pendidikan Teknik Elektro*, 2(2):81–92, 2017.

[4] D. Berrett. How 'flipping'the classroom can improve the traditional lecture. *The chronicle of higher education*, 12(19):1–3, 2012.

[5] J. H. Berssanette and A. C. de Francisco. Active learning in the context of the teaching/learning of computer programming: A systematic review. *Journal of Information Technology Education. Research*, 20:201, 2021.

[6] J. Bishop and M. A. Verleger. The flipped classroom: A survey of the research. In *2013 ASEE Annual Conference & Exposition*, number 10.18260/1-2–22585, Atlanta, Georgia, June 2013. ASEE Conferences. https://peer.asee.org/22585.

[7] M. A. F. Borges. Avaliação de uma metodologia alternativa para a aprendizagem de programação. *Workshop de Educação em Computação - Congresso anual da SBC*, 2000.

[8] I. Calderon, W. Silva, and E. Feitosa. Um mapeamento sistemático da literatura sobre o uso de metodologias ativas durante o ensino de programação no brasil. In *Anais do XXXII Simpósio Brasileiro de Informática na Educação*, pages 1152–1161, Porto Alegre, RS, Brasil, 2021. SBC.

[9] F. E. S. Coelho. Primeiros passos na aprendizagem baseada em problemas. *Congresso Regional sobre Tecnologias na Educação (Ctrl+e)*, 2016.

[10] M. C. Coelho and A. M. A. Guedes. Aprendizagem baseada em problemas aplicada à programação de computadores: Um mapeamento sistemático. *RENOTE*, 18(2):570–580, jan. 2021.

[11] P. A. Cohen, B. J. Ebeling, and J. A. Kulik. A meta-analysis of outcome studies of visual-based instruction. *ECTJ*, 29(1):26–36, 1981.

[12] A. M. C. de A. Oliveira, R. L. Rodrigues, and V. C. Garcia. Um mapeamento sistemático para problem based learning aplicado à ciência da computação. *Anais do Workshop de Informática na Escola - WIE 2012- XXXII Congresso da Sociedade Brasileira de Computação*, 2012.

[13] S. Freeman, S. L. Eddy, M. McDonough, M. K. Smith, N. Okoroafor, H. Jordt, and M. P. Wenderoth. Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences*, 111(23):8410–8415, 2014.

[14] L. M. M. Giraffa and M. C. Móra. Evasão na disciplina de algoritmo e programação: Um estudo a partir dos fatores intervenientes na perspectiva do aluno. *3ª Conferência LatinoAmericana Sobrre el Abandono en La Educación - III CLABES*, 2013.

[15] M. S. Gomes, E. M. H. do Amaral, L. Becker, C. Huegel, and L. Gestaro. Uma ferramenta para o auxílio ao professor na identificação de dificuldades na prática de programação. *Anais do Salão Internacional de Ensino, Pesquisa e Extensão*, 7(2), 2015.

[16] R. Kadar, N. A. Wahab, J. Othman, M. Shamsuddin, and S. B. Mahlan. A study of difficulties in teaching and learning programming: a systematic literature review. *International Journal of Academic Research in Progressive Education and Development*, 10(3):591–605, 2021.

[17] K. Kotovsky. *Problem Solving-large/small, hard/easy, conscious/nonconcious, problem-space/problem-solver: The issue of dichotomization*, page 374 – 384. Cambridge University Press, UK, 2003.

[18] B. J. McNeil and K. R. Nelson. Meta-analysis of interactive video instruction: A 10 year review of achievement effects. *Journal of Computer-Based Instruction*, 1991.

[19] MEC - Ministério Da Educação, SEED - Secretaria de Educação A Distância, and PROINFO - Programa Nacional de Informática Na Educação. *CARTILHA: Recomendações para a Montagem de Laboratório de Informática nas Escolas. 12 p.* Brasília/DF, 2005. Acesso em: 23 de fevereiro de 2019.

[20] L. K. Michaelsen and M. Sweet. Fundamental principles and practices of team-based learning. *Team-based learning for health professions education: A guide to using small groups for improving learning*, pages 9–34, 2012.

[21] J. E. Moström. *A study of student problems in learning to program*. PhD thesis, Umeå University, Department of Computing Science, 2011.

[22] A. Mourão. Uma proposta da eficiência do uso da metodologia ativa baseada em problemas, utilizando dojo de programação, aplicada na disciplina de lógica de programação. *Anais do Workshop de Informática na Escola - WIE 2017- XXXVII Congresso da Sociedade Brasileira de Computação*, 2017.

[23] E. P. Pimentel and N. Omar. Ensino de algoritmos baseado na aprendizagem significativa utilizando o ambiente de avaliação netedu. *Anais do XXVIII Congresso da SBC - WEI – Workshop sobre Educação em Computação*, 2008.

[24] A. Rabelo, L. C. G. Maia, and F. S. Parreiras. Performance analysis of computer science students in programming learning. *26º WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI 2018)*, 26(1/2018), 2018.

[25] C. E. Rapkiewicz. O processo de ensino-aprendizagem de fundamentos de programação: uma visão crítica da pesquisa no brasil.

[26] I. C. Ribeiro, W. Silva, and E. Feitosa. Repositório colaborativo para apoiar a adoção de metodologias ativas no ensino de programação. In *Anais Estendidos do I Simpósio Brasileiro de Educação em Computação*, pages 56–57, Porto Alegre, RS, Brasil, 2021. SBC.

[27] A. Saad and S. Zainudin. A review of project-based learning (pbl) and computational thinking (ct) in teaching and learning. *Learning and Motivation*, 78:101802, 2022.

[28] A. Santos, A. Gorgônio, A. Lucena, and F. Gorgônio. A importância do fator motivacional no processo ensino-aprendizagem de algoritmos e lógica de programação para alunos repetentes. In *Anais*

do XXIII Workshop sobre Educação em Computação, pages 168–177, Porto Alegre, RS, Brasil, 2015. SBC.

[29] J. R. Savery. Overview of problem-based learning: Definitions and distinctions. In *Interdisciplinary Journal of Problem-Based Learning*, volume 1, pages 9–20, 2006.

[30] K. Yongmin. The effects of pbl-based data science education classes using app inventor on elementary student computational thinking and creativity improvement. *Journal of The Korean Association of Information Education*, 24:551–562, 12 2020.

[31] D. Zhang, L. Zhou, R. O. Briggs, and J. F. Nunamaker Jr. Instructional video in e-learning: Assessing the impact of interactive video on learning effectiveness. *Information & management*, 43(1):15–27, 2006.