UM EXPERIMENTO DO USO DE *CODING DOJO* NA APRENDIZAGEM DE PROGRAMAÇÃO ORIENTADA A OBJETOS

Pablo Schoeffel[1,2], Daniel Felipe da Rosa[1], Raul Sidnei Waslawick[2]

[1] Departamento de Engenharia de Software – Universidade do estado de Santa Catarina (UDESC)
2 Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
pabloschoeffel@gmail.com, hellowdan@gmail.com, raul@inf.ufsc.br

Resumo
Aumentar o interesse e aprendizado dos alunos nas disciplinas de programação e capacitar equipes de desenvolvimento de *software* são objetivos que tem motivado a academia e a indústria a encontrar alternativas mais eficientes para a fixação e aplicação de conceitos. Este trabalho apresenta a técnica de *Dojos* de programação ou *coding dojo* (em inglês), que são encontros onde programadores reúnem-se para praticar programação, como uma alternativa para alcançar este objetivo, permitindo o compartilhamento de conhecimento e aprendizado. Além de não existirem muitos trabalhos científicos que abordam essa técnica, esse trabalho contribui ao encontrar evidências quantitativas que o uso de *dojo* de programação é mais eficaz que a técnica tradicional de aulas expositivas. O trabalho relata um experimento que compara o desempenho de um grupo de controle, o qual foi exposto a uma abordagem mais tradicional de aula expositiva, com um grupo que foi exposto ao *Dojo* de programação. Foram aplicados pré e pós-testes para 30 alunos de um curso técnico em Eletrotécnica e Informática, no conteúdo de programação orientada a objetos. A partir das avaliações e tratamento estatístico das notas, percebeu-se que a média do grupo que participou do *Dojo* de programação foram significativamente maiores do que a média do grupo de controle, principalmente nas avaliações práticas.

Palavras-chave: Prática Deliberada. *Dojo* de programação. Ensino-aprendizagem de Programação de computadores.

Abstract
Academic and industrial studies intending to find new ways to improve teaching of programming skills have been motivated by the need to increase students interest in learning programming and improving development teams. This paper assesses the technique known as "coding dojo", which consists of meetings of programmers with the aim of sharing knowledge and learning by collectively producing programming code. This technique has been poorly addressed in literature, and this paper presents a contribution for finding evidences that it is more effective than usual classes based on speeches. The paper reports an experiment that compares the performance of a group that was exposed to the dojo technique and a control group exposed to usual speech-based classes. Pre and post tests were applied to 30 students from a class of technicians in electronics and informatics. The subject of the class was object-oriented programming. The assessment and statistical treatment of the grades revealed that the average grade of the group that has participated at the dojo activities was significantly greater than the average of the control group, especially when regarding evaluation of practice.

Keywords: *Deliberate Practice. Coding Dojo. Computer Programming teaching-leaning.*

1. Introdução

John Locke (1632-1704) demonstrou experimentalmente que, no conhecimento, não existe nada de inato e tudo é aprendido com a experiência. "Para ilustrar essa teoria, Locke recorre a uma

metáfora que se tornou célebre: a mente humana é, ao nascer, um papel em branco sobre o qual a prática do mundo externo e a reflexão individual imprimirão aqueles sinais denominados conhecimento" (ÁLVARES e BATISTA, 2007, p. 2).

> De onde se apreende todos os materiais da razão e do conhecimento? A isso respondo, numa palavra: da experiência. Todo o nosso conhecimento está nela fundado, e dela deriva fundamentalmente o próprio conhecimento. Dessas duas fontes de conhecimento jorram todas as nossas ideias ou as que possivelmente teremos (LOCKE, 1997, p. 57).

Por meio dos pensamentos de John Locke (1632-1704), em sua obra "Ensaio Sobre o Entendimento Humano", é possível observar que o conhecimento é adquirido através do exemplo e da experimentação, como ocorre no desenvolvimento das crianças, que repetem os comportamentos observados ao seu redor e assim constroem o seu conhecimento sobre o mundo, progressivamente (LOCKE, 1997).

Mas somente a experiência talvez não seja suficiente, pois você deve aprender coisas novas, experimentar, pois muitas vezes a prática foca nas coisas que já se sabe como fazer. De acordo com Ericsson, Krampe e Tesch-Romer (1993), a prática deliberada é diferente: ela implica esforços consideráveis e específicos para fazer algo que não se consegue fazer bem - ou nem mesmo se sabe como fazer. Pesquisas mostram que somente trabalhando no que você não consegue fazer, fará com que seja possível tornar-se no especialista que se deseja ser (ERICSSON; KRAMPE; TESCH-ROMER, 1993). Ericsson, Krampe e Tesch-Romer (1993) estudaram os aspectos que influenciavam a aquisição de *expertise* em diferentes domínios, como xadrez, música e esportes, e descobriram que a prática deliberada por um longo período de tempo faz parte da estratégia dos especialistas em seus processos de aprendizagem.

Conforme Relvas (2016), a prática deliberada consiste na realização de tarefas para além do nível atual de competência e conforto, a um nível que seja realista fazê-las com algumas horas de prática. Nesse processo, os profissionais vão gradualmente refinando o seu desempenho com a repetição e *feedback*, como ocorre com músicos e atletas profissionais. Considerando que médicos exercitam suas técnicas em cadáveres, bonecos e fazem suturas em laranjas, Bache (2013) faz a seguinte pergunta: o que os programadores fazem para treinar? A própria autora responde: escrevem código no trabalho, muito código e, quando vão para casa, escrevem mais código pelo prazer de escrever código. Músicos treinam para suas apresentações, mas também praticam escalas. Atletas profissionais treinam para os jogos, mas fazem exercícios específicos para melhorar aquelas habilidades difíceis de adquirir e fáceis de perder. Há formas específicas dos programadores treinarem estas habilidades, que incluem trabalho em equipe, desenvolvimento guiado a testes, refatoração e boas práticas em projeto de *software* (BACHE, 2013).

O ensino e aprendizagem de programação de computadores é mencionado como um dos grandes desafios na área de Educação em Computação (MCGETTICK et al., 2005). Estudos recentes mostram que, apesar dos avanços na área de pedagogia, as taxas mundiais de falhas em disciplinas e cursos introdutórios de computação não têm melhorado (WATSON; LI, 2014), sendo que as taxas mundiais relatadas são 19% a 40% de evasão em cursos de computação e tecnologia (MOONEY et al., 2010; DREW, 2011; HÜSING, 2013). No Brasil, estudos mostram uma taxa média de evasão igual ou superior a 40% (PALMEIRA; SANTOS, 2014; BORBA, 2015; VERONESE e LEMOS, 2015; FERREIRA, 2015; RODRIGUES, 2013).

Um dos motivos para esses altos índices é a dificuldade dos alunos com programação de computadores (NIITSOO et al., 2014). Esse alto índice de evasão e reprovação em programação tem tido a atenção de pesquisadores a fim de investigar suas causas e soluções (TAN; TING; LING, 2009).

Dentre os alunos, aqueles novatos em programação sofrem ainda mais dificuldades. Para resolver isso, teorias novas sugerem o foco não no ensino, mas sim na aprendizagem, tirando o foco do instrutor e passando para o estudante, estimulando o interesse e envolvendo ativamente os estudantes com o curso (ROBINS; ROUNTREE; ROUNTREE, 2003).

É neste contexto que se encontra o aprendizado através da prática deliberada com *Dojos* de programação ou *coding dojo* (em inglês), que são encontros onde grupos de programadores se reúnem para aprender, praticar e trocar experiências (SATO; CORBUCCI; BRAVO, 2008). Numa sessão de *Coding Dojo*, os participantes escolhem um desafio de programação como uma forma de praticar, sem necessariamente resolver o desafio (LUZ; NETO, 2013).

Por meio da aplicação de *Dojos* de programação, este trabalho tem por objetivo apresentar evidências do melhor desempenho da técnica supracitada em relação ao método tradicional de ensino com aulas expositivas. Para isto, foi realizado um experimento com alunos de programação de um curso técnico, no qual foi realizado um encontro de *Dojo* de programação e uma aula expositiva com dois grupos de alunos. Após as aulas e realização de avaliação, foram obtidas as notas que serviram de insumo para um trabalho estatístico de comparação de desempenho entre os grupos. A hipótese verificada no experimento foi se que alunos que são expostos ao tema "Programação Orientada a Objetos" através de *Dojos* de programação conseguem atingir, na média, melhor desempenho que alunos que são expostos ao mesmo tema em aulas expositivas.

O principal fator motivador para a realização deste trabalho é a aplicabilidade real em ambientes acadêmicos e corporativos. É notória a demanda por alternativas que permitam capacitação mais rápida, atrativa e eficiente do que a abordagem puramente teórica dos métodos tradicionais de ensino (BOTOVA, 2012). Outra justificativa para o trabalho são os altos índices de evasão em cursos de computação (MOONEY et al., 2010; DREW, 2011; HÜSING, 2013; FERREIRA, 2015), sendo que as causas muitas vezes estão relacionadas às dificuldades dos alunos em aprender programação.

Embora o assunto de *Dojos* de programação não seja um tema muito explorado na literatura ainda, foram encontrados trabalhos que relatam estudos de caso do uso dessa técnica para o processo de ensino-aprendizagem. Porém, a maioria dos trabalhos encontrados avalia a percepção do aluno a respeito da prática, sem avaliar a aprendizagem de forma quantitativa e sem avaliar estatisticamente a validade dos dados (LUZ; NETO; NORONHA, 2013; FONTES, 2011; CARMO; BRAGANHOLO, 2012; BRAVO; GOLDMAN, 2010).

O trabalho de Rocha, Sabino e Esan (2014) descreve a utilização de *Dojos* de programação durante todo semestre na disciplina de lógica de programação, porém não compara o desempenho frente a outras abordagens metodológicas. Já o trabalho de Estácio et al. (2015a) descreve um experimento, porém comparando *Dojos* de programação com programação em pares e solo, no contexto da criação de projetos de tela (*mockups*) e não programação. Em outro experimento similar, Estácio et al. (2015b) também compara *Dojos* de programação com programação em pares e programação solo, no contexto de programação, fazendo uma avaliação do desempenho por meio de formulários de auto avaliação e análise de anomalias presentes no código de três exercícios aplicados.

Como diferenciais aos trabalhos correlatos, são contribuições desse trabalho: i) a realização de um experimento controlado e com tratamento estatístico dos resultados; ii) a aplicação para o contexto de programação orientada a objetos; iii) a aplicação com alunos de um curso técnico; iv) a comparação do método com uma abordagem tradicional de ensino (aulas expositivas); v) o planejamento estruturado das unidades instrucionais, com disponibilização de planos, avaliações e materiais didáticos. Com isso, o trabalho busca contribuir com novos estudos, permitindo a replicação e comparação dos resultados do experimento e servir como um embasamento do indicativo da eficácia no uso de Dojos de Programação, frente à abordagem tradicional de aulas expositivas.

Este artigo está organizado em seções que abordam, inicialmente, os conceitos estudados durante a pesquisa bibliográfica sobre *Dojos* de programação (Seção 2) e contribuições de trabalhos similares já realizados nesta área (Seção 3). Em seguida, é apresentada a metodologia utilizada para a elaboração e validação do experimento (Seção 4) e o relato da aplicação (Seção 5). Por fim, são apresentados os resultados do trabalho (Seção 6) e considerações finais (Seção 7).

2. *Dojos* de Programação

A palavra *Dojo* tem origem oriental. Traduzido literalmente do idioma japonês, significa "lugar do caminho", sendo utilizado originalmente como espaço de meditação para monges budistas. Pode ser interpretado, portanto, como "lugar onde se estuda a vida", utilizado para a prática de atividades físicas e espirituais. No ocidente, este termo tem sido comumente utilizado para o treino de artes marciais (LUZ; NETO; NORONHA, 2013).

Inspirado neste tema, David Thomas idealizou o primeiro *Dojo* de programação em 2003, em Paris, chegando no Brasil em 2007, com Ivan Sanchez, a partir da criação do grupo *Dojo Floripa* (MATSUDA; SANTOS; RODRIGUES, 2013).

De acordo com Bache (2013), *Dojo* de programação possui um ambiente descontraído, longe de gerentes, prazos e *bugs*, permitindo que as pessoas superem a timidez para demonstrar quão bem produzem códigos, assim como dar dicas e conselhos aos demais. Segundo Heinonen et al. (2013), *coding dojo* é uma forma de aprendizagem que valoriza a experiência concreta num contexto real. Segundo Sato, Corbucci e Bravo (2008), os princípios dos *Dojos* de programação são: um ambiente colaborativo, inclusivo e não competitivo, além de ser um lugar onde as pessoas podem aprender continuamente.

Sato, Corbucci e Bravo (2008) sugerem quatro passos para o planejamento e execução de um *Dojo* de programação, porém com as primeiras etapas focando mais no problema em si: escolha do problema, discussão do problema, seção de codificação e retrospectiva. Já Bache (2013) define que, para organizar um *Dojo* de programação, planejar antecipadamente o encontro está ligada ao atingimento dos objetivos desejados. Ao planejar um *Dojo*, é necessário pensar sobre como conduzir os seguintes elementos: apresentação, acordo entre os participantes, codificação e retrospectiva (BACHE, 2013):

a) Apresentação: a apresentação do *Dojo* deve ajudar os participantes a sentirem-se seguros, principalmente os novatos. É importante mencionar os princípios do *Dojo* e estimular o respeito mútuo entre as pessoas;
b) Acordo entre os participantes: antes da atividade prática, é interessante ter um consenso sobre o que o grupo deseja focar e conversar um pouco sobre o que desejam aprender. É uma boa oportunidade para aplicar lições aprendidas de encontros anteriores;
c) Codificação: a parte principal do encontro é a codificação, na qual os participantes trabalham na tentativa conjunta de construir uma solução para um desafio de programação proposto. É importante encorajar discussões, questionamentos e sugestões. Nesta etapa as pessoas que estiverem produzindo código devem explicar o seu raciocínio aos outros participantes, sem exibicionismo ou competição;
d) Retrospectiva: após finalizar a codificação, ocorre a retrospectiva, que é o momento onde todos refletem sobre o que foi aprendido na atividade, estimulando os participantes a compartilharem suas impressões sobre o encontro, sejam elas positivas ou negativas.

O último momento do encontro é a confraternização. Esta atividade possibilita uma maior integração entre as pessoas, que aproveitam este tempo para conversar e discutir sobre as horas de programação e aprendizado que vivenciaram.

Existem pelo menos três formas de realizar um encontro de *Dojo* de programação, e cada uma pode ser mais indicada para determinado cenário ou objetivo a se alcançar: *Kata*, *Randori* e *Kake* (MATSUDA; SANTOS; RODRIGUES, 2013)

No Dojo Kata, o apresentador prepara o roteiro do código a ser apresentado e a plateia vê o conteúdo mostrado em um projetor. Durante a apresentação, o apresentador deve explicar o passo-a-passo executado e a plateia pode fazer perguntas a qualquer momento (MATSUDA; SANTOS; RODRIGUES, 2013). A Figura 1 ilustra a dinâmica das interações do *Dojo Kata*.

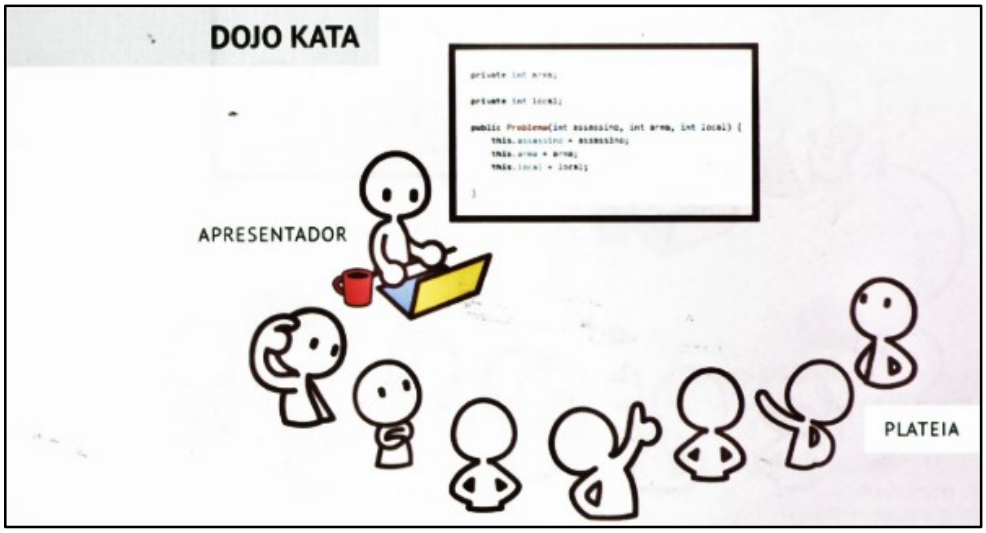


Figura 1: dinâmica do *Dojo Kata*
Fonte: MATSUDA; SANTOS; RODRIGUES, 2013, p. 9

Outra variação é o Dojo Kake, sendo que nessa configuração toda a plateia é dividida em pares, sendo que uma pessoa comanda o teclado (piloto) e outra ajuda nas questões técnicas e lógicas (copiloto). Então se escolhe um único problema que todos resolverão. Cada par é livre para escolher a linguagem de programação e a estratégia de solução. A cada intervalo, o piloto de cada par assume o posto de copiloto em outro par e o copiloto permanece, agora como piloto. O ciclo se repete até que todos rotacionem entre os pares (MATSUDA; SANTOS; RODRIGUES, 2013). A Figura 2 ilustra a dinâmica das interações do *Dojo Kake*.

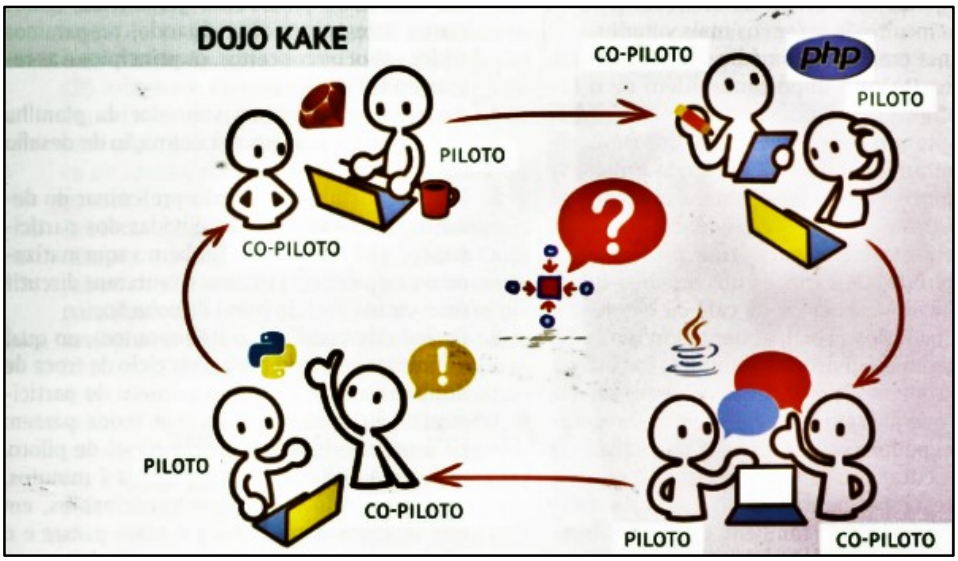


Figura 2: dinâmica do *Dojo Kake*
Fonte: MATSUDA; SANTOS; RODRIGUES, 2013, p. 11

No *Dojo Randori*, todas as pessoas participam da criação de um único código que é apresentado como um desafio de programação ao grupo. Um par escolhido inicia o desenvolvimento da solução, sendo um piloto e um copiloto. "Após um intervalo, normalmente de 5 a 7 minutos, o piloto deixa o computador e se junta à plateia, o copiloto se torna o novo piloto e alguém da plateia assume o posto de copiloto. Esse ciclo se repete até que todos tenha sido piloto" (MATSUDA; SANTOS; RODRIGUES, 2013, p. 10). A Figura 3 ilustra a dinâmica das interações do *Dojo Randori*.

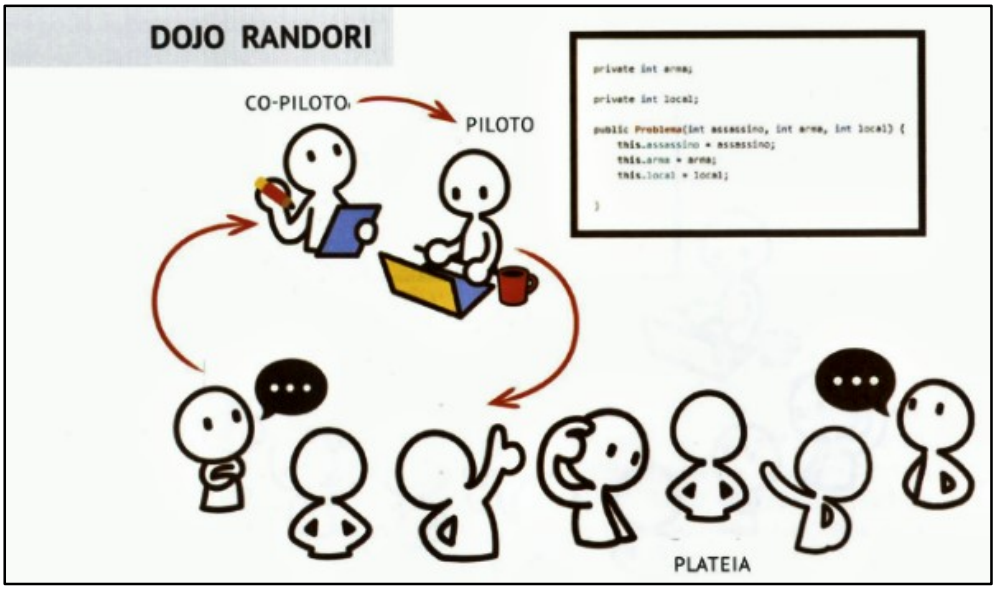


Figura 3: dinâmica do *Dojo Randori*
Fonte: MATSUDA; SANTOS e RODRIGUES, 2013, p. 10

Com relação à quantidade de participantes de um *Dojo*, Bache (2013) afirma que os melhores resultados foram obtidos com grupos de 5 a 15 pessoas. "Um grupo muito pequeno perde variedade de opiniões e estilos. Um grupo muito grande torna as discussões incontroláveis e algumas pessoas podem se retrair" (BACHE, 2013, p. 57).

De acordo com Bache (2013), para a realização de um *Dojo*, é importante atentar para alguns itens de infraestrutura. O principal deles é uma sala com uma mesa, cadeiras e espaço suficiente para as pessoas levantarem e interagirem com os colegas. Também é essencial computador para a prática de programação, conectado a um projetor, para que todos possam visualizar o código que está sendo criado no computador. Para as discussões, é recomendável um quadro branco.

Quando a atividade de programação inicia, faz-se importante a presença do *Sensei*, ou facilitador. Conforme Bache (2013), esta função consiste em criar uma boa atmosfera, levantar discussões e manter a evolução do código. Requer habilidades sociais para manter o clima de colaboração, de forma que os participantes se sintam seguros para aprender. É importante também não interromper um raciocínio, e, diante de uma falha de programação, estimular o aprendizado com perguntas e intervenções sutis. "Deve-se lembrar que o objetivo não é a finalização do desafio de programação, mas sim o processo de aprendizado" (BACHE, 2013, p. 64).

3. Trabalhos Correlatos

O uso de *Dojos* tem sido adotado por alguns professores como alternativa para o ensino de programação, principalmente quando relacionado a métodos ágeis. Um conjunto de trabalhos conduzidos por Ramiro Batista da Luz (LUZ, 2013; LUZ; NETO, 2013; LUZ; NETO; NORONHA, 2013) apresenta um estudo sobre a percepção e aceitação da técnica de *Dojos* de programação para o aprendizado de práticas ágeis, avaliando por meio de formulários de pesquisa de satisfação e entrevistas com participantes, além de entrevistas com especialistas. Como resultado, os autores relatam que o ambiente do *Dojo* de programação foi favorável para a integração e inclusão de programadores no ambiente de aprendizado, sendo que deve ser intercalado com aulas expositivas para a apresentação de conceitos, e as entrevistas com especialistas resultaram num conjunto de vantagens e desvantagens do uso de *Dojos*. Porém, os trabalhos não especificam qual o contexto do experimento (nível de ensino, conteúdo e tipo de *Dojo*), tão pouco utiliza testes estatísticos para validação dos resultados.

A monografia de Bernardo Botelho Fontes (FONTES, 2011) descreve uma pesquisa sobre a experiência de 174 pessoas com *Dojos* de Programação e conclui que a prática de *Dojos* de programação pode potencializar o aprendizado de programação nas universidades. Porém, afirma que substituir o método tradicional de ensino por *Dojos* de programação é uma ilusão, pois a mudança no sistema de ensino não deve ser encarada com uma abordagem simplista, ou seja, envolve diversos aspectos que devem ser avaliados e pode envolver um conjunto de métodos. O trabalho também traz o relato de uso de Dojos de programação na Universidade Federal Fluminense, mas não quantifica resultados estatisticamente.

No trabalho "Um estudo sobre o uso didático de *DOJOs* de programação", dos autores Daniel H. Carmo e Vanessa Braganholo (CARMO; BRAGANHOLO, 2012), foi introduzida a técnica de Dojos numa turma de 25 alunos na disciplina de Estrutura de Dados num curso de graduação, sendo utilizado TDD (*Test Driven Development*) para apoiar a atividade. Foram testadas técnicas de *Dojos* em grupos e individuais e avaliado, por meio de questionários pós-teste, o nível de satisfação dos participantes. Constatou-se que houve bastante resistência em relação à utilização dos *Dojos*, principalmente por aspectos motivacionais e pela timidez, pois todos foram obrigados a participar e nem todos ficaram à vontade com o nível de exposição. Também se constatou que o perfil dos participantes influencia neste resultado, pois o nível de aceitação variou entre "totalmente contra *Dojos*" e "fãs da abordagem", o que leva à conclusão de que a participação por iniciativa própria pode ser um fator essencial para o sucesso da utilização de *Dojos* de programação como metodologia de ensino. Não foram demonstrados tratamentos estatísticos dos resultados.

Um experimento de longa duração foi realizado por Fábio Rocha, Rosimeri Sabino e Josephine Esan, no trabalho "O uso do *Dojo* na prática pedagógica do ensino de lógica de programação" (ROCHA; SABINO; ESAN, 2014). Por meio de uma investigação com 54 alunos de um curso de lógica de programação de um curso técnico, com carga horária de 140 horas e utilizando a ferramenta *Scratch* (MIT MEDIA LAB, 2016) nas práticas dos *Dojos* de programação, verificou-se que os resultados foram significativos. O índice de evasão foi nulo, com apenas 6% de falta, sendo o histórico anterior de 23%. Os alunos alegaram que a motivação os fez evitar faltar às aulas. Constatou-se, também que, após a adoção de *Dojos* de programação, as notas dos alunos melhoraram consideravelmente, indo de 6 a 7 em turmas anteriores para 7,5 a 8 nas turmas onde foram aplicados *Dojos*. Os autores concluem que a adoção de Dojo como prática pedagógica é viável, porém, apesar de avaliarem a aprendizagem por meio das notas dos estudantes, não demonstram validação estatística desses resultados.

O trabalho de Estácio et al. (2015a) descreve um experimento com uma turma de 14 alunos de graduação, comparando as técnicas de programação solo, Dojo Randori e programação em pares (PP) no contexto da criação de projetos de tela (*mockups*). Os resultados, a partir de questionários pós-teste com participantes, indicam que benefícios, com relação à motivação e aprendizagem, no uso de abordagens colaborativas (programação em pares e Dojos).

Já Estácio et al (2015b) fizeram um experimento similar com uma turma de 17 alunos de graduação, no contexto de programação ágil, comparando programação solo, em pares e *Dojo* de programação estilo Randori (DPR) na realização de três exercícios práticos. Por meio de formulários de auto avaliação e da verificação número de anomalias de código para cada grupo, o estudo conclui PP e Dojo de programação tiveram desempenho maior do que programação solo tanto na aprendizagem quanto na motivação. Comparando PP com DPR percebeu-se que obtiveram um desempenho similar na aprendizagem, porém os alunos reportaram uma maior motivação no uso de PP.

Heinonen et al. (2013) relata a aplicação de *Coding Dojos* na disciplina de Engenharia de Software num curso de graduação, com 73 alunos. Foram aplicados questionários pós-testes para avaliar a percepção dos participantes quanto à atmosfera, aprendizagem e participação. Os

resultados descrevem uma satisfação regular dos alunos, embora o trabalho não faça análise estatística dos resultados.

O Quadro 1 mostra um comparativo entre os principais trabalhos correlatos encontrados, incluindo o proposto nesse trabalho.

Quadro 1 – Comparativo de trabalhos correlatos

Trabalho	Tipo de Pesquisa	Público-alvo	Avaliação	Trat. Estat.
Luz (2013)	Percepção e aceitação da técnica de Dojo	64 Alunos de graduação	Formulários e entrevista	Não
Fontes (2011)	Percepção e aceitação da técnica de Dojo	174 Estudantes de graduação e profissionais	Formulário	Não
Carmo e Braganholo (2012)	Percepção e aceitação da técnica de Dojos individuais e em grupo	26 Alunos de graduação da disciplina de Estrutura de Dados II	Formulário	Não
Rocha, Sabino e Esan (2014)	Avaliação da aplicação de dojos durante um semestre em duas turmas.	54 Alunos da disciplina de lógica de programação de curso técnico	Observatória / descritiva	Não
Estácio et al. (2015a)	Comparação do uso de Programação Solo, em Pares e Dojos na aprendizagem e motivação no contexto de criação de projetos de tela (*mockups*)	14 Alunos de graduação	Questionário pós-teste, verificando a percepção do aluno quanto à prática e aprendizagem	Não
Estácio et al. (2015b)	Comparação do uso de Programação Solo, em Pares e Dojos na aprendizagem de programação	17 Alunos de graduação	Avaliação prática pós-teste, verificando qualidade do código gerado (número de anomalias) e questionário verificando a percepção do aluno quanto à prática e aprendizagem	Sim
Heinonen (2013)	Avaliação da aprendizagem e percepção da participação em sessões de dojos de programação	50 alunos de graduação	Formulário pós-teste	Não
Este trabalho	Comparação no desempenho entre uso de dojos e aulas expositivas no aprendizado de programação orientada a objetos	30 alunos de curso técnico	Testes teóricos pré e pós testes, além de testes práticos pós teste com grupo de controle e tratamento.	Sim

Como diferenciais aos trabalhos correlatos, esse trabalho realiza comparação, com tratamento estatístico, entre os resultados atingidos em um *Dojo* de programação e em uma aula

expositiva, realizando um experimento controlado com grupo de controle, num contexto diferenciado, de um curso técnico. O trabalho que mais se assemelha a esse estudo é o de Estácio et al. (2015b), pois faz uma avaliação de aprendizagem, porém com diferenciais de que naquele foram realizadas somente avaliações pós-teste, a comparação é entre programação solo, em pares e *Dojo*, além da aprendizagem ter sido avaliada baseada na percepção de aprendizado dos alunos e na análise de anomalias inseridas no código em três exercícios práticos. Além disso, diferentemente do estudo proposto nesse artigo, o trabalho de Estácio et al. (2015b) não descreve a abordagem de ensino-aprendizagem utilizada e também não avalia o conhecimento teórico dos estudantes. O trabalho de Heinonen (2013) também avalia a aprendizagem, porém sob o ponto de vista da percepção do aluno (auto avaliação). Todos os demais trabalhos encontrados a percepção e aceitação da técnica, mas não a evolução na aprendizagem.

4. Metodologia

A pesquisa constitui um experimento, utilizando uma abordagem quantitativa, comparando o desempenho, em avaliações pré e pós-teste, entre um grupo de controle e um grupo de tratamento, divididos aleatoriamente. Para o grupo de tratamento, foi utilizado o *Dojo* de programação do estilo *Randori* como metodologia de ensino e, para o grupo de controle, aulas expositivas.

O experimento foi realizado com 30 alunos do curso técnico em Eletrotécnica e Informática do Serviço Nacional de Aprendizagem Industrial de Santa Catarina (SENAI), unidade de Timbó-SC, sendo abordado o conteúdo de programação orientada a objetos, o qual foi exposto em uma unidade instrucional de 4 horas para cada grupo.

O desenvolvimento da unidade instrucional seguiu o modelo ISD (*Instructional Systems Design*) de Dick e Carey (2000), excetuando a etapa de revisar instrução, visto que foi desenvolvida e aplicada somente uma unidade instrucional sem repetição. O modelo é proposto para desenvolvimento de unidades instrucionais, composto por 10 etapas, conforme Figura 4, sendo que a descrição da aplicação das etapas será detalhada na Seção 5.

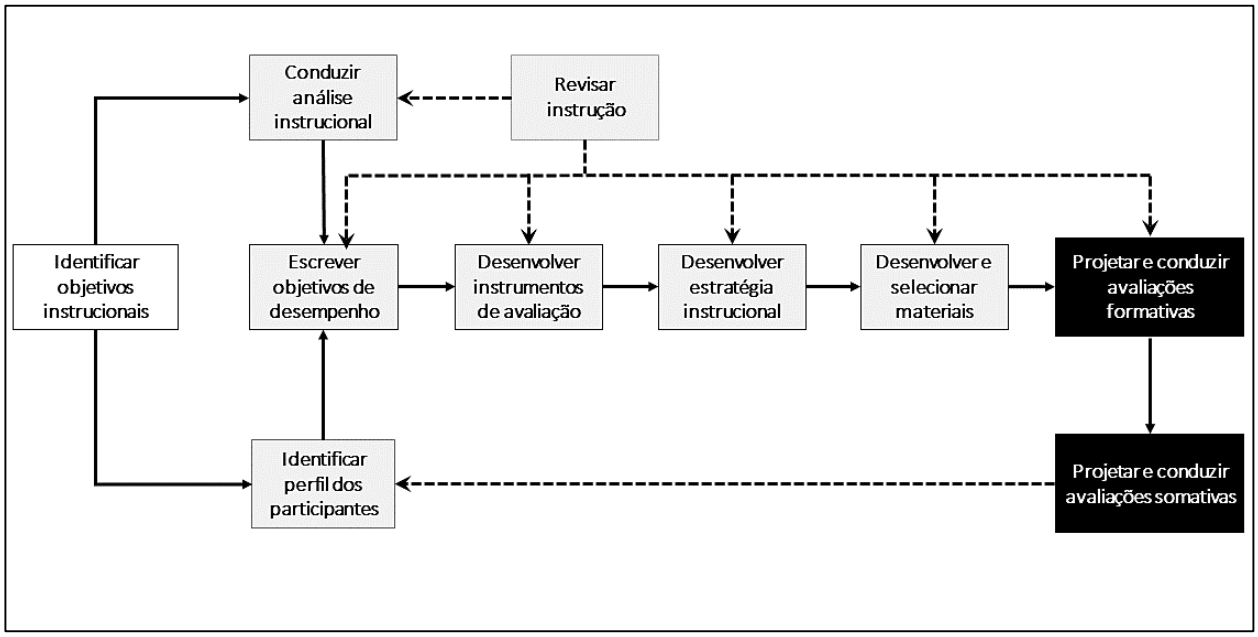


Figura 4: modelo de design instrucional de Dick e Carey
Fonte: Dick e Carey (2000)

O perfil dos aprendizes foi levantado por questionário aplicado no início do processo (Apêndice A). Os objetivos de desempenho e avaliações foram definidos e descritos utilizando a técnica GQM (*Goal Question Metric*) de Basili, Caldiera e Rombach (1994).

Para a execução dos *Dojos*, seguiu-se a metodologia proposta por Bache (2013), seguindo as etapas: i) apresentação; ii) acordo entre os participantes; iii) codificação; e iv) retrospectiva.

Para análise dos resultados, as notas obtidas nas avaliações dos alunos foram submetidas a testes estatísticos. A Figura 5 ilustra o processo do tratamento estatístico.

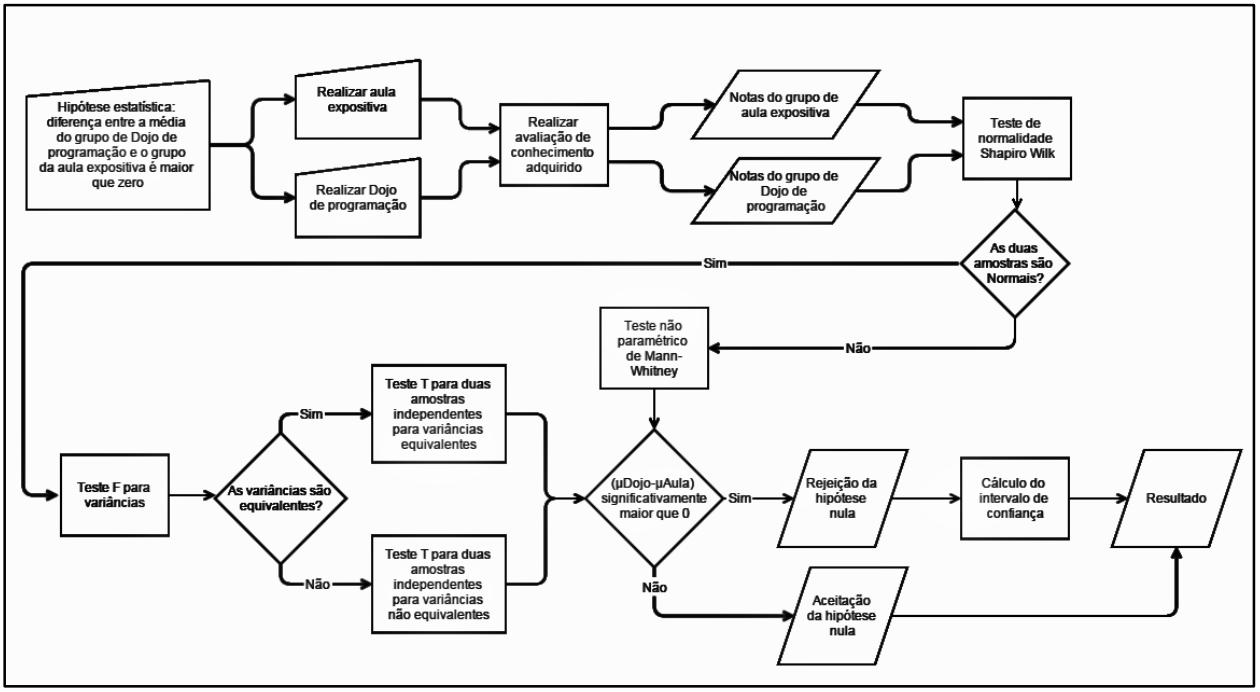


Figura 5: diagrama do processo da metodologia do trabalho estatístico

Como pode ser visto na Figura 5, a resposta para o teste de hipótese proposto neste trabalho é fornecida pelo teste *t* para duas amostras independentes. Tais amostras são representadas pelos conjuntos das notas dos dois grupos que participaram dos encontros de *Dojo* de programação e aula expositiva. Porém, há testes anteriores que precisam ser executados como pré-requisitos. Inicialmente, é executado um Teste de Normalidade para cada uma das duas amostras. Caso existam evidências significativas de que as amostras possuem distribuição Normal, deve-se executar o teste *f* para variâncias, que definirá qual teste *t* para duas amostras independentes deverá ser empregado (variâncias equivalentes ou não).

A pergunta de pesquisa testada neste trabalho é se a utilização de *Dojos* de programação como metodologia de ensino possui desempenho significantemente melhor que a metodologia de ensino tradicional com aulas expositivas. A hipótese nula (H_0) é a média das notas do grupo que participou do *Dojo* é igual ou menor à média das notas do grupo que participou da aula expositiva. Portanto, a hipótese alternativa (H_1) é a de que a média das notas do grupo que participou do *Dojo* é significantemente maior que a média das notas do grupo que participou da aula, conforme o Quadro 2.

Quadro 2: hipótese estatística

$$H_0: (\mu Dojo \leq \mu Aula)$$
$$H_1: (\mu Dojo > \mu Aula)$$

A hipótese alternativa H_1 caracteriza um teste unilateral à direita. O nível de confiança adotado foi de 95%. Desta forma, a hipótese nula H_0 é rejeitada somente se o resultado da amostra for tão diferente do valor suposto que uma diferença igual ou menor ocorreria com uma probabilidade máxima de 5%.

5. O Experimento

O experimento foi realizado entre os dias 24 de setembro de 2014 e 01 de outubro de 2014. Neste período foram realizados três encontros para aula expositiva, *Dojo* de programação e avaliação, respectivamente.

Seguindo o modelo de Dick e Carey (2000) e utilizando o ciclo PDCA para ensino e avaliação segundo Mergen et al. (2014), é detalhado a seguir como foi desenvolvido o design instrucional da unidade aplicada.

5.1. Planejamento

O planejamento envolveu desde a definição do local, turma e conteúdo, até a identificação do perfil dos alunos e objetivos da unidade instrucional. Foram realizadas as seguintes etapas do modelo de Dick e Carey (2000):

a) Identificar metas instrucionais: as metas instrucionais foram identificadas juntamente com o professor, seguindo o objetivo pedagógico do curso e andamento da turma. A meta almejada para a unidade foi que os alunos consigam desenvolver programas simples utilizando conceitos básicos de orientação a objetos;
b) Conduzir análise instrucional: identificou-se que, para atingir as metas, os alunos precisariam entender os conceitos básicos de programação orientada à objetos (objeto, atributos, métodos, classes, instanciação), conhecer os conceitos de abstração, herança, encapsulamento, relacionamentos e polimorfismo e saber aplicar os conceitos básicos utilizando alguma linguagem de programação;
c) Escrever objetivos de desempenho: utilizando a técnica GQM, foram identificadas as seguintes habilidades a serem desempenhas pelos alunos:
    i. Aplicar abstração para transformar problemas reais simples em uma solução orientada a objetos;
    ii. Desenvolver classes que manipulem características (atributos) dos objetos reais;
    iii. Desenvolver classes que simulem comportamentos (métodos) dos objetos reais;
    iv. Desenvolver classes simples que interagem entre si.
d) Analisar aprendizes e contextos: foi realizada por meio de uma pesquisa do perfil dos alunos, além de entrevista com o professor da turma, para identificar o contexto atual da turma. O perfil dos aprendizes é detalhado na Seção 5.1.1.

5.1.1. Perfil dos Aprendizes

Dos 35 alunos da turma, apenas 30 participaram do experimento. Os demais faltaram em algum dos encontros promovidos e, portanto, não foram considerados nas análises.

De acordo com levantamento realizado com esses alunos, apenas 4 tinham ouvido falar sobre programação orientada a objetos, porém, não possuíam experiência no assunto. A Figura 6 apresenta dados sobre a distribuição da turma por gênero e idade.

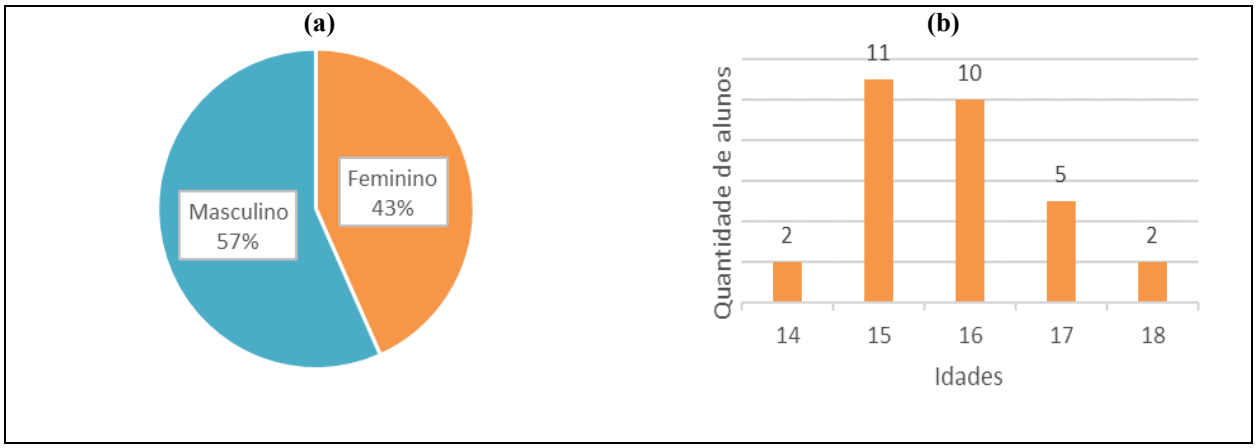


Figura 6 - Distribuição da turma por gênero (a) e idade (b)

5.2. Projeto e Construção da Unidade Instrucional

No planejamento e desenvolvimento das aulas foram desenvolvidos os planos de aula, instrumentos de avaliação e os materiais didáticos utilizados. Seguindo o modelo de Dick e Carey (2000), foram realizadas as seguintes etapas:

a) Desenvolver a estratégia instrucional;
b) Desenvolver os instrumentos de avaliação;
c) Desenvolver e selecionar materiais instrucionais.

5.2.1. Desenvolver a Estratégia Instrucional

Nessa etapa foram criados os planos de aula, contendo o cronograma tanto das aulas expositivas quanto dos *Dojos*, definindo as atividades a serem realizadas (Apêndice B) e carga horária prevista.

A aula expositiva foi planejada contemplando a explicação introdutória dos conceitos de programação orientada a objetos, com tempo de uma hora para a realização de exercícios de fixação. O Quadro 3 apresenta o cronograma previsto para o encontro.

Quadro 3: cronograma da aula expositiva

Atividade	Carga horária
Apresentação do instrutor	10 minutos
Explicação sobre introdução à programação orientada a objetos	2 horas
Intervalo	20 minutos
Explicação sobre introdução à programação orientada a objetos	30 minutos
Apresentação da ferramenta *Scratch*	10 minutos
Prática com a ferramenta *Scratch*	50 minutos

Os conceitos de programação orientada a objetos foram organizados em uma apresentação de *slides*, para ser utilizada como material de apoio pelo instrutor, responsável pela execução da aula expositiva. Foram abordados os aspectos históricos que levaram à utilização dos conceitos de orientação a objetos na programação, além dos conceitos de objeto, atributos, métodos, classes, instanciação, abstração, herança, encapsulamento, relacionamentos e polimorfismo (Apêndice B), com ênfase para os conceitos básicos (objeto, atributos, métodos, classes, instanciação).

Foi planejada uma atividade de construção de um programa de forma colaborativa com os alunos, para que, durante esta atividade, os alunos tivessem contato com a ferramenta *Scratch*

(MIT MEDIA LAB, 2016), e assim exercitassem os conceitos apresentados. Foram reservados os 50 minutos finais da aula para a criação de uma animação que estabelece a comunicação entre dois objetos, programando os atributos e métodos de cada um. Ao final, foram indicados livros para leitura para aprimorar os conceitos.

A infraestrutura necessária para a execução da aula expositiva foi disponibilizada pela instituição SENAI: uma sala de aula com 37 computadores, dos quais 19 foram utilizados pelos alunos do curso e um pelo instrutor, além de um projetor para exibir a apresentação de *slides* e demonstrar a utilização da ferramenta *Scratch*.

A atividade de *Dojo* foi planejada para ser realizada num auditório e também ter duração de 4 horas. Para cumprir o requisito da técnica de *Dojo* de programação de que no grupo devem existir pessoas que possam compartilhar o conhecimento (BACHE, 2013), o instrutor convidou dois programadores profissionais e conhecedores de programação orientada a objetos para participar da atividade. Porém, é importante salientar que estas pessoas não foram avaliadas nem consideradas nos resultados desse trabalho.

Foi optado pelo estilo de *Dojo Randori*, pelo fato de gerar mais interação entre toda turma e permitir uma maior intervenção do instrutor. O Quadro 4 apresenta a programação do *Dojo*.

Quadro 4: cronograma do *Dojo* de programação

Atividade	Carga horária
Apresentação do *Dojo*	15 minutos
Apresentação do *Sensei*	5 minutos
Apresentação de conceitos	30 minutos
Apresentação da ferramenta *Scratch*	10 minutos
Apresentação do desafio de *Dojo*	5 minutos
Dojo Kata	15 minutos
Dojo Randori	105 minutos
Retrospectiva	20 minutos
Indicação de leitura	5 minutos
Confraternização	30 minutos

Conforme o cronograma apresentado no Quadro 4, para a parte inicial do encontro planejou-se a explicação da técnica *Dojo* de programação e a apresentação do *Sensei*, seguida de uma breve apresentação dos conceitos de programação orientada a objetos. Em seguida, foi prevista a apresentação da ferramenta *Scratch*, bem como a apresentação do desafio de *Dojo*: o desenvolvimento de um jogo. Após a etapa de programação, planejou-se a discussão e registro da retrospectiva, indicação de livros para a leitura e confraternização.

5.2.2. Desenvolver os Instrumentos de Avaliação

Foram planejadas duas avaliações: um teste de conhecimentos conceituais e outro de conhecimentos práticos de programação utilizando a ferramenta *Scratch* (MIT MEDIA LAB, 2016). Os testes foram planejados para serem realizados na semana seguinte aos encontros, sendo aplicados à toda turma, sem divisão em subgrupos.

O teste sobre conhecimentos conceituais (Apêndice C) foi definido em 8 questões objetivas referentes ao conteúdo abordado, nas quais os alunos tinham, além das respostas possíveis, uma alternativa de fuga para ignorar a questão. O teste teve peso 5 na nota final de cada aluno, sendo que cada questão teve peso de 0,625.

Para o teste prático, foi planejado que os alunos criassem um jogo que explorasse os conceitos de objetos, atributos, métodos e classes (Apêndice E). Os critérios de avaliação foram pré-estabelecidos de forma que cada componente do programa tivesse um peso na composição na

nota final da avaliação prática (Apêndice F). A nota final individual poderia variar entre 0 e 10, composta por 5 pontos da avaliação teórica e outros 5 da avaliação prática.

5.2.3. Desenvolver e Selecionar Materiais Instrucionais

Para a aula expositiva, foram desenvolvidos *slides* com o conteúdo previsto no plano de aula. Esses *slides* foram baseados em autores consagrados na área e continham conceitos teóricos e aplicação prática dos conceitos em linguagens de programação.

Para a atividade prática de aplicação dos conceitos de orientação a objetos, foi optado por utilizar a ferramenta *Scratch*, por ser mais lúdica e de rápido aprendizagem. Para isso, foram desenvolvidos dois exemplos para serem trabalhados tanto com o grupo de controle quanto com o grupo de tratamento.

5.3. Execução do Experimento

A execução refere-se à interação propriamente dita com os aprendizes, incluindo as aulas e avaliações. O grupo de 30 alunos foi dividido aleatoriamente em dois subgrupos. O primeiro, composto por 19 alunos, foi relacionado para participar de uma aula expositiva. O segundo, composto por 11 alunos, foi relacionado para participar de um *Dojo* de programação. A seção 5.3.1 descreve o relato da aula expositiva (grupo de controle), enquanto a seção 5.3.2 relata o experimento utilizando Dojos (grupo de tratamento).

A divisão dos grupos em tamanhos diferentes deu-se por uma limitação imposta pela técnica do *Dojo* de programação, apresentada no capítulo 2 deste trabalho, que recomenda que o tamanho viável de uma turma de *Dojo* é de 5 a 15 indivíduos. Considerando que foram incluídos ainda dois convidados para participar do encontro do *Dojo* de programação, o tamanho total do grupo ficou estabelecido em 13 participantes, sendo que apenas os 11 alunos foram avaliados e considerados no resultado estatístico deste trabalho.

Conforme Triola (2008), a divisão dos grupos em tamanhos diferentes não interfere no teste estatístico, pois, entre os requisitos que devem ser satisfeitos, basta que os dois tamanhos amostrais sejam maiores que 30 ou que ambas as amostras provenham de populações com distribuições normais.

Seguindo a metodologia de execução de Dojos proposta por Bache (2013), foram realizadas as seguintes etapas:

a) Apresentação;
b) Acordo entre os participantes;
c) Codificação; e
d) Retrospectiva.

Seguindo o modelo de Dick e Carey (2000), foram realizadas ainda as seguintes etapas:

e) Projetar e conduzir avaliações formativas;
f) Projetar e conduzir avaliações somativas.

5.3.1. Relato da Aula Expositiva

Não houve imprevistos com relação à infraestrutura disponível, e, durante a aula, os alunos demonstraram receptividade diante da mudança de professor e também em relação ao tema. A Figura 7 apresenta o grupo de controle na aula expositiva.

Figura 7: aula expositiva

O primeiro ato da aula foi a clarificação do objetivo do trabalho e os participantes foram informados sobre o motivo da divisão em dois subgrupos, assim como a anuência dos termos de consentimento. A ausência de critério para a divisão do grupo facilitou a aceitação do cenário proposto.

Após a apresentação do objetivo da pesquisa e do instrutor, teve início a apresentação conceitual sobre programação orientada a objetos. Conforme planejado, foi utilizada uma apresentação de *slides* projetada na frente da sala como referencial para a condução para as explicações.

Foram abordados todos os conceitos previamente planejados: objeto, atributos, métodos, classes, instanciação, abstração, herança, encapsulamento, relacionamentos e polimorfismo. Porém, a prioridade nas explicações foi para os conceitos de objeto, atributos, métodos e classes.

Na última hora da aula foram aplicados os conceitos de objeto, atributos, métodos e classes na ferramenta *Scratch*. O instrutor demonstrou a utilização da ferramenta através da reprodução da área de trabalho do seu computador no projetor.

Em seguida, os alunos foram encorajados a produzir um programa animando dois objetos, definindo suas características e comportamentos, além de estabelecer um relacionamento entre eles através de instruções lógicas. Pode-se considerar que a atividade prática foi bem-sucedida, pois os 19 alunos do grupo conseguiram finalizar o programa.

5.3.2. Relato do *Dojo* de Programação

O local para a realização do *Dojo* de programação foi cuidadosamente preparado, a fim de enquadrar-se melhor na dinâmica do *Dojo Randori*. Para isto, foi escolhido o auditório da instituição, que dispõe de uma mesa de trabalho, espaço para a plateia e projetor multimídia.

Não houve imprevistos com relação à infraestrutura disponível. Durante o *Dojo*, principalmente no início, percebeu-se uma certa desconfiança por parte dos participantes, pois ainda não haviam sido expostos a uma técnica parecida. A Figura 8 apresenta dois alunos construindo o código fonte para a resolução do desafio, enquanto a plateia assiste a tudo através da projeção multimídia.

Figura 8: *Dojo* de programação

A aplicação do *Dojo* seguiu os passos propostos por Bache (2013):

a) Apresentação;
b) Acordo entre os participantes;
c) Codificação; e
d) Retrospectiva.

Apresentação: o instrutor apresentou-se *Sensei* do *Dojo* e, em seguida, falou brevemente sobre suas experiências profissionais em programação orientada a objetos e docência. Os dois programadores profissionais convidados também foram devidamente apresentados.

Após as apresentações pessoais, o *Sensei* utilizou uma apresentação de *slides* para explicar a dinâmica do *Dojo* de programação. Além dos tipos *Kata* e *Randori*, foram apresentadas as práticas ágeis utilizadas em um *Dojo*: programação em pares, passos de bebê e comunicação.

Antes do início da resolução do desafio de *Dojo*, o *Sensei*, juntamente com um convidado, demonstrou a utilização da ferramenta *Scratch* por meio de 3 rodadas de *Dojo Kata*. Para a demonstração, foi desenvolvida uma animação na qual dois objetos tiveram suas características e comportamentos programados, dando uma visão da ferramenta e das categorias de ações que podem ser executadas por cada objeto (similar ao exercício prático do grupo de controle).

Acordo entre os participantes: nessa etapa foram explicadas todas as regras do *Dojo Randori* e quais as configurações seriam utilizadas. Foram escolhidos os dois participantes que seriam, no início, piloto e copiloto, o *timebox* (tempo para troca dos pares), ordem de troca e também a forma de interação do par com a plateia e com os convidados. Foi definido que cada ciclo deveria ter 5 minutos, e para controlar isto, foi instalado um contador de tempo no computador.

Nessa etapa também foi explicado o objetivo da pesquisa e a necessidade do preenchimento dos formulários de perfil e avaliações posteriores, solicitando a ciência e acordo em participação dos alunos.

Codificação: o desafio de *Dojo* proposto à turma foi a construção de um jogo, no qual uma bola se movimenta pela tela e é rebatida por uma barra horizontal na parte inferior. No momento em que houver falha na rebatida e a bola encostar na base da tela, o jogo é encerrado. Foi solicitado aos alunos que identificassem os objetos presentes na tela, bem como as características e comportamentos de cada um. A Figura 9 ilustra o jogo.

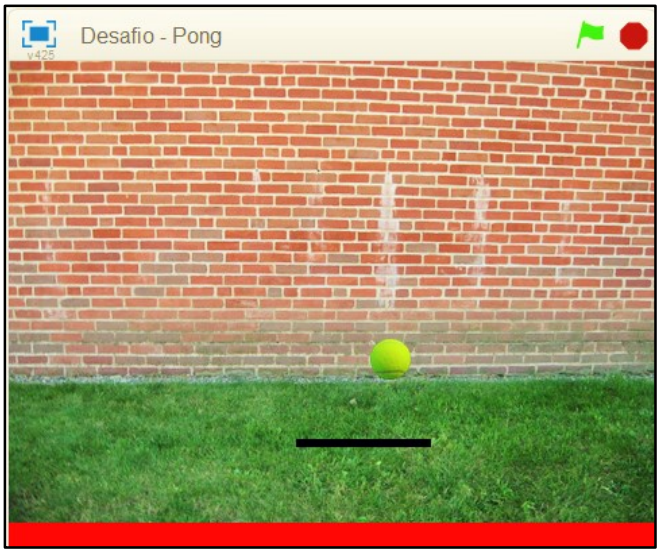


Figura 9: desafio de *Dojo* - Jogo *Pong*

O término da apresentação do desafio de *Dojo* foi o gatilho para o início do *Dojo* de programação no estilo *Randori*. A execução do *Dojo* teve duração de 80 minutos. Sendo que sobraram 45 minutos do tempo estimado (considerando o tempo da retrospectiva), os alunos optaram por utilizar o tempo restante para programar um segundo desafio (jogo de labirinto). Para que todos pudessem participar, o tempo dos ciclos de utilização do computador foi reduzido para 3 minutos. Porém, o jogo de labirinto não foi concluído em tempo hábil.

Durante o trabalho de programação, os alunos que estavam utilizando o computador foram constantemente lembrados pelo *Sensei* sobre a necessidade de explicar seus raciocínios à plateia, além de relatar as atividades que estavam realizando. Além disso, o *Sensei* interviu estrategicamente em determinados momentos para evidenciar a aplicação de conceitos de programação orientada a objetos.

Retrospectiva: após o término do *Dojo Randori*, os participantes foram encorajados a relatar os aspectos positivos e as oportunidades de melhoria observadas durante a realização do evento. A maioria dos comentários foi positiva, e os temas mais comentados foram relacionados à diversão, integração e quebra de rotina. Destaca-se uma frase referente à refatoração: “ver novas formas de pensar ao ver as mudanças”.

Sobre as oportunidades de melhoria, foi relatado que 5 minutos é pouco tempo para desenvolver um raciocínio, e também que o início do *Dojo* foi caótico, precisando de um tempo até que a turma atingisse um bom nível de produtividade. A Figura 10 apresenta os principais relatos fornecidos pelos alunos, agrupados por similaridade.

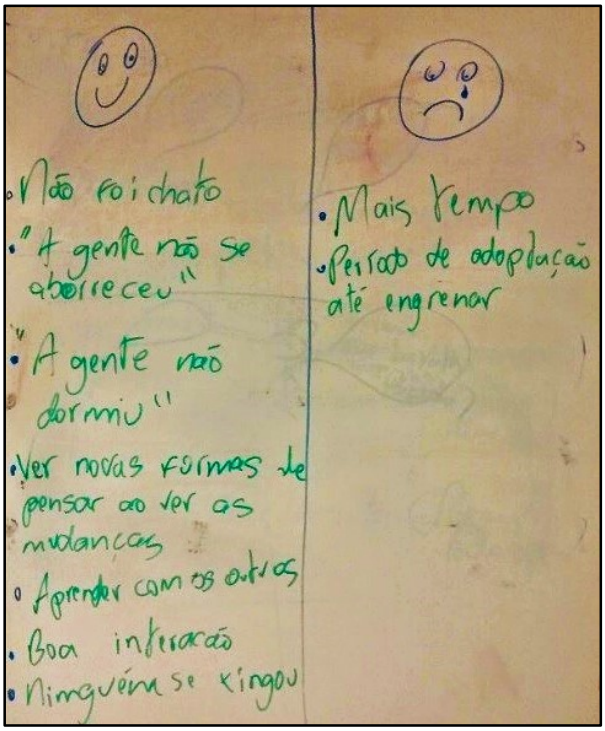


Figura 10: Retrospectiva do *Dojo* de programação

Percebe-se na Figura 10 que a maioria dos relatos foram positivos e relacionados, principalmente, à motivação, interação entre participantes e aprendizagem.

Mesmo com a característica essencialmente prática, após duas horas de realização de *Dojo*, a turma também apresentou desgaste e ensaiou uma dispersão, o que reforça que um *Dojo* de programação não deve se estender por muito tempo. Quando questionados se voltariam a participar de um *Dojo* de programação, a resposta positiva foi unânime.

Como a turma que participou do *Dojo* de programação não teve acesso às explicações conceituais sobre programação orientada a objetos, o *Sensei* solicitou aos participantes uma pesquisa para ser entregue na semana seguinte. A encomenda foi um manuscrito contendo a interpretação dos conceitos de objeto, atributo, método e classe. Este trabalho não foi avaliado, e teve como único objetivo o incentivo à leitura. A mesma lista de leitura e atividades foi sugerida para o grupo de controle (aulas expositivas), para evitar vieses nos resultados em decorrência dessa atividade diferenciada.

Após a retrospectiva, foi realizada uma confraternização, onde os dois grupos foram reunidos, para que houvesse interação entre os participantes. Uma constatação interessante sobre este momento é que os alunos que participaram do *Dojo* de programação continuavam a falar sobre a solução do desafio.

5.4. Controle e Avaliação

Uma semana após os encontros foram realizadas as avaliações planejadas previamente. Optou-se por realizar a avaliação após um certo período de tempo por dois motivos: i) verificar a fixação do conhecimento a médio prazo (não limitado ao relembrar fatos ou dados que os participantes tinham acabado de estudar); e ii) devido ao cronograma de aulas e disponibilidade da turma.

Os alunos responderam oito questões objetivas sobre conceitos de programação orientada a objetos em um formulário *online* (Apêndice C), além de programar os objetos de um jogo de corrida (Apêndice E) na ferramenta *Scratch*. As notas foram obtidas segundo o gabarito das questões objetivas (Apêndice D) e os critérios da avaliação prática (Apêndice F).

6. Resultados

A avaliação pré-teste sobre conhecimentos em programação orientada a objetos mostrou que os alunos possuíam pouco ou nenhum conhecimento sobre o assunto. A nota média (considerando nota máxima 10) ficou em 1,12 com desvio padrão de 1,32. Como no pré-teste foi incluída uma alternativa de escape (Não sei) e para evitar vieses devido ao eventual acaso de maior quantidade de acertos para alunos que tentaram responder todas as questões, para fins de comparação foi utilizado a diferença entre acertos e erros. A Tabela 1 mostra os valores encontrados.

Tabela 1 – Médias da avaliação pré-teste

Grupo	*Média*	*Desvio Padrão*
Aula expositiva	-1,74	2,38
Dojo	-0,64	2,46
Média Geral	-1,41	2,39

Os dados da Tabela 1 mostram que o desempenho inicial dos alunos que integraram o grupo que participou do *Dojo* teve desempenho melhor que o grupo que participou da aula expositiva, apesar da aleatoriedade na divisão dos grupos.

Considerando somente as avaliações teóricas e comparando a evolução dos grupos, percebeu-se que ambos os grupos tiveram evolução significativa, sendo que o grupo que participou da aula expositiva teve evolução levemente superior, embora as médias do grupo de tratamento (*Dojos*) tenha sido maior, conforme Tabela 2.

Tabela 2 – Comparação da avaliação pré e pós tratamento

Grupo	*Pré Teste*	*Pós Teste*	*Diferença*	*Valor p*
Aula expositiva	-1,74	2,00	3,74	
Dojo	-0,64	2,73	3,36	0,4133
Média Geral	-1,41	2,27	3,68	

Aplicando o teste *t* para verificar se existe diferença significativa entre os grupos, percebe-se que não se pode rejeitar a hipótese de que os grupos são diferentes ($p > 0,05$), concluindo-se que a evolução média dos grupos, na avaliação teórica, foi similar.

A fim de avaliar o impacto do experimento considerando também a avaliação prática, foram realizados testes comparando o desempenho dos grupos na avaliação pós tratamento, envolvendo ambas as notas. Nesse caso, os dados amostrais afirmam, estatisticamente, que o método de ensino utilizando *Dojos* de programação produziu médias significativamente maiores que o método tradicional de ensino utilizando de aulas expositivas, com mais de 99% de confiança.

As notas foram classificadas de acordo com os grupos: 19 participantes da aula expositiva e 11 participantes do *Dojo* de programação. Os valores são apresentados abaixo:

a) Notas da aula expositiva: {4.28, 4.43, 4.45, 4.53, 4.53, 4.63, 4.8, 4.85, 5.88, 6.08, 6.15, 6.2, 6.25, 6.35, 6.45, 6.5, 6.9, 7.63, 8.08}.
b) Notas do *Dojo* de programação: {4.48, 6.2, 6.28, 6.28, 6.95, 7.1, 7.85, 7.9, 9.03, 9.28, 9.65}.

As duas amostras foram submetidas ao Teste de Normalidade de Shapiro-Wilk. Após a execução do teste, verificou-se que haviam evidências significativas de que as amostras possuem distribuição Normal ($p > 0.05$), embora a amostra das notas de aulas expositivas tenha ficado próximo ao limite para ser considerada normal. A Figura 11 apresenta os valores *p* obtidos para as notas do grupo de aulas expositivas (a) e do grupo do *Dojo* (b) e a distribuição dos valores das amostras.

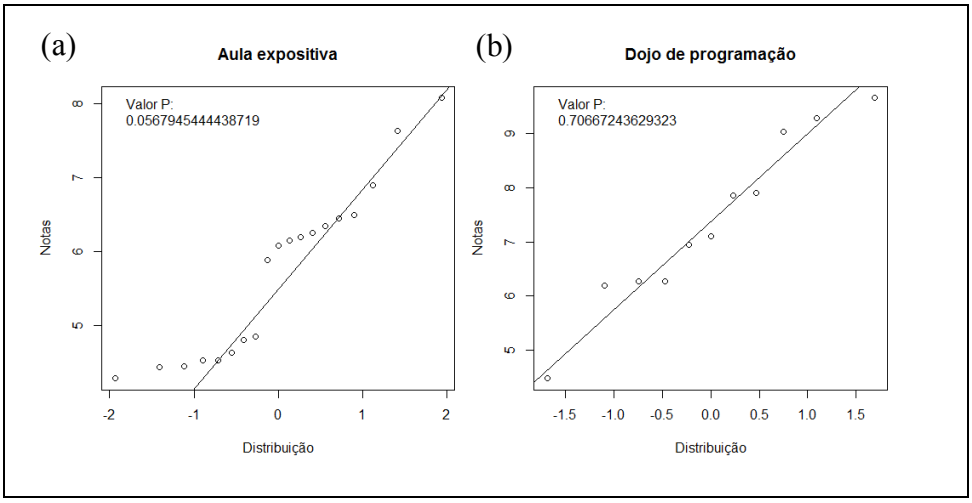


Figura 11 - Teste de Normalidade das notas

Para parametrizar corretamente o teste *t*, foi necessário descobrir se as variâncias das amostras eram equivalentes, através do teste *f* para variâncias. O resultado deste teste apontou que as variâncias das duas amostras não são significativamente diferentes. O valor calculado de *p* (0.2513) foi maior que o nível de significância de 0.05, sendo possível aceitar a hipótese de que as variâncias são equivalentes e o teste *t* deve ser parametrizado desta forma.

O resultado da execução do teste *t* levou à rejeição da hipótese nula, com nível de confiança de 95% (p = 0,004). A diferença entre a média das notas do grupo que participou do *Dojo* de programação foi significativamente maior do que zero. Os valores das médias são listados abaixo, juntamente com o resultado da diferença entre as duas médias:

a) $\mu Dojo$: 7.36;
b) $\mu Aula$: 5.74;
c) Diferença $\mu Dojo - \mu Aula$: 1.62.

A Figura 12 apresenta a distribuição dos valores das duas amostras, permitindo uma abordagem visual da significância da diferença entre as médias. É importante notar que os valores de máximo e mínimo se referem à amplitude amostral, não existindo valor probabilístico associado aos mesmos, ou seja, não representam um intervalo de confiança.

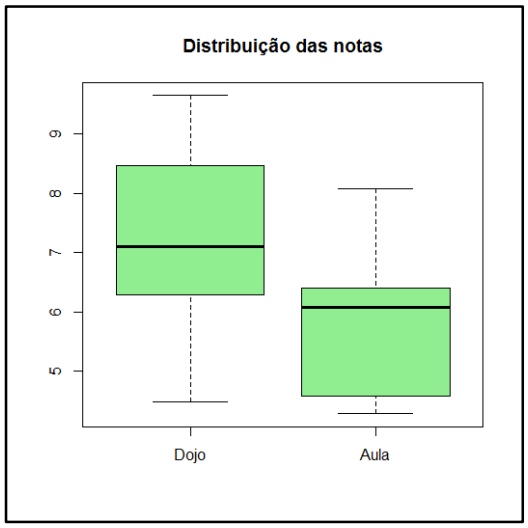


Figura 12 - distribuição das notas das amostras

Por fim, foi computado o intervalo de confiança baseado na estatística t, considerando nível de confiança de 95%. Este cálculo resultou em intervalos disjuntos:

a) *Dojo* de programação: variação entre 6.31 e 8.41
b) Aula expositiva: variação entre 5.18 e 6.29

A representação gráfica deste resultado pode ser visualizada a seguir, nas áreas destacadas na Figura 13.

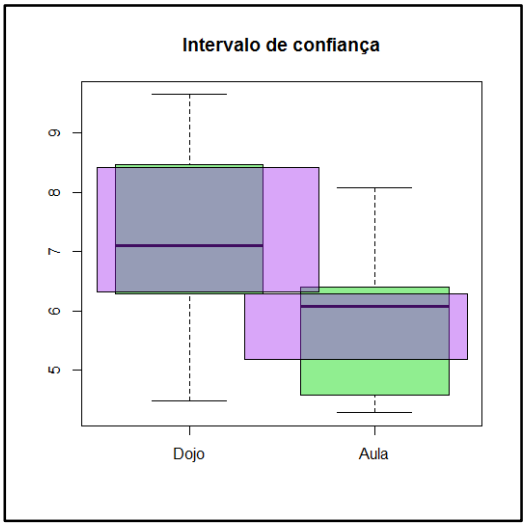


Figura 13 - demonstrativo do intervalo de confiança de cada amostra

Para avaliação do teste prático, os conteúdos foram divididos em três categorias: i) criação objetos; ii) propriedades dos objetos; e iii) comportamento dos objetos. A Tabela 3 compara as médias dos grupos de controle e tratamento para cada categoria de conteúdo, assim como o resultado da aplicação do teste *t,* a fim de verificar se existe diferença significativa entre os grupos.

Tabela 3 – Comparação do desempenho nos conteúdos da avaliação prática

Conteúdo	*Grupo de Controle*	*Grupo de Tratamento*	*Valor p*
Criação	81,1%	81,8%	0,494
Características	47,1%	54,8%	0,277
Comportamento	32,1%	68,2%	0,049
GERAL	40,2%	61,7%	0,028

A Tabela 3 mostra que, apesar do desempenho dos estudantes do grupo de tratamento ter sido melhor nas três categorias de conteúdo, somente nas questões referente ao comportamento de objetos e na média geral pode-se evidenciar diferença significativa, estatisticamente a um nível de confiança de 95% ($p < 0,05$). A média geral considerou ainda fatores como dinâmica do jogo e lógica utilizada.

Analisando os resultados, foi constatada uma possível diferença entre o desempenho de estudantes do sexo masculino e feminino. A Tabela 4 mostra uma comparação entre as médias obtidas para cada gênero em cada avaliação e a análise da significância dessa diferença utilizando o teste *t*.

Tabela 4 – Desempenho por gênero

	Avaliação Teórica[1]		*Avaliação Prática*[2]		*Nota Final*	
Gênero	Geral	Grupo de Tratamento	Geral	Grupo de Tratamento	Geral	Grupo de Tratamento
Feminino	3,38	1,75	2,67	3,08	5,84	6,20
Masculino	3,76	4,29	3,47	4,54	6,71	8,03
Valor de p	0,409	0,178	0,028	0,026	0,056	0,037

[1] Considera a evolução (diferença entre pré e pós teste)
[2] Considerando nota máxima 6,5
[3] Considerando a média ponderada entre avaliação teórica (5) e prática pós testes (5)

Pode-se notar, na Tabela 4, que existe uma diferença entre os gêneros na média final (média dos estudantes de sexo masculino é maior que do sexo feminino), com 94,4% de confiança ($p = 0,056$). O mesmo acontece para a avaliação prática ($p = 0,028$). Percebe-se também que essa diferença aumenta se considerar somente o grupo que participou do *Dojo* de programação. Para as avaliações teóricas, embora perceba-se diferenças, não foi possível comprovar estatisticamente.

6.1. Discussão dos Resultados

Os resultados indicam, de maneira geral, que o uso de *Dojos* de programação como estratégia de ensino para programação orientada à objetos tem um resultado positivo significante na aprendizagem. É interessante destacar que na avaliação teórica, embora o grupo que participou do *Dojo* tenha tido pouca exposição de teoria, obteve uma evolução similar ao grupo de controle. Isso pode indicar que a aprendizagem por meio da prática, além de ajudar a desenvolver as habilidades de programação efetivamente, utilizando linguagens, ferramentas, etc., não prejudica o entendimento e conhecimento da teoria.

Alguns resultados secundários foram analisados, após serem verificados os dados e evidenciado tendências nos resultados, considerando os fatores de gênero dos participantes e desempenho por conteúdo. Para isso, foram realizadas comparações dos resultados nas avaliações teóricas, práticas e nota final entre os participantes do sexo feminino e masculino, além da comparação do desempenho para cada categoria de conteúdo (criação, características e comportamentos de objetos) para os dois grupos. Pode perceber-se, com isso, que existiram diferenças de desempenho, tanto relacionado ao conteúdo de comportamento de objetos (onde o grupo que participou do Dojo obteve um desempenho significativamente melhor), quanto nas médias por gênero (onde os participantes masculinos tiveram um desempenho significativamente maior).

Embora não tenha sido o principal objetivo do experimento, percebeu-se também, por meio da observação e *feedback* dos alunos, que houve maior envolvimento e motivação dos alunos que participaram dos *Dojos*.

6.2. Ameaças à Validade da Pesquisa

Alguns fatores resultantes das observações do autor, bem como os próprios parâmetros da pesquisa, podem ter influência no resultado apresentado. O curto período do experimento, pois o único encontro de quatro horas tanto para o *Dojo* de programação quanto para a aula expositiva, mediu a aquisição imediata e parcial de conhecimento, não necessariamente sendo generalizado para períodos mais longos.

O fato do estudo abordar somente um tema também é uma ameaça à validade da pesquisa, pois conhecimentos prévios e/ou pré-disposição para entendimento do assunto trabalhado por ter resultado num desempenho melhor para alguns alunos, o que não seria representado em outros conteúdos.

Outro fato que merece destaque é que as contribuições dos dois programadores profissionais no *Dojo* de programação podem ter influência no melhor resultado do grupo, embora seja uma prática sugerida para *Dojos* estilo Randori. Além disso, a não realização de encontros recorrentes não permitiu avaliar o desinteresse gradativo por alguma das metodologias de ensino.

A participação dos alunos não foi encorajada, mas sim obrigatória, assim como não houve critério para divisão dos grupos. Existe o risco de alunos com melhor desempenho terem sido direcionados, mesmo que aleatoriamente, para o grupo que participou do *Dojo* de programação, embora a abordagem estatística do trabalho apresente a Normalidade das amostras.

7. Considerações Finais

Esse trabalho teve como objetivo avaliar o desempenho de alunos na aprendizagem de programação orientada a objetos com a utilização da técnica de *Dojo* de programação. Para isso, foi realizado um experimento, onde foi aplicada uma unidade instrucional para um grupo de controle, seguindo uma abordagem metodológica de aulas expositivas e exercícios individuais, e o mesmo conteúdo foi aplicado para um grupo de tratamento, adotando a abordagem de *Dojos* de programação, estilo *Randori*.

Foram realizadas avaliações pré e pós-teste teóricas e pós-teste práticas, cujos resultados foram submetidos a testes estatísticos. O experimento resultou em evidências significativas, no contexto aplicado, de que os alunos que participaram do *Dojo* de programação demonstraram melhor desempenho, rejeitando a hipótese nula. Considerando as notas pós-teste, incluindo avaliações teóricas e práticas, a média do grupo que utilizou *Dojo* foi de 7.36, enquanto a média do grupo que teve aula expositiva foi de 5.74, resultando num valor $p = 0.004$ (teste *t*) e evidenciando a diferença significante entre as médias.

Já nos testes comparando a evolução do desempenho dos grupos nas avaliações teóricas (pré e pós teste), não foram encontradas evidências de diferença significativa. Isso pode indicar que, mesmo que os alunos do grupo de controle tenham tido uma abordagem mais teórica, o desempenho nas avaliações foi similar aos alunos do grupo de tratamento.

Alguns resultados secundários também foram percebidos e se mostraram interessantes. Por exemplo, foi identificado que, nas avaliações teóricas, o gênero não influenciou nos resultados, porém nas avaliações práticas e na nota final a média dos estudantes do sexo masculino foi significativamente maior que do sexo feminino. Considerando somente os participantes do grupo de participou do *Dojo*, essa diferença foi ainda mais significativa. Isso pode ser um indício de que os *Dojos* de programação são mais eficientes para estudantes masculinos ou mesmo que as meninas possuem mais dificuldades em atividades práticas de programação. Porém, essas suposições requerem mais investigações para quaisquer confirmações.

Percebeu-se também que a maior diferença entre os grupos, na avaliação prática, foi relacionada ao conteúdo de comportamento de objetos. Uma hipótese para tal resultado é que os demais conteúdos (criação e características de objetos) podem ser mais facilmente aprendido por aulas expositivas e mais teóricas, enquanto a definição e programação dos comportamentos dos objetos (métodos), requer uma abordagem mais prática e colaborativa.

Foram encontrados poucos trabalhos sobre o assunto de Dojos de programação e nenhum dos trabalhos encontrados propõe uma avaliação da evolução da aprendizagem (pré e pós testes) e a análise estatística comparando essa evolução com a abordagem tradicional de aulas expositivas. Essa é a principal contribuição do trabalho, sendo um dos primeiros estudos que sugere a eficácia da técnica de Dojos de programação na aprendizagem.

Esses são resultados iniciais e precisam de mais investigações em outros grupos para conclusões mais precisas. Conforme explica Luz (2013), a utilização de *Dojos* de programação deve ser intercalada com aulas expositivas para apresentação de conceitos. Isso pode ser verificado quando, após um determinado tempo, iniciou-se uma dispersão dos alunos participantes do Dojo, mesmo que tenha sido identificado um considerável envolvimento e motivação.

Outro fator importante que foi observado e relatado pelos participantes na retrospectiva do *Dojo* de programação foi o alto grau de socialização. Além de pessoas consideradas tímidas terem ficado à vontade com a exposição, houve muito diálogo e respeito entre as pessoas. É importante ressaltar que essa foi uma observação empírica e baseada em relatos dos participantes, mas não foi realizado um estudo que pudesse comprovar ou não alguma alteração no grau de socialização dos alunos na atividade de Dojo em comparação com as aulas expositivas.

Acredita-se que o uso de *dojos* para o ensino não deva ser utilizado como única estratégia de ensino, mas sim intercalada com outras técnicas. Acredita-se também que a técnica possa ter

resultados positivos com outros assuntos e diferentes públicos-alvo. Para isso, sugere-se a replicação ou evolução desse trabalho para outros contextos.

Como próximos trabalhos a serem realizados nesta área, sugere-se a possibilidade de um estudo comparativo do início ao fim de um ano/semestre letivo, buscando identificar a forma mais eficiente de ensino de programação, através da combinação de técnicas e ferramentas. Além disso, é de interesse do mercado capacitar profissionais de desenvolvimento de *software* com menores custos e melhores resultados. Portanto, a utilização de *Dojos* de programação em ambientes corporativos, avaliando o desempenho neste cenário, também é uma sugestão de trabalho futuro.

Sugere-se também que se ampliem estudos e experimentos na utilização de *Dojos* de programação para o ensino, replicando esse estudo em outros níveis de ensino e conteúdos, para que possam ter mais evidências das vantagens desse tipo de metodologia e também verificar a quais contextos são aplicáveis com eficácia.

Referências

Alvares, L. and Batista, F. F. (2007). Ciência da Informação e Gestão do Conhecimento**:** a convergência a partir da Sociedade da Informação. Encontro Nacional de Pesquisa em Ciência da Informação. Salvador: Outubro, 2007.

Bache, E. (2013). The Coding Dojo Handbook: *A practical guide to creating a space where good programmers can become great programmers.*

Basili, V., Caldiera G. and Rombach, D. (1994). The Goal Question Metric Paradigm.

Borba, S. F. P. (2015) Ações realizadas em um curso superior de tecnologia para reduzir sua evasão. In: XXXV Congresso da Sociedade Brasileira de Computação. Recife, 2015.

Bovota, G. et al. (2012) Teaching Software Engineering and Software Project Management: an integrated and practical approach. In: 34th International Conference on Software Engineering (ICSE), 2012. p. 1155 – 1164.

Bravo, M. Goldman, A. Reinforcing the Learning of Agile Practices Using Coding Dojos. (2010). In: Agile Processes in Software Engineering and Extreme Programming. Sillitti, A. Martin, A. Wang, X. Whitworth, E. 11th International Conference, XP 2010, Trondheim, Norway, June 1-4, 2010. Proceedings.

Carmo, D. H. and Braganholo, V. (2012). Um Estudo sobre o Uso Didático de DOJOs de Programação. In: XX Workshop sobre Educação em Computação (WEI2012), p. 1-10..

Dick, W., Carey, L. (2000). The Systematic Design of Instruction. Glenview, IL: Scott, Foresman, and Company.

Drew, C. (2011). Why Science Majors Change Their Minds (It's Just So Darn Hard). New York Times. 4 nov. 2011. Disponível em: <http://www.nytimes.com/2011/11/06/education/edlife/why-science-majors-change-their-mind-its-just-so-darn-hard.html?_r=3&pagewanted=all>.

Ericsson, K. A., Krampe, R. T., Tesch-Romer, C. (1993). The role of deliberate practice in the acquisition of expert performance. Psycological Review, 100(3): 363-406, 1993.

Estacio, B. Valentim, N. Rivero, L. Conte, T. Prikladnick, R (2015a). "Evaluating the Use of Pair Programming and Coding Dojo in Teaching Mockups Development: An Empirical Study". In: 48th Hawaii International Conference on System Sciences, IEEE, p. 5084-5093.

Estacio, B. Oliveira, R. Marczak, S. Kalinowski, M. Garcia, A. Prikladnicki, R. Lucena, C. (2015b). Evaluating Collaborative Practices in Acquiring Programming Skills: Findings of a

Controlled Experiment. In: Proceedings of Congresso Brasileiro de Software: Teoria e Prática (CBSOFT 2015), Belo Horizonte, MG, Brazil, 2015.

Ferreira, C. E. (2015) Evasão: alguns dados e mutas dúvidas . In: XXXV Congresso da Sociedade Brasileira de Computação. Recife, 2015.

Fontes, B. B. (2011). Coding Dojo: Novas Possibilidades Para o Ensino de Programação. 60 p. Monografia, Instituto de Computação, Universidade Federal Fluminense. Niteroi, 2011.

Heinonen, K. Hirvikosko, K. Lukkainen, M. Vihavainen, A. (2013). "Learning Agile Software Engineering Practices using Coding Dojo". In: Proceeding of SIGITE'13, Orlando, EUA.

Hüsing, T. Korte, W. Fonstad, N. Lanvim, B. Cattaneo, G. Kolding, M. Lifonti, R. Welsum, D. (2013). e-Leadership: e-Skills for Competitiveness and Innovation Vision, Roadmap and Foresight Scenarios . Report for European Comission.Relvas, A. (2016). Objetivo Lua: como a prática deliberada e o coaching cria profissionais de alto desempenho. Disponível em: <http://objetivolua.com/pratica-deliberada-coaching-desempenho/>. Acesso em 12/02/2016.

Locke, J. (1997). Ensaio Acerca do Entendimento Humano. São Paulo: Nova Cultural, 1997. (Coleção Os Pensadores).

Luz, R. (2013). A influência do Dojo de Programaçao no ensino de práticas ágeis. 69 f. Dissertação – Programa de Pós-graduação em Computação Aplicada, Universidade Tecnológica Federal do Parana. Curitiba, 2013.

Luz, R. Neto, A. (2013). Usando Dojos de Programação para o Ensino de Desenvolvimento Dirigido por Testes. Departamento Acadêmico de Informática Universidade Tecnológica Federal do Paraná (UTFPR). Curitiba, 2013.

Luz, R. Neto, A., Noronha, R. V. (2013). Teaching TDD, the Coding Dojo Style. In: 2013 IEEE 13th International Conference on Advanced Learning Technologies. IEEE. p.371-375, 2013.

Matsuda, M. K., Santos, V., Rodrigues, R. (2013) Aprendizado contínuo na prática com Coding Dojos. Revista MundoJ, Curitiba-PR, Novembro e Dezembro. 2013.

McGettrick A.D. Boyle, Ibbett, R. Lloyd, J. Lovegrove, G. Mander, K. (2005). Grand challenges in computing: Education - a summary. Computer Journal, v. 48, p. 42-48, 2005.

Mergen, S.L.S. Kepler, F.N. Silva, J.P.S. Cera, M.C. (2014). Using PDCA as a General Framework for Teaching and Evaluating the Learning of Software Engineering Disciplines. iSys – Revista Brasileira de Sistemas de Informação, Rio de Janeiro, vol. 7, n. 2, p. 3-24.

MIT Media Lab. (2016). Scratch, versão 2.0. Lifelong Kindergarten Group. Disponível em: <http://scratch.mit.edu>. Acesso em 12/02/2016.

Mooney, O. Patterson, V. O'Connor, M. Chantler, A. (2010). A Study of Progression in Irish Higher Education . Report by the HEA – High Education Authority.

Niitsoo, M. Paales, M. Pedaste, M. Siiman, L. Tõnisson, E. (2014). Predictors of Informatics Students Progress and Graduation in University Studies. Proceedings of INTED2014 Conference 10th-12th March 2014, Valencia, Spain .

Palmeira, L.B. Santos, M. P. (2014). Evasão no Bacharelado em Ciência da Computação da Universidade de Brasília: análise e mineração de dados . Monografia do curso de Ciência da Computação da UnB – Universidade de Brasilia. Brasília : UnB, 2014. 287 p.

Relvas, Ana. Objetivo Lua. Como a prática deliberada e o coaching cria profissionais de alto desempenho. (2016). Disponível em: <http://objetivolua.com/pratica-deliberada-coaching-desempenho/>. Acesso em 10/04/2016.

Robins, A. Rountree, J. Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. Computer Science Education, 2003, v. 13, n. 2, p. 137-172.

Rocha, F. G., Sabino, R. F., Esan, J. (2014). O Uso de Dojo na Prática Pedagógica do Ensino de Lógica de Programação. In: Congresso Brasileiro de Software: Teoria e Prática. FEES 2014: VII Fórum de Educação em Engenharia de Software, p. 62-65, 2014.

Rodrigues, F. S. (2013) Estudo Sobre A Evasão No Curso De Ciência Da Computação Da UFRGS. Trabalho de Conclusão do Curso de Ciência da Computação da UFRGS. Porto Alegre, 2013.

Sato, D., Corbucci, H., Bravo, M. (2008). Coding Dojo: an environment for learning and sharing Agile practices. In: Proceedings of Agile 2008 Conference. IEEE, p. 459-464, 2008.

Tan, P. Ting, C. Ling, S. (2009). Learning Difficulties in Programming Courses: Undergraduates' Perspective and Perception. In: 2009 International Conference on Computer Technology and Development.

Triola, M. F. (2008). Introdução à Estatística. 10a Edição. Editora LTC. 2008.

Veronese, T. B. Lemos, R. M. (2015). SAPERE - Sistema de Apoio ao Ensino para Redução da Evasão . In: XXXV Congresso da Sociedade Brasileira de Computação. Recife, 2015.

Watson, C. and Li, F. W. (2014). Failure rates in introductory programming revisited. In: Procedings of Innovation and Technology in Computer Science Education (ITiCSE 2014). ACM.

APÊNDICES

APÊNDICE A - Formulário de pesquisa de conhecimento prévio em Orientação a Objetos

Informe seu nome completo *

_____

Informe sua idade *

_____

1. Você já ouviu falar sobre programação orientada a objetos? *
( ) Sim
( ) Não

2. Marque a alternativa que melhor representa o conceito de objeto:
( ) Somente entidades físicas (ex. aluno, carro)
( ) Somente entidades conceituais (ex. empréstimo, processo químico)
( ) Somente entidades de software (ex. lista, fila)
( ) Entidades físicas, conceituais e de software
( ) Nenhuma das alternativas acima
( ) Não sei

3. Marque a opção correta. Um objeto possui três características elementares, são elas:
( ) Hierarquia, polimorfismo de sobrecarga, polimorfismo de sobreposição
( ) Método estátatico, variável e atributos
( ) Estado, comportamento e identidade
( ) Público, privado e friend
( ) Nenhuma das alternativas
( ) Não sei

4. Um aluno chamado José Carlos do curso de Informática efetua matrícula nas disciplinas de Matemática, Física e Programação. Quais são os objetos da sentença?
( ) Aluno, curso, matrícula e disciplina
( ) José Carlos, Informática e disciplinas
( ) José Carlos, Informática, Matemática, Física e Programação
( ) Computador, José Carlos, Informática, Matemática, Física e Programação
( ) Não há objetos, somente classes
( ) Não sei

5. Marque a alternativa incorreta em relação à definição de classe:
( ) Uma classe é uma lista de objetos.
( ) Uma classe serve como template (modelo) para se criar objetos.
( ) Uma classe é uma representação de um conjunto de objetos que possuem atributos, relações e operações idênticas
( ) Instâncias de classes são objetos.
( ) Classes não representam necessariamente um conjunto de objetos do mundo real, podem representar objetos de software.
( ) Não sei

6. Em relação aos atributos, é correto dizer que:

( ) Uma classe deve ter ao menos um atributo.
( ) Atributos e operações são sinônimos
( ) São propriedades de uma classe que definem os valores possíveis para as instâncias das propriedades
( ) Atributos somente são utilizados em tabelas de banco de dados.
Não sei

7. Um dos conceitos básicos de orientação a objetos é o fato de um objeto, ao tentar acessar as propriedades de outro objeto, deve sempre fazê-lo por uso de métodos do objeto ao qual se deseja atribuir ou requisitar uma informação, mantendo ambos os objetos isolados. A essa propriedade da orientação a objetos se dá o nome de:
( ) Herança
( ) Abstração
( ) Polimorfismo
( ) Mensagem
( ) Encapsulamento
( ) Não sei

8. Polimorfismo é
( ) A multiplicidade de atributos de determinada classe.
( ) A propriedade de um diagrama de classes ter múltiplas classes possuidoras de atributos.
( ) A habilidade de um atributo ou variável poder identificar instâncias de classes com atributos dependentes.
( ) A propriedade de uma instrução poder apontar para múltiplos objetos de uma mesma classe sem implicações de desempenho.
( ) A habilidade pela qual uma única operação ou nome de atributo pode ser definido em mais de uma classe e assumir implementações diferentes em cada uma dessas classes.
( ) Não sei

9. Dentro dos paradigmas da orientação a objetos, um recurso é utilizado para indicar a capacidade de abstrair várias implementações diferentes em uma única interface. Nesse caso, um objeto pode enviar a mesma mensagem para objetos semelhantes, mas que implementam a sua interface de formas diferentes. Este recurso é conhecido por:
( ) Polimorfismo
( ) Encapsulamento
( ) Não sei

10. São conceitos chaves do paradigma Orientado a Objetos:
( ) Classes, objetos, regras e funções.
( ) Casamento de padrões, herança, classes e objetos.
( ) Classes, objetos, herança e polimorfismo por inclusão.
( ) Polimorfismo por inclusão, casamento de padrões, transparência referencial e herança.
( ) Sobrecarga, inferência lógica, backtracking e herança.
( ) Não sei

11. Em um programa orientado a objetos, em que as tarefas são definidas como comportamento de objetos, cada objeto é criado a partir da instanciação de uma classe onde os seus métodos estão implementados.
( ) Certo
( ) Errado

( ) Não sei

12. Na programação orientada a objetos, subprogramas (ou sub-rotinas) são encapsuladas nos próprios objetos e passam a designar-se:
( ) Atributo
( ) Herança
( ) Instância
( ) Método
( ) Encapsulamento
( ) Não sei

13. Na orientação a objetos, é um recurso que serve para inicializar os atributos e é executado automaticamente sempre que um novo objeto é criado:
( ) Método
( ) Polimorfismo
( ) Interface
( ) Classe
( ) Construtor
( ) Não sei

14. Objetos de software interagem e comunicam-se com os outros por meio de mensagens. Por exemplo, quando o objeto A deseja que o objeto B execute um de seus métodos, envia a este uma mensagem. Algumas vezes, o objeto receptor precisa de mais informação para que saiba exatamente o que deve fazer, de modo que essa informação seja transmitida juntamente com a mensagem por meio de parâmetros.
( ) Certo
( ) Errado
( ) Não sei

15. Um objeto apresenta três características básicas, o estado, a identidade e o comportamento. A parte de dados de um objeto é definida por um conjunto de mensagens, e a porção funcional, por um conjunto de atributos.
( ) Certo
( ) Errado
( ) Não sei

16. Um dos conceitos em programação orientada a objetos é o de abstração, por meio da qual as características do mundo real podem ser modeladas, por exemplo, mediante o agrupamento de objetos e classes.
( ) Certo
( ) Errado
( ) Não sei

17. Um objeto é, necessariamente, instância de apenas uma classe, mesmo quando existe herança múltipla em uma hierarquia de classes.
( ) Certo
( ) Errado
( ) Não sei

APÊNDICE B - Conteúdo programático da aula expositiva

Conteúdo programático

( ) 0. Apresentação do Professor
( ) 1. Abordagem histórica da origem da Orientação a Objetos
( ) 1.1. Crise do Software nos anos 70
( ) 1.2. Surgimento da Engenharia de Software com suas técnicas
( ) 1.3. Problemas de produtividade e qualidade persistiam nos anos 80
( ) 2. Realidade atual
( ) 3. Reutilização
( ) 3.1. Reutilização na computação
( ) 4. O que é orientação a objetos?
( ) 5. O que é um objeto?
( ) 6. O que são objetos computacionais?
( ) 7. Análise orientada a objetos
( ) 8. Programação orientada a objetos
( ) 9. O que é POO na prática?
( ) 10. História do POO
( ) 11. O que é paradigma?
( ) 12. Vantagens
( ) 13. Desvantagens
( ) 14. Conceitos básicos
( ) 14.1. Objetos
( ) 14.2. Classes
( ) 14.3. Abstração
( ) 14.4. Encapsulamento
( ) 14.5. Atributos
( ) 14.5.1. Exercício de identificação de atributos
( ) 14.6. Classe x Objeto
( ) 14.6.1. Exercício de Classe x Objeto
( ) 14.7. Métodos
( ) 14.8. Herança
( ) 14.9. Instanciação
( ) 14.10. Exercícios
( ) 14.11. Relacionamentos
( ) 14.12. Polimorfismo
( ) 15. Exercícios
( ) 16. Prática com o Scratch
( ) 17. Orientações de pesquisa

APÊNDICE C - Avaliação do aprendizado teórico - Programação Orientada a Objetos

Informe seu nome completo *

_____

Informe sua idade *

_____

Informe o dia em que você participou da aula *
( ) Participei da aula do dia 24/09 (quarta-feira)
( ) Participei da aula do dia 25/09 (quinta-feira)

1. Um atributo é uma propriedade dos objetos de uma classe. *
( ) Certo
( ) Errado
( ) Não sei

2. A herança, em programação orientada a objetos, é um relacionamento pelo qual uma classe herda todos os comportamentos e estados possíveis de uma classe ancestral. *
( ) Certo
( ) Errado
( ) Não sei

3. Marque a opção correta. Um objeto possui três características elementares, são elas: *
( ) A. Hierarquia, polimorfismo de sobrecarga, polimorfismo de sobreposição
( ) B. Método estático, variável e atributos
( ) C. Estado, comportamento e identidade
( ) D. Público, privado e friend
( ) E. Não sei

4. Os objetos podem ser iguais, apresentando as mesmas características e identificador. *
( ) Certo
( ) Errado
( ) Não sei

5. Na programação orientada a objetos, os métodos representam: *
( ) A. a implementação das ações das classes
( ) B. as associações estabelecidas entre as classes.
( ) C. o tipo de herança existente entre as classes.
( ) D. os tipos de linguagens de programação utilizados.
( ) E. não sei

6. O agrupamento dos objetos em uma classe ocorre quando eles possuírem as mesmas características e ações. *
( ) Certo
( ) Errado
( ) Não sei

7. Uma das maiores desvantagens na utilização da POO é a reutilização *
( ) Certo

( ) Errado
( ) Não sei

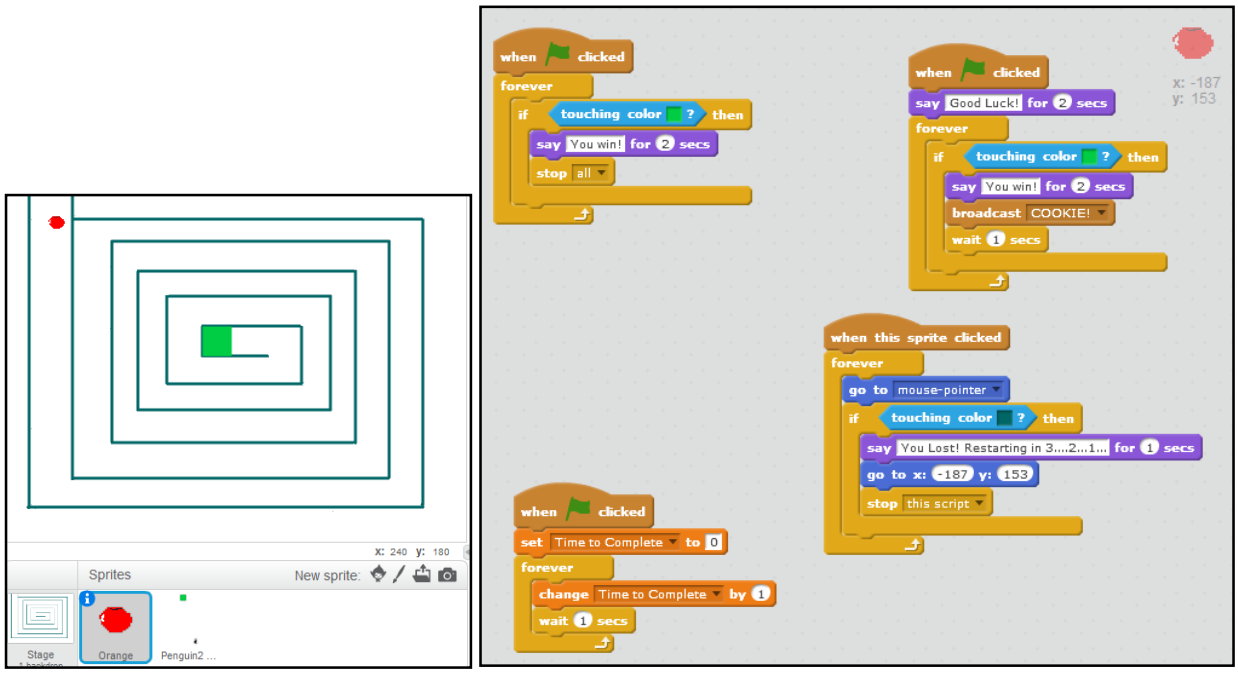
8. Nos conceitos de orientação a objetos, ___ é uma estrutura composta por ___ que descrevem suas propriedades e também por ___ que moldam seu comportamento. ___ são ___ dessa estrutura e só existem em tempo de execução. Para completar corretamente o texto, as lacunas devem ser preenchidas, respectivamente, por *

( ) A. objeto, métodos, assinaturas, Classes, cópias.
( ) B. polimorfismo, funções, métodos, Herança, cópias.
( ) C. classe, atributos, operações, Objetos, instâncias.
( ) D. multiplicidade, símbolos, números, Classes, herdeiros.
( ) E. domínio, diagramas, casos de caso, Diagramas de classe, exemplos.
( ) F. não sei

APÊNDICE D – Gabarito da avaliação do aprendizado teórico

1. Certo
2. Certo
3. C
4. Errado
5. A
6. Certo
7. Errado
8. C

APÊNDICE E – Jogo solicitado como avaliação de aprendizado prático na ferramenta *Scratch*



Cenário e comportamento desejado para o objeto "*Orange*"

APÊNDICE F – Critérios da avaliação de aprendizado prático

Criação dos objetos	Disco	Existe = sim	0,2
	Pista	Existe = sim	0,2
	Chegada	Existe = sim	0,1
Características dos objetos	Disco	Tamanho = ~17x13	0,2
		Cor = possuir cor diferente dos outros objetos	0,1
	Pista	Deve ser desenhado sobre o plano de fundo	0,1
		Cor = possuir cor diferente dos outros objetos	0,1
		Formato = deve ser viável para que o disco o percorra até o fim, com no mínimo 7 curvas	0,1
		Pista = deve ser fechada, impedido saídas, com exceção do início	0,1
		Grau de dificuldade = medido pelo número de curvas (Fácil = 7 curvas; Difícil = 15 curvas)	0,1
		Extra = pontos de dificuldade inseridos propositalmente	0,25
		Extra = contagem de tempo de jogo	0,5
		Extra = mudança de formato em tempo de execução	0,25
	Chegada	Tamanho = ~30x30	0,1
		Cor = possuir cor diferente dos outros objetos	0,1
Comportamento dos objetos	Disco	Deve ser posicionado no começo da pista ao iniciar o jogo	0,25
		Deve acompanhar o ponteiro do mouse sobre a pista	0,5
		Ao tocar na chegada, o disco deve ser paralisado	0,5
		Ao tocar na chegada, deve ser exibida uma mensagem informando o jogador que ele venceu o jogo	0,25
		Ao tocar em alguma extremidade da pista, o disco deve ser posicionado no começo da pista	0,75
		Ao tocar em alguma extremidade da pista, deve ser exibida uma mensagem informando o jogador que ele perdeu o jogo	0,25
Dinâmica do jogo	Jogo	O início do jogo deve ser disparado por um evento específico	0,2
		É possível mexer o disco de alguma forma?	0,2
		É possível percorrer a pista?	0,2
		É possível ganhar o jogo?	0,2
		É possível perder o jogo?	0,2
Lógica	Jogo	O volume de alterações para fazer o jogo funcionar é pequeno? (2 alterações, no máximo)	0,5