Uso de Dojos no ensino superior de computação

Carla Delgado[1], **Rodrigo de Toledo**[1], **Vanessa Braganholo**[2]

[1]Departamento de Ciência da Computação – Universidade Federal do Rio de Janeiro (UFRJ)

[2]Instituto de Computação – Universidade Federal Fluminense (UFF)

{carla,rtoledo}@dcc.ufrj.br, vanessa@ic.uff.br

Abstract. *The majority of Brazilian undergrad courses adopts classic didactic methods which were originally proposed in a time when information access and the social dynamics were very different from today. Nowadays, more participative dynamics are being used, as reported by lecturers that are experimenting new approaches in class, as for example* coding Dojo*, to teach programing. The practical and collaborative perspectives of the Dojo approach make it appealing for the computer science undergrad course. This paper discusses the points of convergence and divergence among* coding Dojo *and the needs of undergrad courses, and proposes adaptations to use* coding Dojo *in different Computer Science lectures.*

Resumo. *A maior parte dos cursos de graduação no Brasil adota métodos didáticos clássicos, originalmente propostos para uma situação onde o acesso à informação e as dinâmicas sociais eram bem distintos dos atuais. O uso de dinâmicas participativas tem crescido, e surgem cada vez mais relatos de professores que tentam utilizar abordagens diferentes em suas aulas, dentre elas o Dojo para o ensino de programação (*coding Dojo*). As vertentes prática e colaborativa dessa abordagem a tornam atrativa para adoção no ensino de graduação em computação. Este artigo discute os pontos de convergência e divergência entre a abordagem Dojo e as necessidades do ensino superior, e propõe adaptações para a sua utilização em diferentes disciplinas de cursos de graduação em computação.*

1. Introdução

A introdução da tecnologia como facilitadora da comunicação e do acesso à informação estabeleceu uma nova referência [Beniger 1986] e teve grande impacto na área educacional [Delgado et al. 2004]. Porém, o modelo da aula expositiva tradicional, adotado largamente no ensino superior no Brasil, não se beneficia da liberdade de acesso e de pensar proporcionada por esta nova realidade. Essa forma de ensino centraliza na figura do professor a gestão e condução do processo de ensino-aprendizagem, o que inibe a participação dos alunos e os induz a trabalhar isoladamente. Conforme observado por [Laurillard 2002], este tipo de aula não é interativa nem adaptativa, e não oferece tempo para acomodar as reflexões dos estudantes.

Outra mudança no panorama do ensino universitário advém de fatores econômicos. Pressões para aumentar os números de ingressos em um curto espaço de tempo acarretam em turmas cheias e heterogêneas. Alunos em diferentes estágios de conhecimento e diferentes ritmos de aprendizado não são o público ideal para uma aula

expositiva, pois a fala do professor é única para todos. Com turmas cheias, o professor não chega a conhecer os perfis de cada aluno, o que torna difícil motivá-los. De forma bem humorada, Laszlo e Castro [Lazlo and Castro 1995] documentam uma observação interessante: as aulas são a melhor forma de transferir o conhecimento das notas de aula do professor para os cadernos dos alunos, sem passar pela mente de nenhum deles.

Os atuais estudantes universitários pertencem a uma geração que desenvolveu com a tecnologia uma relação mais intensa que as gerações anteriores. Por sua vez, a área de computação é marcada por contínuas modificações e conhecimentos em distintos estágios de maturidade [Delgado et al. 2004]. Esses dois fatores geram um desafio aos professores do ensino superior em computação: compatibilizar suas práticas de ensino com os conhecimentos atuais da área e dialogar com estudantes cuja alfabetização tecnológica foi muito diferente da sua. Os professores que ignoram esses desafios se distanciam de seus alunos, ministrando aulas que, por estarem dessincronizadas de sua audiência, não despertam interesse.

Buscando melhorias na dinâmica das aulas presenciais e procurando compatibilizá-las com os interesses e o grau de alfabetização tecnológica dos alunos de cursos de graduação de computação, os autores deste artigo utilizaram adaptações do método de codificação Dojo nas aulas de diferentes disciplinas. Dojos têm sido usados com sucesso por comunidades de programadores, com o objetivo de melhorar as habilidades de programação dos participantes. Dentre as vantagens do Dojo estão: a velocidade de ensino ajustada à capacidade de absorção; teoria puxada pela prática; discussões feitas em torno de um código que compila e não sobre texto no quadro negro. No entanto, para que possa ser usado com sucesso no ambiente de ensino de graduação, o Dojo precisa ser instanciado nesse contexto. Contribuímos com uma discussão detalhada sobre os pontos de convergência e divergência entre a filosofia Dojo e o ambiente do ensino universitário, e apresentamos nossas sugestões de adaptação. Essas sugestões são baseadas na experiência dos autores na aplicação de Dojos de programação nas disciplinas de Estruturas de Dados, Inteligência Artificial e Computação Gráfica, além de levar em conta relatos de colegas que aplicam Dojos em suas disciplinas.

O restante deste artigo está estruturado como segue. A Seção 2 apresenta o conceito de Dojos de programação. A Seção 3 discute trabalhos relacionados. A Seção 4 apresenta a discussão sobre os obstáculos para a adaptação de Dojos ao ambiente de ensino, e apresenta sugestões de contorno para os obstáculos identificados. Finalmente, a seção 5 conclui o trabalho.

2. Dojo

O termo *coding Dojo*, ou simplesmente Dojo, vem sendo utilizado para denotar uma reunião de pessoas com o propósito de resolver um problema [Sato et al. 2008] através da codificação de um programa de computador. A filosofia do Dojo está muito ligada à Aprendizagem Baseada em Problemas (*Problem Based Learning*) [Wood 2003] e à programação orientada a testes (*Test Driven Development*) [Beck 2003].

Normalmente, uma sessão de Dojo tem duração de cerca de duas horas, e requer uma sala para acolher os participantes, um computador e um projetor. O encontro ocorre de acordo com um processo bem definido, discutido a seguir.

- Acerto da data da próxima reunião;

- Discussão para decidir o foco dessa sessão, ou seja, o problema a ser resolvido;
- Escolha da linguagem de programação a ser usada;
- Ciclo de codificação (em pares);
- Discussão de fechamento ou Retrospectiva.

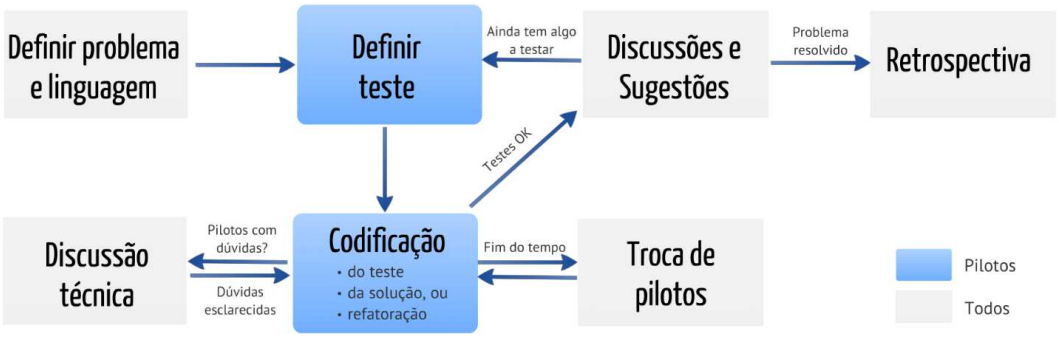

Figura 1. Ciclo de codificação do Dojo.

Existem duas variações principais deste processo [Sato et al. 2008]. Descrevemos aqui a mais comum, também conhecida como *RandoriKata*. Vale ressaltar que mesmo considerando especificamente a variação *RandoriKata* não há um consenso quanto ao ciclo de codificação. A figura 1 ilustra o que adotamos neste artigo.

O ciclo de codificação (Figura 1) é feito usando práticas de programação ágil [Teles 2004]: programação em pares e orientada a testes. Tudo começa com a escolha de um piloto e um co-piloto. Ambos devem sentar-se de frente para a plateia, que vê o que está sendo feito no computador através da tela do projetor. O piloto fica responsável por operar o computador, escrevendo código de acordo com o discutido com o co-piloto. A primeira coisa a ser feita é escrever um teste. Enquanto o código escrito até o momento não passa no teste, todos na plateia devem ficar em silêncio – é como se houvesse um sinal vermelho para a plateia. Apenas o piloto e o co-piloto devem falar, e explicar à plateia como eles pretendem fazer o teste passar. De tempos em tempos (por exemplo, a cada cinco minutos), o co-piloto assume como piloto, o piloto volta para a plateia, e alguém da plateia assume como co-piloto. Quando o código passa no teste, é como se um sinal ficasse verde, e a plateia pode se manifestar, dando sugestões para melhorar o código. Após o término das sugestões, o piloto escreve novo teste, e o processo se repete.

A plateia em uma sessão de Dojo é voluntária. Está ali porque quer participar. Normalmente todos da plateia participam pelo menos uma vez como piloto. O ambiente costuma ser amigável, e erros e inseguranças não são valorizados. No entanto, esse ambiente não costuma ser o ambiente de uma sala de aula. Como adaptar um Dojo para que possa ser usado em sala de aula? Tentaremos responder a essa questão na seção 4.

3. Trabalhos Relacionados

Algumas dissertações e projetos de fim de curso discutem o uso de Dojos como ferramenta de ensino em cursos superiores [Fontes 2011, Cukier 2009, Bravo 2011].

A pesquisa do projeto de final de curso de Fontes [Fontes 2011] defende a necessidade de melhorar o ensino de programação, e aponta o uso de Dojos como possível solução para vários aspectos necessários a essa melhoria. A dissertação de mestrado de

Bravo [Bravo 2011] investiga duas formas diferentes de ensinar práticas ágeis aos alunos: o uso de DOJOs de programação e o uso de ferramentas de observação, onde um grupo trabalha e os demais observam. A autora destaca que ainda são necessárias avaliações mais completas sobre o uso de Dojos no ensino, de forma a mensurar melhor o nível de aprendizado atingido pelos alunos. Por fim, o trabalho de Cukier [Cukier 2009] investiga padrões utilizados na introdução de novas ideias na indústria de software, dentre eles o Dojo. O autor enfatisa o feedback entre os participantes como mecanismo fundamental para o aprendizado. Porém, o foco do estudo é o ambiente empresarial e não o acadêmico. Assim, a questão de como adaptar Dojos para o ensino universitário permanece em aberto.

4. Uso de Dojo no ensino de graduação

A internet está repleta de depoimentos de professores que têm tentado aplicar Dojo em sala de aula, como forma de estimular o aprendizado dos alunos[1,2,3]. De fato, o uso de DOJO no ensino pode significar uma mudança no paradigma de aprendizado: saímos de um cenário passivo, onde o aluno apenas absorve o conhecimento, para um cenário mais ativo, onde o aluno participa e aprende com os erros e acertos dos colegas.

No entanto, não é possível aplicar o Dojo em sua forma tradicional à uma aula de graduação. Algumas características do programa da disciplina e tamanho da turma limitam a flexibilidade oferecida pelos Dojos. Além disso, o foco original do Dojo é o aumento da habilidade em programação, enquanto que na maioria das disciplinas de graduação em computação a programação é instrumento e não objetivo.

Os autores deste artigo aplicaram Dojos em suas disciplinas, e mantêm contato com outros professores que fazem ou já fizeram o mesmo. Diante dessa experiência nas disciplinas de Estrutura de Dados, Inteligência Artificial e Computação Gráfica, conseguiram reunir um conjunto de adaptações ao Dojo para que ele possa ser aplicado a qualquer disciplina de graduação. Esse conjunto está refletido na Figura 2 e é detalhado a seguir.

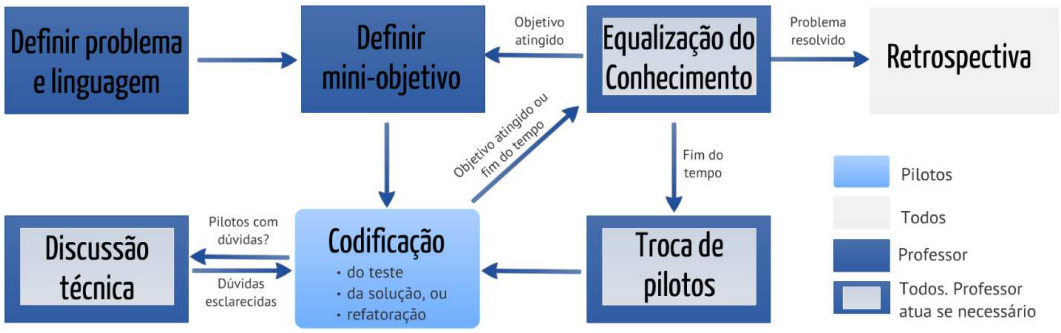

Figura 2. Ciclo de cofificação Dojo incluindo *baby goals*.

4.1. Escolha do problema a ser resolvido

Em uma disciplina de programação de computadores, não há limitação em relação ao problema a ser resolvido em uma sessão de Dojo. No entanto, determinados problemas

[1]UFSC (`http://pet.inf.ufsc.br/dojo/`)
[2]IFPB (`http://dojorio.wordpress.com/category/dojocampos/page/2/`)
[3]UNIFOR (`http://www.javace.org/i-coding-dojo-javace/`), entre outros.

podem exigir o uso de estruturas que ainda não foram vistas em aula. Dessa forma, o ideal é que o professor assuma o papel de propor o tema, e escolha um problema que possa ser resolvido usando o que os alunos já sabem e o próximo conteúdo ou habilidade a ser aprendido. Em outras disciplinas, os alunos já dominam a programação, mas não é qualquer problema que está no escopo da disciplina. O professor deve então encarregar-se de propor um problema que esteja dentro do programa da disciplina, impingindo a ordem desejada para o encadeamento dos conteúdos do programa. Numa abordagem intermediária, o professor pode propor diferentes problemas cujas soluções envolvem o aprendizado que se deseja adquirir naquele momento, permitindo que os participantes façam a sua escolha.

Outro aspecto que pode ser explorado com o Dojo é a comparação de métodos. Um mesmo problema pode ser atacado por diferentes métodos, o que fará com que os alunos, além de praticar os métodos, comparem sua eficiência e seus resultados. O fornecimento de perspectivas diferentes sobre a resolução de um mesmo problema serve para instigar a curiosidade sobre quais propriedades levam às diferenças observadas. Essa prática trabalha simultâneamente o senso crítico, critérios de avaliação da área de computação e a capacidade de interpretar resultados. Abaixo seguem alguns exemplos de problemas por disciplinas de cursos de graduação em computação:

Em uma disciplina de **estruturas de dados**, o professor pode propor como problema "encontrar um valor X dentro de uma lista encadeada". Esse problema permite várias abordagens e é natural que as mais simples sejam sugeridas pelos alunos. O professor pode conduzir o melhoramento das abordagens, induzindo métodos mais eficientes de busca. Os métodos devem ser então comparados, com o estudo da complexidade algoritmica de cada um.

Em uma disciplina de **inteligência artificial**, pode-se propor o problema da coloração de mapas para trabalhar algoritmos de busca em amplitude e busca em profundidade. A comparação dos métodos evidencia as propriedades do problema que tornam a busca em profundidade mais apropriada neste caso. A busca em profundidade limitada pode suceder da análise de características do método da busca em profundidade aplicada a esse problema, dada a observação de que a profundidade na árvore de busca em que uma solução pode ser encontrada é previsível. Para trabalhar o tema de busca local (algoritmo de *hill climbing*, por exemplo) pode ser utilizado o problema da geração de estados iniciais para o conhecido jogo "campo minado". Por ser um problema do universo de interesse dos alunos, ao testarmos a aplicação deste problema em sala de aula, a participação e o interesse na atividade foi bem superior à média. As funções de otimização sugeridas pelos alunos foram surpreendentes.

Numa disciplina de **computação gráfica**, existe um encadeamento não linear dos assuntos da ementa. Ou seja, mesmo que os temas sejam escolhidos numa sequência lógica de aprendizagem, há uma possiblidade de que alguns dos assuntos previstos para a disciplina nunca sejam abordados. Para isso, cabe ao professor a escolha do próximo problema, visando cobrir os assuntos previstos. Por exemplo, se em um Dojo o tema foi curvas de Bézier (2D), o assunto do próximo Dojo poderá ser superfícies de Bézier (3D) ou uma outra representação geométrica 2D, como a *half-edge*.

4.2. Escolha da linguagem de programação

É comum, em sessões de Dojo, os participantes alternarem entre diversas linguagens de programação. Porém isso não é adequado para algumas disciplinas de graduação, já que as disciplinas normalmente têm uma linguagem alvo. Por exemplo, em disciplinas de programação de computadores, existe uma linguagem que precisa ser ensinada (C, por exemplo), e o foco de todas as aulas deve estar nessa linguagem.

Em disciplinas que não possuem restrição em relação à linguagem, isso pode ser flexibilizado. No entanto, focar em uma linguagem é um recurso de que o professor pode lançar mão para trabalhar habilidades específicas. Por exemplo, em uma disciplina de **estruturas de dados**, é importante que o aluno exercite a manipulação de ponteiros para manipular uma lista encadeada, uma árvore ou um grafo. Linguagens que escondem esses recursos, como Java, serão menos adequadas para esta prática. Em contrapartida, as propriedades de uma estrutura de dados ficam mais evidentes quando a estrutura encapsula os detalhes técnicos. Assim, uma segunda implementação da mesma estrutura ou de uma estrutura similar em uma linguagem orientada à objetos daria outra perspectiva ao mesmo conceito.

Técnicas de **inteligência artificial** e **computação gráfica** podem se beneficiar muito das características de uma ou outra linguagem. O uso de uma linguagem que possua bibliotecas específicas é adequado nestes cursos, pois alguns algoritmos são complexos e a existência de implementações prontas para os algoritmos básicos é um adianto.

4.3. Codificação dos testes

Depois de escolhido o problema, piloto e co-piloto começam pela codificação de um teste unitário. Enquanto o código escrito não passar no teste, a plateia não pode opinar. A escrita dos testes é um exercício importante, pois faz o aluno pensar em diversas condições que o código tem que satisfazer. Para exemplificar, suponha que o problema escolhido foi buscar um valor X em uma lista encadeada. Um teste é a busca em uma lista vazia, outro a busca de um elemento em uma lista onde só há esse elemento, outro a busca de um elemento em uma lista não vazia que não contém esse elemento. Esses são alguns exemplos de testes, mas, para garantir a funcionalidade desejada, outros são necessários.

O delineamento dos casos de teste nem sempre é simples. Em algumas disciplinas, os testes podem ser demasiadamente complicados de construir, e neste caso, deveriam ser codificados de antemão pelo professor. Um exemplo é a manipulação arquivos estudada em disciplinas de **estrutura de dados**. Suponha que o problema escolhido foi inserção em árvores B+ armazenadas em disco. Para codificar um teste, é necessário gerar o arquivo de entrada, que será usado pelo método de inserção implementado pelos alunos, e o arquivo de gabarito, que será comparado com o arquivo gerado após a inserção. Essa codificação não é trivial, e, se deixada a cargo dos alunos, pode consumir uma sessão inteira de Dojo.

Ao estudar técnicas para problemas de satisfação de restrições, típico de disciplinas introdutórias de **inteligência artificial**, o professor também já deve trazer pronta uma função que teste se uma solução satisfaz às restrições do problema. Esta função pode inclusive ser revisada junto com os alunos, e consiste por si só em um recurso didático para observar os aspectos do problema a ser atacado, antes da exploração, durante a seção de Dojo, das técnicas de resolução dessa classe de problemas.

4.4. Miniobjetivos (*baby goals*)

Programar com testes automatizados é algo que deve ser sempre estimulado e, de fato, é um padrão de qualidade que deveria ser exigido cotidianamente. Porém, como dito na subseção anterior, exigir que o código de teste seja programado durante a execução de um Dojo pode não ser interessante. Na seção 2, explicamos o uso dos testes como sinalizador para a participação da plateia. Em alguns casos, nós propomos a substituição de testes como sinalizadores por miniobjetivos (*baby-goals*).

Para que a solução do problema do Dojo seja atingida, o professor pode propor uma sequência de miniobjetivos que levam ao resultado maior. Por exemplo, num Dojo de **computação gráfica**, o desafio era desenhar uma curva de Bezier 2D, dados quatro pontos de controle. Os miniobjetivos eram:

1. Plotar um ponto na tela.
2. Plotar dois pontos e o segmento de reta ligando ambos.
3. Interpolar linearmente a reta como uma sequência de 10 pequenos segmentos.
4. Plotar um terceiro ponto e desenhar outro segmento com o procedimento anterior.
5. Desenhar a parábola que interpola linearmente as duas interpolações.
6. Plotar um quarto ponto, desenhando uma segunda parábola.
7. Desenhar a Bézier, interpolando linearmente as parábolas.

Nos exemplos acima, a cada conquista era gerado um sinal verde, que permitia a participação da plateia. Ao começar o desenvolvimento de um novo miniobjetivo, o sinal se transformava em vermelho, onde a participação da plateia só poderia acontecer no estouro do cronômetro ou sob demanda dos pilotos. Sugerimos que esses miniobjetivos fiquem expostos no quadro para que todos percebam o avanço na conquista da solução ao problema do Dojo.

4.5. Equalização do Conhecimento

Devido a heterogenidade de uma turma de graduação, é importante que haja uma equalização do conhecimento de todos. No contexto do ensino em universidade, nós sugerimos que isso aconteça inclusive no estouro do cronômetro (vide ”Equalização do Conhecimento” na Figura 2). É provável que haja um conjunto de alunos cujo conhecimento esteja inferior à produção da dupla de pilotos assim como um conjunto de alunos com conhecimento superior. A equalização deve acontecer segundo os seguintes passos.

(1) Dúvidas? Nesse ponto equilibra-se o grupo de conhecimento inferior com a dupla mais recente.
(2) Sugestões? Nesse ponto equilibram-se todos com o grupo de conhecimento superior.
(3) Sugestões do Professor. O professor só deve sugerir algo se estiver observando um desvio no objetivo que não foi observado pela plateia.

Para que essa sequência seja cumprida, é necessário prudência por parte do professor para que ele não faça sugestões sem passar pelos passos (1) e (2).

4.6. Troca dos pilotos

A participação voluntária é um dos grandes segredos do sucesso dos Dojos. No entanto, sua aplicação ao cenário de Dojos em disciplinas de graduação deve ser diferente. Muitas vezes o aluno não está na aula por vontade própria, mas porque é obrigado a cumprir uma

carga horária mínima. Existe também o problema das turmas excessivamente cheias que afeta vários cursos de graduação atualmente. Há ainda um agravante quanto a maturidade, pois em média, os alunos de graduação são imaturos quando comparados aos profissionais em empresas. Dessa forma, a questão de como conduzir a participação dos alunos nas sessões de Dojos é crucial. Em nossa experiência, já vimos várias práticas para lidar com a resistência dos alunos em participar. Detalhamos algumas delas a seguir:

Uma opção é oferecer algum tipo de **bônus** aos alunos que participam. Pode ser, por exemplo, um bônus na média ou na nota da prova, caso o aluno participe de um determinado número de sessões de Dojo como piloto. Isso normalmente funciona bem.

Outra alternativa é **obrigar todos os alunos a participarem** como pilotos, impondo algum tipo de penalidade para os que não participam. De acordo com nossa experiência, essa alternativa é **catastrófica**, pois gera dois comportamentos por parte dos alunos: (i) eles faltam à aula quando sabem que haverá Dojo; ou (ii) eles vão à aula, participam como pilotos, mas ficam apenas "passeando pelo código" até o tempo deles acabar.

A prática de **listar nomes** das pessoas no quadro a medida que elas vão participando é também uma forma de motivação. As pessoas vão observando o aumento da lista e se questionam se não deveriam participar, criando uma pressão implícita pois os professores e colegas podem reparar quais alunos ainda não participaram. Essa prática também é útil para a comunicação e para relembrar quais foram os pares formados.

Uma outra prática interessante é a **participação do professor ou monitor** como um piloto ou co-piloto. Essa tática pode ser usada para quando há uma demora muito grande para alguém da plateia se voluntariar, ou quando, por algum motivo, a implementação está precisando de uma mudança (ou *refactoring*) maior. O resultado é psicologicamente interessante, pois humaniza o professor, trazendo-o à condição de um participante como os demais. Às vezes é possível adotar essa prática na primeira rodada do Dojo, ajudando a transpassar alguma barreira de ordem tecnológica (ex: uma IDE nova ou uma chamada a uma API desconhecida).

Seja como for, é notório que uma parcela da turma se entusiasma e participa mesmo que não seja oferecido nenhum bônus ou ônus. No entanto, uma outra parcela razoável da turma tem resistências em participar, seja por timidez, insegurança ou medo de errar na frente dos colegas. Instigar e valorizar a participação da audiência é uma forma de remediar isso, fazendo com que mesmo os alunos que não estão pilotando ou co-pilotando a máquina sejam envolvidos no processo e acompanhem o processo de construção de conhecimentos de desenvolvimento de habilidades que a seção de Dojo suporta.

4.7. Comunicação durante a codificação

A explicação do piloto é fundamental para que a plateia entenda o que está sendo feito. No entanto, alguns alunos costumam ter dificuldades nessa parte. Ocorre que uma parcela considerável dos estudantes de graduação tem problemas em falar em público, e essa dificuldade torna-se aparente durante os Dojos. Outros falam, mas por timidez, falam muito baixo, e a turma não consegue escutar o que está sendo dito. Temos notícias de tentativas frustradas de tentar resolver esse problema. Um professor propôs o uso de um microfone: o piloto deveria usar o microfone ao falar. Isso só fez a situação piorar:

os alunos então apenas balbuciavam palavras ininteligíveis, tentando evitar ao máximo a captação do som pelo microfone.

Uma alternativa é a intervenção, nessas situações, do professor. Ao detectar que o piloto não está explicando, o professor deve insistir para que o aluno explique o que foi feito. Caso o resultado seja demasiado conciso, o professor deve perguntar por mais detalhes. Mesmo se o aluno explicar em voz baixa, o professor pode repetir a informação em um tom audível pelo restante da turma.

Uma boa prática é distanciar os pilotos. Por exemplo, o co-piloto pode ficar próximo à tela projetada enquanto o piloto digita de costas para a plateia no computador. Separados por alguns metros, o diálogo entre eles se torna mais audível pois eles devem falar mais alto do que se estivessem apenas lado-a-lado. O co-piloto pode apontar para a tela de projeção e o piloto pode usar o mouse quando quiser apontar algo para o co-piloto. Nessa prática, há também um outro efeito positivo pois o que está sendo apontado fica visível pela plateia, deixando de ser conhecimento exclusivo entre os pilotos.

4.8. Ajuste do tempo no Dojo

Usualmente, o Dojo é descrito de forma que o par piloto/co-piloto permaneça programando por 5 minutos. Porém, esse tempo pode ser ajustado. Empiricamente, percebemos que os seguintes fatores contribuem para esse ajuste:

- Se a plateia é grande, o tempo deve ser reduzido, de modo que aumente o percentual de pessoas que participarão da pilotagem. Além disso, reduz o risco de dispersão, que inevitavelmente é maior com um grande número de alunos.
- Se a plateia é inexperiente, o tempo também deve ser reduzido, pois corre-se o risco de uma sessão inteira estar caminhando na direção errada!
- Se a complexidade do problema é grande, com uma plateia experiente, o tempo pode ser aumentado, de modo que seja possível atingir um miniobjetivo pelo par de pilotos em atuação.

Vale observar que o mais importante para isso é a experimentação do professor e ajustes de acordo com o resultado. Os autores deste artigo experimentaram com tempos entre 3 e 5 minutos validando-os em cada contexto.

Como visto, existem várias adaptações e obstáculos relativos à adoção de Dojos no ensino. A pergunta que fica é: quão efetivo isso é do ponto de vista do aluno? Em um trabalho relacionado [Carmo and Braganholo 2012], conduzimos um survey com alunos de uma disciplina onde estava sendo aplicado DOJO, ao longo de um semestre inteiro. Os resultados mostram que o Dojo é efetivo, mas que precisa de adaptações para que funcione a contento. Relatamos nesta seção as adaptações que propusemos e aplicamos ao ministrar disciplinas de cursos de graduação em computação.

5. Considerações Finais e Trabalhos Futuros

O avanço da tecnologia e fatores econômicos que pressionam por turmas cada vez mais cheias têm grande impacto no ensino universitário. Para se adequar a isso, os professores vêm propondo novas formas de interação dos alunos com o conteúdo das disciplinas. Uma dessas formas é o uso de Dojos de programação. Os Dojos se adequam bem ao ensino de

programação porém algumas adaptações são necessárias para que ele possa ser utilizado em outras disciplinas de cursos de computação.

As sugestões apresentadas nesse trabalho são baseadas em adaptações já empregadas pelos autores deste artigo. Os autores consideram que tais adaptações são efetivas, porém ainda estão em estágio de maturação. Superar a timidez dos alunos ainda é um desafio.

Elaborar os problemas que serão propostos aos alunos é o maior investimento extra-classe necessário à adaptação do material didático para a utilização de Dojos. Como trabalho futuro, pretendemos compilar problemas para sessões de Dojo adequados ao conteúdo de algumas disciplinas de cursos de graduação em computação. Esperamos que outros professores sejam encorajados a adotar Dojos em suas aulas, e que estes possam contribuir para o refinamento dessa proposta.

Referências

Beck, K. (2003). *Test-Driven Development: By Example*. The Addison-Wesley Signature Series. Addison-Wesley.

Beniger, J. R. (1986). *The control revolution: technological and economic origins of the information society*. Harvard University Press, Cambridge, MA, USA.

Bravo, M. (2011). Abordagens para o ensino de práticas de programação extrema. Dissertação de Mestrado, Universidade de São Paulo, São Paulo, SP.

Carmo, D. and Braganholo, V. (2012). Um estudo sobre o uso didático de dojos de programação. In SBC, editor, *Workshop de Educação em Computação (WEI)*. Sociedade Brasileira de Computação.

Cukier, D. (2009). Padrões para introduzir novas ideias na indústria de software. Dissertação de Mestrado, Universidade de São Paulo, São Paulo, SP.

Delgado, C., Xexéo, J. A. M., Souza, I. F., Campos, M., and Rapkiewicz, C. E. (2004). Uma abordagem pedagógica para a iniciação ao estudo de algoritmos. In SBC, editor, *Anais do XII Workshop de Educação em Computação (WEI)*. Sociedade Brasileira de Computação.

Fontes, B. (2011). Coding dojo: novas possibilidades para o ensino de programação. Projeto de Diplomação, Universidade Federal Fluminense, Niteroi, RJ.

Laurillard, D. (2002). *Rethinking University Teaching: A Conversational Framework for the Effective Use of Learning Technologies*. Taylor and Francis.

Lazlo, A. and Castro, K. (1995). Technology and values: Interactive learning environments for future generations. *Educational Technology*, 35:7–13.

Sato, D. T., Corbucci, H., and Bravo, M. V. (2008). Coding dojo: An environment for learning and sharing agile practices. In *Proceedings of the Agile 2008*, AGILE '08, pages 459–464, Washington, DC, USA. IEEE Computer Society.

Teles, V. M. (2004). *Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade*. Novatec Editora.

Wood, D. (2003). Abc of learning and teaching in medicine: problem based learning. *BMJ*, 326(7384):328–330.