UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

BERNARDO BOTELHO FONTES

CODING DOJO: NOVAS POSSIBILIDADES PARA O ENSINO DE PROGRAMAÇÃO

Niterói
2011

BERNARDO BOTELHO FONTES

CODING DOJO: NOVAS POSSIBILIDADES PARA O ENSINO DE PROGRAMAÇÃO

Monografia apresentada ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Orientador: Prof. D. Sc. ISABEL LEITE CAFEZEIRO

Niterói

2011

CODING DOJO: NOVAS POSSIBILIDADES PARA
O ENSINO DE PROGRAMAÇÃO

Monografia apresentada ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Aprovada em Dezembro de 2011.

BANCA EXAMINADORA

Prof. D.Sc. ISABEL LEITE CAFEZEIRO – Orientador
UFF

Prof. D.Sc. LEONARDO GRESTA PAULINO MURTA
UFF

Prof. M.Sc. e Ph.D. IVAN DA COSTA MARQUES
UFRJ

Niterói
2011

AGRADECIMENTOS

À minha avó, Dona Adélia, por me ajudar durante esses cinco anos morando sozinho me alimentando com os seus deliciosos quitutes congelados;

Ao meu pai, Henrique, por ser um exemplo e, acima de tudo, um ídolo para mim;

À minha mãe, Marcia, por sempre se preocupar, me aconselhar e me dar tudo o que precisei para me tornar o profissional que sou hoje;

Aos meus irmãos, Eduardo e Thiago, pelos anos juntos, histórias divididas no apartamento e exemplo de união;

À minha namorada, Paula, que ainda que tão longe manteve-se perto sempre me ajudando a olhar para frente;

A toda a minha família sempre presente na minha vida;

Aos amigos da faculdade, especialmente o João Luis e o Gabriel, irmãos para mim, por esses anos em que, juntos, dividimos sucessos e fracassos;

Aos amigos de Friburgo, especialmente o Wanderson, o Fernando e o Luis Felipe, por todas as histórias que tivemos e por, mesmo depois de sair da cidade, continuarmos juntos;

Aos amigos Flávio, Thiago, Henrique e Luciano, companheiros na Dekode e que deram a motivação final para esse projeto;

Aos novos amigos Gerardo, Elias, João Felipe e Luisão, que acolheram o Coding Dojo da UFF e hoje seguem com ele que

A todas as pessoas que já participaram do Coding Dojo da UFF, este que por várias vezes enxerguei como um filho;

À todos os que já estiveram no Pós-Dojo da UFF pelas excelentes conversas e noites divertidas;

Às comunidades do Rio de Janeiro e, principalmente, à DojoRio porque se não fossem vocês nada do que escrevi existiria;

Por último, ao Álvaro, ao Henrique e ao Vinícius. Obrigado pelas pílulas vermelhas.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

RESUMO

Fomentar o ensino da programação é um ato importante para uma época em que a demanda por profissionais e pesquisadores na área vem aumentando continuamente. Entretanto, temos evidências de que o ensino que vem sendo aplicado hoje não vem alcançando resultados satisfatórios. Um indicador disso é o número de artigos publicados que abordam esse tema.

Existem várias ferramentas e metodologias utilizadas pelos professores da área de programação para trabalhar com seus alunos, visando potencializar a maneira de educarem. Uma nova metodologia que vem crescendo e pode ser utilizada é a prática do Coding Dojo, um encontro da experiência coletiva com o aprendizado individual.

O objetivo deste trabalho é abordar o Coding Dojo e entendê-lo como uma ferramenta auxiliar para o ensino de programação.

Palavras-chave: coding dojo, ensino, programação, didática.

ABSTRACT

Encourage the teaching of programming is an important act for a time when the demand for professionals and researchers in the area is continually growing. However, we have evidence that the teaching that is being applied today has not achieved satisfactory results. One indicator of this is the number of published articles that address this issue.

There are several tools and methodologies used by professors in the area of programming to work with their students in order to enhance the way we educate. A new methodology that has been growing and can be used is the practice of Coding Dojo, a meeting of the collective experience with the individual learning.

The objective of this study is to approach the Coding Dojo and understand it as an auxiliary tool for teaching programming.

Keywords: coding dojo, teaching, programming, teaching.

Capítulo 1

INTRODUÇÃO

Vivemos em uma época em que cada vez mais lidamos com processos automatizados por máquinas, interagimos com sistemas computacionais projetados para atender necessidades específicas e interage-se cotidianamente com pelo menos um aparelho que tenha alguma computação envolvida. É perceptível uma demanda crescente por pessoas que dominem a arte da programação e a saibam praticar com maestria. Entretanto, como chegar a tal grau de habilidade?

Sendo as aulas de programação uma das primeiras portas de entradas para esse mundo, é coerente expor as pessoas a práticas e metodologias que exponenciem a possibilidade de aprendizado e propagação de conhecimento. Além disso, a formação inicial é essencial para a construção de uma base sólida para um desenvolvedor de software ou para um pesquisador da área.

Entretanto, mesmo com essa importância, temos hoje o ensino de programação preso ao modelo de ensino antigo, limitando os alunos a um professor, um quadro e uma sala de aula. Numa tentativa de escapar dessa configuração, por muitas vezes são apresentadas tecnologias, como computadores ou aulas em laboratórios, mas que são utilizadas apenas como substitutas ao quadro e não geram, necessariamente, melhorias no aprendizado. Tratar o ensino de uma área tão nova e tão rica de uma maneira arcaica é, além de paradoxal, prejudicial ao aluno.

Proveniente das práticas de metodologias ágeis, o Coding Dojo pode ser utilizado dentro da sala de aula como uma maneira de dinamizar a troca de conhecimento entre o aluno e professor e, também, entre os próprios alunos. Trata-se de subverter um pouco o modelo de ensino tradicional e adicionar a ele mais uma ferramenta que visa somente aperfeiçoar o aprendizado através da prática da programação conjunta.

1.1 OBJETIVO

O objetivo deste trabalho é apresentar o que é Coding Dojo, sua história, os seus pontos pedagógicos e filosóficos, o seu processo e componentes, sempre focado em como utilizá-lo como uma ferramenta de ensino. Entretanto, a apresentação do mesmo não basta.

Pretende-se também abordar as consequências reais que a prática de Coding Dojo pelos alunos do Curso de Ciência da Computação trouxe para suas vidas, para a comunidade e para a própria faculdade e professores.

Capítulo 2

A BASE CONCEITUAL

Existem muitos conceitos que são inerentes ao processo de uma sessão de Coding Dojo, mas que já foram estudados anteriormente por filósofos, pensadores e pesquisadores. Ter a ciência desses conceitos é importante para tornar mais fácil a contextualização de certas decisões dentro da dinâmica do processo. Esse capítulo tem o intuito de apresentar os pensamentos e práticas que estão ligadas ao processo de aprendizado e disseminação do Coding Dojo, já que estes são os focos desta monografia.

2.1 CONSTRUTIVISMO

Ao estudar a formação da palavra “construtivismo”, podemos enxergar que ela tem origem derivada do verbo *struere*, em latim, que significa organizar, criar estrutura. Olhando tal significado, podemos entender que é um movimento que possui em seus princípios a construção, não a formação. A diferença nesse ponto se dá ao passo que o ato de construir exige a ação de uma pessoa para gerenciar a organização porque o processo de construção é um processo de lidar com o novo. A formação, por sua vez, como a própria palavra já traz, já possui uma forma, ou seja, uma organização pré-estabelecida (CASTAÑON, 2009).

Para tratar o tema, trataremos a visão Jean Piaget, criador do termo construtivismo. A sua visão aproxima-se muito da maneira de enxergar o relacionamento das pessoas com a informação durante uma sessão de Coding Dojo.

Jean Piaget foi um filósofo suíço que viveu de 1896 a 1980 e teve como principal trabalho o desenvolvimento da sua teoria da *Epistemologia Genética*. PIAGET (1973), através da *Epistemologia Genética*, tenta abordar de maneira científica a teoria do conhecimento. O caminho que ele escolhe é o de compreender como as estruturas cognitivas do sujeito são construídas através do tempo.

A grande pergunta que Piaget tenta responder é a de como o conhecimento é construído. Para essa pergunta ele propõe a sua resposta construtivista. Piaget defende que a construção do conhecimento se dá na interação entre o sujeito que conhece e o objeto

conhecido. Ao interagir com o objeto digno do conhecimento, o sujeito constrói, a partir de suas ações, a suas próprias representações de mundo (ROSA, 2010, p. 5).

Foi seguindo esse pensamento que Piaget conseguiu construir um modelo de desenvolvimento cognitivo construtivista em que apresenta o sujeito como principal responsável pela construção de suas próprias estruturas cognitivas ao longo do tempo. Para explicar esse processo, ele faz uso dos conceitos de *assimilação* e *acomodação* (CASTAÑON, 2009).

A assimilação tem base no encontro do sujeito com um objeto em um cenário novo. Ao ver-se nessa situação uma pessoa, por exemplo, tende a encaixar o cenário com que está interagindo dentro de suas experiências anteriores. Mas, como é algo novo, ela percebe que aquele cenário não se encaixa dentro das estruturas cognitivas que ela já havia construído com as suas percepções anteriores. Nesse ponto, a estrutura pessoal tende a se moldar e se acomodar para uma nova configuração em que esse cenário consiga se encaixar. Portanto, enxergamos aqui que a soma entre as experiências vividas anteriormente e o novo cenário que implica em uma nova experiência é essencial ao mecanismo de construção do conhecimento.

CASTAÑON (2009) Diz que o principal a ser considerado é o conhecimento construído através da interação com o novo. Conviver em um cenário padrão não posiciona a pessoa no processo de assimilação e acomodação proposto por Piaget. Podemos dizer que o processo automatizado de ensino derivado da repetição alimenta uma falsa sensação de maestria no assunto discutido. Deve ser valorizado o contato direto e frequente com o novo. Do resultado do impacto com o que é novo é que surge o alimento para a construção de um conhecimento individual mais sólido e mais adaptável.

Piaget traz o conceito de o ambiente ser adaptado dentro da estrutura cognitiva individual. Ao falar disso, é importante ressaltar que o ambiente é resultado da construção da interação entre vários atores quando falamos do ambiente da sala de aula. Nesta abordagem entendemos como atores não somente professores e alunos, como também todos os artefatos que estão presentes no ambiente (em particular, na sala do Coding Dojo) que interagem no processo de aprendizagem. Por exemplo, a imagem projetada da tela do computador e o próprio computador também são atores que exercem um papel fundamental no processo de aprendizagem, em sua interação com professores e alunos. Então, nessa estrutura, além do novo, é importante também criarmos interações com esse meio que é dinamicamente construído por cada célula que é componente desse organismo. Além de um conhecimento individual sólido e adaptável, a fomentação de um conhecimento plural também o torna mais elástico para a adaptação ao lidar com o que é novo.

O pensamento construtivista e a maneira com que propõe a construção do conhecimento são contemplados na prática do Coding Dojo. Existe a preocupação em alimentar o novo rotineiramente. Ter o assunto discutido agora em mente é primordial para podermos assimilar tópicos relevantes à prática do Coding Dojo, que serão abordados nos próximos capítulos desta monografia.

2.2 PROBLEM BASED LEARNING

O Problem Based Learning – PBL – ou Aprendizagem Baseada em Problemas, em português, é uma prática pedagógica que posiciona o aluno como condutor responsável e autônomo por seu aprendizado, enquanto o guia (professor orientador) fomenta a orientação cognitiva em um processo de ensino que dissemina o conhecimento através de contextos e problemas reais que aquele aluno poderá encontrar em sua vida profissional. O PBL é uma prática em que os alunos guiam o processo de aprendizagem, colocando o professor com um papel de tutor e facilitador do processo.

O PBL surgiu no final da década de 1960 no curso de medicina da McCaster Univerity, no Canadá, e foi criado por Howard Barrows (NEVILLE, 2009). No caso de um curso de medicina, os alunos eram apresentados a casos clínicos reais. Mas, apesar de ter sido criado dentro da área médica, a maneira de lidar com a convivência em grupo e os seus conceitos propostos pelo PBL podem ser disseminados para outras áreas, como física, computação, direito, matemática, economia, engenharia e muitas outras (WOOD, 2003). Isso é possível por possuir um caráter muito genérico e não limitado ao funcionamento da área médica.

Ao colocar o aluno como centro de todo o processo educacional (REHM, 1998), o PBL subverte o modelo do aluno passivo e receptor e propõe um modelo em que este tome uma postura ativa e que possa criar conhecimento. Isso ocorre pela maneira com que o assunto a ser ensinado é abordado. Ao invés de seguir o fluxo em que toda a teoria é apresentada e só depois disso são dirigidos aos alunos problemas reais, o PBL propõe o fluxo inverso. A lógica defendida é a de que, dado o domínio da matéria, escolhe-se um cenário real para ser apresentado para o aluno e, a partir daí, as informações surgem sempre seguindo as demandas criadas pelos alunos para a resolução desse problema. Ao proporcionar um caso real para os alunos, estes se tornam os responsáveis por definirem quais os objetivos a serem atingidos naquele caso. Ou seja, qual vai ser o assunto que eles querem abordar na reunião.

Nesse processo, o papel do professor também é subvertido. Este, que hoje é visto como o condutor do sistema de ensino com a função de determinar o momento em que cada assunto será abordado, transforma-se em um guia, um catalisador do processo. Essa postura é importante por deixar com que a resolução do problema cresça das necessidades dos alunos e que o professor tenha a responsabilidade de potencializar as ideias e não já as apresentando. A construção nesse processo é coletiva e é criado um laço de relacionamento mais pessoal e menos formal entre aluno e professor, já que os alunos passam a enxergar o professor como um guia que tem a função de auxiliá-los e não mais como um dono da verdade sobre o tema.

O PBL também propõe que a construção do conhecimento não se dê de forma individual, mas sim em grupos, que, geralmente, vão de 8 a 10 participantes (WOOD, 2003). Essa visão é importante por acreditar-se que a atividade em grupo tende a sanar vícios individuais com determinados temas e fomenta a discussão intelectual entre os alunos. Principalmente em função do último motivo, o que é importante em PBL é a dinâmica do aprendizado e não o resultado a ser obtido. O importante é fazer com que os alunos adquiram o conhecimento e sejam capazes de repassá-lo e discuti-lo com outros alunos.

O trabalho em grupo não se limita a facilitar a troca de conhecimento, mas também valoriza outros atributos das pessoas como a maneira com que se comunicam, trabalho em equipe, resolução de problemas, responsabilidade em gerenciar a maneira que aprende de maneira independente, compartilhamento de informação e respeito pelo próximo. Portanto, podemos enxergar o PBL como um método que desenvolve o aluno não só na parte técnica, mas também na sua formação pessoal.

O PBL possui uma organização interna de responsabilidades separada em quatro papéis que são preenchidos pelos participantes (REHM, 1998). Os papéis e suas responsabilidades básicas são: o *escrivão* é o responsável por fazer as anotações do encontro e auxiliar os demais a organizarem suas ideias; o *tutor*, geralmente o papel do professor, é o responsável por auxiliar no processo removendo eventuais dúvidas e guiar a evolução do encontro; o *presidente* é quem mantém os integrantes no foco do problema evitando possíveis dispersões; os *membros* possuem a função de semearem a discussão com perguntas, ideias, resposta e qualquer tipo de informação necessária. A responsabilidade do último papel, o dos *membros*, é compartilhada por todos os outros papéis. Não bastando ter essa divisão, os participantes vão se alternando durante os encontros para que cada aluno possa viver todas as experiências do processo.

WOOD (2003), pesquisadora da área de educação médica, traz alguns pontos que são considerados vantagens do PBL e que valem a pena serem discutidos para entendermos

melhor os seus benefícios. O primeiro e mais importante ponto que ela cita é o foco no aluno. Por ser a base dessa dinâmica, esse ponto já foi discutido nos parágrafos anteriores.

Outro ponto defendido é o estímulo de competências genéricas do aluno. Nos parágrafos anteriores, já foram mencionadas as competências da parte pessoal, como comunicação, trabalho em equipe, etc. Mas, além destas, o método também propicia competências genéricas de caráter profissional. Podemos afirmar isso, já que como é necessária a utilização de cenários reais e é difícil um problema teórico ser reproduzível de maneira idêntica em vários cenários, vemos que uma multiplicidade de conhecimento pode ser viabilizada ao passo que cenários diferentes vão sendo escolhidos. Mesmo que o tema central seja comum, o ambiente em que aquele problema está inserido é diferente de um cenário para o outro. Isso potencializa o aluno a desenvolver habilidades mais amplas e menos limitadas ao domínio teórico.

Além disso, o fato de não termos um cenário ideal e isolado nos leva a construir cenários em que diferentes tópicos de um assunto inevitavelmente possuem interseções. Temos na física, por exemplo, o caso da aceleração e movimento de um objeto com o atrito deste objeto com a superfície em que ele está apoiado. É comum lidarmos com problemas teóricos que desprezam o atrito; entretanto, no mundo real, fazer isso é impossível. Portanto, ao lidar com este impossível, não basta ao aluno saber qual é a equação que determina a aceleração daquele objeto num instante de tempo. Ele precisa saber como o atrito vai influenciar nessa aceleração para que ele consiga o resultado real. Vemos então que, ao contrário do processo teórico, podemos trabalhar o conhecimento mais integrado e não isolando cada parte da área estudada em uma célula isolada. Isso fortalece ainda mais adaptabilidade do aluno a diferentes problemas reais.

Também podemos enxergar como um ponto vantajoso a relação que o aluno estabelece com o problema real. O objetivo do aluno em aprender nesse novo processo deixa de ser o de resolver uma prova e conseguir uma nota em uma prova abstrata e teórica, e passa a ser resolver algo concreto. Isso gera um engajamento maior do aluno em seguir com o desenvolvimento, porque existe uma motivação ao enxergar algo físico ser construído e ser resolvido. Ligar a teoria com a prática, ou seja, sair do campo de abstração dos teoremas e a proposições, traz o assunto para a vida do aluno e transcende as linhas copiadas em um caderno. O aluno passa a viver aquilo, a discutir aquilo com pessoas de fora, pois há o que ele mostrar, há vida no conhecimento que ele vem construindo. O processo puramente teórico, por sua vez, limita o conhecimento às páginas de livros que não possuem tanta expressividade e não causam tanta empolgação nas pessoas como uma experiência real.

Por fim, podemos enxergar no PBL uma abordagem idealmente construtivista. A relação do sujeito (o aluno) com o objeto (o cenário a ser estudado) é o centro de toda a dinâmica. Podemos enxergar as ideias defendidas por Piaget de maneira nítida em todo o sistema do PBL. Para o Coding Dojo, o PBL se torna importante, dado que o Coding Dojo reproduz a lógica de construção de um problema, da existência de um grupo para que se possa trabalhar e de papéis definidos e de donos não fixos, como veremos mais a frente. Será curioso notar mais a frente que, apesar de terem surgido de áreas totalmente diferentes – o PBL veio da pesquisa médica e o Coding Dojo da área de computação – as duas dinâmicas possuem, em seus processos de funcionamentos, diversas áreas em comum.

Capítulo 3

O CODING DOJO

Este capítulo tem como intuito apresentar o objeto central desta monografia o Coding Dojo. Para estabelecer um conjunto de informações organizadas, vamos abordar os temas principais separadamente. Primeiramente falaremos sobre a história do surgimento Coding Dojo até a sua chegada no Rio de Janeiro. Depois, falaremos dos propósitos e dos conceitos por trás do Coding Dojo para só então falarmos sobre a sua dinâmica e a maneira com que ele acontece.

3.1 HISTÓRIA

O começo do Coding Dojo se deu quando Dave Thomas publicou em sua página pessoal uma série de textos questionando a maneira com que os desenvolvedores de software aprendem no ano de 2003. Ele fez um paralelo entre o processo de tornar-se um bom músico e o processo de tornar-se um bom programador. Em ambos é necessário ter talento, conhecer o instrumento que é utilizado (musical ou computacional), ter o conhecimento teórico e, acima de todos estes pontos, é necessário ter prática (THOMAS, 2007b). Nesse ponto é que o músico diverge do desenvolvedor de software.

Em seus textos, Thomas questiona o modelo de aprendizado no mercado de desenvolvimento de software a partir do momento em que afirma que as empresas treinam os seus desenvolvedores apenas alimentando-os de conhecimento teórico. Não existe um momento para o desenvolvedor experimentar e errar (THOMAS, 2007a). Como consequência, o momento em que o desenvolvedor experimenta e erra é o no trabalho. Construindo uma analogia, é como se o músico deixasse para aprender uma nova música no meio do processo de gravação de um disco.

Como solução, ele propõe a ideia de o desenvolvedor ter um tempo próprio para poder praticar a programação. Além disso, ele propõe pequenos problemas que ele chamou de k*atas*. O nome veio da língua japonesa onde a palavra kata é utilizada no meio das artes marciais e significa um exercício de repetição em que o aprendiz imita os movimentos do seu mestre. A

ideia da repetição é que, através dela, o aprendiz possa internalizar os movimentos e compreendê-los em sua essência (THOMAS, 2007c).

No final de 2003, após ler as propostas de *kata* de Dave Thomas, o desenvolvedor francês Laurent Bossavit expandiu a ideia de que desenvolvedores também devem praticar. Em sua página pessoal Bossavit fez a primeira proposta do que veio a ser o Coding Dojo no futuro.

Ele defende sua ideia construindo uma comparação, da mesma maneira que Thomas fez quando propôs os katas. Mas, dessa vez, Bossavit faz uma comparação da maneira com que se aprende a lutar judô com a maneira que se aprende orientação a objetos em Java. Para o primeiro, existe o dojo, ambiente para treinamento de artes marciais, em que o aluno pode ir e terá um horário fixo em que aprenderá semanalmente e irá conhecer novos alunos. Para o segundo, a solução adotada é matricular o desenvolvedor em um curso de Java (BOSSAVIT, 2004).

Como solução, Bossavit foi explícito e quis criar um ambiente que refletisse na programação o que o dojo é para as artes marciais. A ideia é a criação de um espaço com seus próprios valores e rico de interações entre as pessoas. Um ambiente em que fosse possível aprender com as técnicas dos outros e também fosse possível ensinar as técnicas aprendidas a outros. Enfim, um lugar que fosse seguro errar para aprender (BOSSAVIT, 2011). Com o amadurecimento dessas ideias, Bossavit fundou, junto com Emmanuel Gaillot, o Coding Dojo de Paris.

Antes de chegar ao Rio de Janeiro, o Coding Dojo começou em São Paulo e foi trazido por Danilo Sato após ter conhecido o Emmanuel Gaillot em um evento de desenvolvimento de software em Paris (SATO, 2007). No Rio de Janeiro, o Coding Dojo iniciou-se no final de 2008 após a edição da PyconBrasil – conferência da comunidade Python brasileira – que aconteceu no Rio de Janeiro.

Os participantes do Coding Dojo de São Paulo que vieram para participar do evento promoveram uma sessão durante o evento e foi quando os participantes do que veio a futuramente ser o DojoRio – comunidade de participantes de Coding Dojo do Rio de Janeiro – tiveram o primeiro contato com a prática de Coding Dojo. Depois desse primeiro contato, iniciaram-se os encontros que passaram por diversas empresas e faculdades, incluindo a UFF.

3.2 PROPÓSITO

Como analisar quantitativa e qualitativamente o propósito de um grupo de pessoas diferentes entre si? Como estabelecer uma meta se temos pessoas com anseios diferentes, expectativas diferentes e, o mais importante, experiências de vida diferentes? Essa é uma característica curiosa, intrigante e comum entre as pessoas que participam de sessões de Coding Dojo. Elas não acreditam que o propósito seja único.

Há, pelo menos, uma maneira de classificarmos um propósito comum que leve pessoas de diferentes níveis de programação – variando desde calouros de cursos de Ciência da Computação até já renomados profissionais no mercado – a programarem juntas por algumas horas: ser livre. Tentar unificar os propósitos individuais derivados dessa liberdade é cair no erro de massificar e padronizar o que há de mais importante em cada Coding Dojo, que é a sua singularidade.

Esse é o erro que vejo tanto nas universidades quanto nas escolas quando definem um propósito comum para uma determinada disciplina. Normalmente, o propósito acaba guiando o processo de ensino da disciplina. Assim, caso uma pessoa não se adapte ou não aceite esse processo definido, ela está automaticamente excluindo-se ou marginalizando-se de todo o processo de ensino. Simplesmente porque, para ela, o que está definido não funciona. Passa-se então a trocar o grau de importância entre processos e o impacto que ele gera nas pessoas. Se o processo não atende as necessidades das pessoas, ele deve ser mudado o quanto antes.

Por este motivo que uma sessão de Coding Dojo é muito mais maleável a adaptável a real necessidade das pessoas quando comparado a um sistema com propósitos predefinidos. As pessoas podem, através de seus medos, dúvidas, conhecimentos e questionamentos, definirem qual é o propósito que querem seguir. O que vai ser feito e produzido emerge de um senso coletivo e não de uma banca de dois ou três professores. E isso só pode ser feito quando damos liberdade para as pessoas escolherem os seus próprios caminhos.

Obviamente, toda essa filosofia não foi criada dentro do ambiente do Dojo. Pensarmos em pessoas de diferentes conhecimentos juntas, por livre e espontânea vontade e desenvolvendo código, nos remete imediatamente à filosofia do modelo open-source e à visão do desenvolvimento coletivo do código. Esses não são propósitos explícitos do Dojo, mas sim efeitos colaterais de extrema importância e que guiam alguns dos conceitos e práticas do Dojo.

A maneira de criar código coletivamente subverte a visão tradicional e individual que temos do conhecimento do que é feito. O padrão é pensarmos em aprender mais para usarmos

o que aprendemos ao nosso favor em troca de uma possível escalada profissional ou um renome acadêmico. A visão proposta pelo desenvolvimento coletivo é oposta. Essa visão defende que as pessoas querem sempre aprender mais para, além de se tornarem melhores desenvolvedores, ajudarem outras a aprenderem também. Assim, troca-se o ambiente competitivo por um ambiente sustentável e colaborativo para o desenvolvimento do software.

Aliado a esses conceitos, o Coding Dojo reforça o sentimento de que o indivíduo faz parte de uma comunidade. O Coding Dojo tem o intuito claro de reunir pessoas. Pessoas conectadas por um anseio em comum: o de poder programar, aprender e ensinar guiados pela maneira que lhes for mais conveniente. Essa intenção é bem próxima do conceito de comunidades de desenvolvimento de software livre. Pessoas conectadas entre si com o intuito de produzir e/ou trabalhar com tecnologias que acreditam que são mais úteis e interessantes.

Esses fatores exercem impacto direto na maneira com que produzimos código e nos comunicamos. Cientes disso, atualmente, além do ambiente acadêmico e dos alunos de informática e computação já terem percebido os benefícios do Coding Dojo, empresas da área de tecnologia da informação enxergam nele oportunidades e soluções práticas para a resolução de problemas da empresa, principalmente quando se trata de pessoas (CUKIER, 2009).

Com o Coding Dojo, as empresas perceberam que é mais fácil gerar um alinhamento entre os seus funcionários. A criação de um ambiente calmo e seguro para os desenvolvedores resulta em um maior conforto para discussões entre as pessoas. Assim, ideias e opiniões sobre diversos assuntos, conhecimento sobre novas tecnologias, metodologias de desenvolvimento, comunicação entre diferentes equipes e outras questões vão sendo tratadas sem que se torne uma burocracia ou apenas outra reunião geral da empresa.

Como o Coding Dojo tem o caráter de ensino e aprendizado, é mais fácil de o gerente, por exemplo, introduzir novas metodologias e práticas no desenvolvimento do dia a dia dos funcionários. Fazê-los experimentar essas ideias de maneira prática junto com toda a equipe é mais produtivo do que focar em fornecer um curso para um funcionário aprender um tema e responsabilizá-lo por passar o conhecimento para os demais. No processo do Dojo, você fomenta muito mais a chance de questionar os porquês e os benefícios que os novos processos poderiam trazer para as equipes do que centralizar o conhecimento em uma só pessoa.

Por fim, as empresas estão começando a perceber que novos talentos no mundo da programação tem surgido no Coding Dojo e continuam frequentando o mesmo. Como numa sessão tanto a habilidade pessoal quanto a habilidade de desenvolvimento do programador é mostrada, investir no Coding Dojo, para a empresa, significa poder estudar as pessoas que se encaixam mais no perfil do trabalho. Além disso, investir nessa comunidade resulta em

despertar o interesse de pessoas que tem como partes do perfil o interesse em tornarem-se bons profissionais e a preocupação em aumentar o seu conhecimento.

Podemos perceber então diferentes pontos de vista sobre os motivos de se participar de um Coding Dojo. Podemos separá-los em grupos em que consiste o indivíduo como unidade e determinado pelas suas expectativas; a *comunidade*, que tende a alinhar conceitos comuns que podem emergir durante as sessões; as *empresas*, que veem uma grande oportunidade de crescimento e aproveitamento nos reflexos dos encontros sobre os seus funcionários.

3.3 CONCEITO

O principal conceito do Coding Dojo é o foco nas pessoas. Como se trata de uma congregação de pessoas com interesses em comum, é natural que essas pessoas sejam a peça principal para a existência do encontro. Os conceitos adicionais e metodologias existentes no Coding Dojo visam potencializar o rendimento e o entendimento das pessoas sobre o que está sendo feito.

O caráter social torna-se extremamente importante a partir do momento que os conceitos, as maneiras de abordagem do problema e o código em si são tratados coletivamente. Portanto, são normais situações em que as pessoas devem expressar verbalmente as suas ideias e fazer com que as outras pessoas as entendam. O fato de as ideias não serem impostas e sim discutidas, negociadas, adaptadas e possivelmente aceitas, obriga a esse tipo de diálogo ser feito da melhor maneira possível. Ter uma comunicação limpa e respeitosa agiliza e otimiza tal conversa, portanto, não é interessante manter para isso um processo de comunicação burocrático. O efeito que isso resulta nas pessoas é o de elas acabarem formulando seus pontos de vista de maneira mais reflexiva e ponderando o momento de falar, o que é um aprimoramento da maneira de elas se expressarem.

Além da expressão verbal, é notório que o próprio código produzido torne-se expressivo. Um código expressivo, ou seja, de fácil entendimento, gera uma assimilação mais rápida pelas pessoas. Sendo assim, as pessoas acabam por praticar a escrita de um código mais semântico e coerente. Para isso, os participantes fazem uso tanto de padrões de projeto, como Orientação a Objetos (FREEMAN, 2010b), como também abordam questões que às vezes são julgadas como detalhes e esquecidas, como nomes de variáveis e métodos mais significativos ou o uso de uma identação correta. A ausência desses fatores gera incômodo aos

participantes do Coding Dojo, já que o código fica mais incompreensível e estes acabam por prezar pelo uso dessas boas práticas.

Como dito anteriormente, o código é coletivo e, como tudo que é regido por um grupo de pessoas, está sujeito a confronto de ideias. É normal que uma nova ideia venha de encontro com a solução que já foi implementada. Caso essa nova ideia seja julgada melhor por todos, pelo princípio de no Coding Dojo as pessoas estarem interessadas em moldar o processo utilizando sempre o que julgam melhor, ela deve ser absorvida e utilizada. Pode-se perceber que esse processo gera um impacto sobre as pessoas de duas maneiras. A primeira é que o participante se torna mais suscetível a ouvir, entender, ponderar e aceitar ou não outras opiniões. A segunda é aumentar a facilidade com que ele consegue reagir a mudanças já que nem sempre a decisão que ele tomou e programou vai perdurar para sempre. Ele, ao ver o que ele criou ser alterado, precisa entender, assimilar e programar a nova ideia.

Tal análise comportamental de um grupo de pessoas não pode ser feito sem que existam outros fatores que propiciem as naturezas de relacionamento descritas acima. Um dos alicerces para esse tipo de convívio é a construção de um ambiente que suporte às atividades de maneira sustentável, sem espaço para agressividades. Tal ambiente possui alguns pontos chave e que devem ser garantidos para o sucesso da sessão de Coding Dojo.

A primeira e principal característica desse ambiente é ele não ser competitivo. A competição naturalmente impõe papéis de mais fracos e mais fortes sobre as pessoas. Sob a influência desses papéis, as atitudes e reações individuais são modificadas. Possíveis decisões serão tomadas por imposição de uns ou omissão de outros. Possíveis questionamentos e discussões poderão ser deixados de lado por medo de serem reprimidos ao serem levantados. Portanto, podemos concluir que em um ambiente competitivo é mais difícil exigir que as pessoas se comportem de maneira livre, já que elas estarão em uma constante luta para assumir o papel do mais forte. Isso inviabiliza a existência e prática todos os conceitos sociais levantados anteriormente.

Dado que existe um ambiente não competitivo, precisa-se estendê-lo ainda mais para propiciar um relacionamento aberto entre as pessoas. É necessário que este ambiente seja colaborativo. É necessária a conscientização de que, além de todos terem graus de importância iguais dentro da sessão, todos podem colaborar igualmente uns aos outros. Em função desta característica que podemos contar com opiniões colaborativas e não egoístas emergindo.

Essa visão vai de direto encontro à visão do ensino tradicional. Na visão tradicional, a competição é declarada. As ferramentas de medição são trabalhos e provas que pontuam a habilidade de um aluno aprender – ou decorar – um determinado tema. Em poucos casos é

avaliada a habilidade real desse aluno de praticar o conhecimento adquirido, de questioná-lo e de discuti-lo. Isso se dá pois, além do ambiente competitivo, existe a figura central do professor como dono do conhecimento. Esse modelo não prevê a socialização do conhecimento em vários fluxos, mas sim a imposição marcada pelo fluxo professor versus aluno.

Por fim, temos que o ambiente é também inclusivo. Sabe-se que ele não é competitivo e é colaborativo. O somatório dessas duas características pode refletir em um aumento pela sede de conhecimento. E, se o conhecimento se dá através da divisão, a única maneira de aumentar o volume de informação trocada é com a chegada de novos pensamentos, ou seja, novos pensadores. Assim, percebe-se um fenômeno convidativo natural nos participantes com outros que ainda não experimentaram e não participaram do processo. É um efeito similar a lei de ação e reação da física – descrita pela Terceira Lei de Newton - já que quem ensina quer aprender e quem aprende quer também ensinar.

Todas essas características são utilizadas para criar-se um ambiente seguro e livre de dúvidas. Mas a criação de um ambiente como este se torna complicada se a liberdade individual não for respeitada. Ao firmar um ambiente livre e respeitoso, as pessoas se sentem mais a vontade para perguntar, pois sabem que não serão avaliadas. Sabem que não terão que provar o que aprenderam mediante um processo com que não se adaptaram ou não gostam. Vão respeitar somente os seus próprios sistemas de avaliação. Dado que o aprendizado é fruto do questionamento, vemos que o Coding Dojo, utilizado de maneira correta, pode ser uma grande fonte para a satisfação pessoal acerca do tema.

Entender esse ambiente e aliado ao sentimento de liberdade, implicará em quebrarmos com o modelo mestre e aprendiz. Mestres e aprendizes diferenciam-se não em suas qualidades, mas em seus “defeitos”. Enquanto os aprendizes não possuem o conhecimento técnico para resolver o problema, os mestres, por sua vez, o possuem. Entretanto, este conhecimento vem carregado de vícios e abordagens mecânicas de lidar com alguns tipos de problemas adquiridos com o tempo de desenvolvimento que nem sempre são as ideais, mas resolveram os problemas do mestre até então. Portanto, cabe ao mestre nessas circunstâncias passar o conhecimento e, ao aprendiz, questionar a maneira com que o conteúdo se forma. Nesse momento, o mestre poderá enfrentar os seus problemas, preencher as suas lacunas e aprender com as respostas que terá que dar. Percebe-se então uma troca de papel em que quem não sabia nada, agora ensina por questionar o que aprendeu e, quem antes respondia, agora aprende ao refletir. Nesse processo, os papéis de mestre e aluno não são mais fixos e sim dinâmicos de acordo com a situação.

Esse pensamento é fruto do ambiente e da visão do Coding Dojo. Só conseguimos aprender algo quando assumimos para nós mesmos que nunca saberemos tudo que cerca aquele tema. Essa reflexão remove egos pessoais e abre a pessoa para o conhecimento. Aprender a perguntar e aceitar que a sua solução não é sempre a melhor é a chave-mestra para o participante tirar o maior proveito possível durante uma sessão de Coding Dojo.

Como dito anteriormente, com este ambiente consolidado, percebemos que conceitualmente modificamos o padrão social a que estamos acostumados a ensinar e aprender. A visão singular de aluno e professor dá lugar a uma visão plural de aprendizado coletivo. Agora, o que deve ser feito proceduralmente para manter o foco e a motivação das pessoas durante uma sessão de Coding Dojo e, ao mesmo tempo, fazer com que elas aprendam algo?

O Coding Dojo utiliza a maneira de lidar com o problema proposta pelo PBL que inverte o fluxo normal de introdução, apresentação do conteúdo e problemas para resolver. O resultado dessa inversão de fluxo é que o aluno primeiro lida com a dificuldade para resolver o problema. Isso gera uma reflexão e ponderação acerca de possíveis maneiras de resolver o problema. Após discussões sobre o como fazer, normalmente é decidida uma melhor maneira e é essa a utilizada. Passando por esse processo, conseguimos enxergar nitidamente a necessidade real daquilo que aprendemos e que nos é ensinado.

Como visto, o Coding Dojo assemelha-se bastante ao PBL em relação ao problema. Entretanto, ele ainda dá ao problema um tratamento adicional. A visão é desburocratizar o aprendizado e, para isso, trazer o problema para o contexto normal das pessoas. Um problema de roteamento de uma requisição HTTP dentro de uma infraestrutura de servidores não é natural para a maioria das pessoas. Entretanto, lidar com a movimentação financeira de uma bilheteria de cinema levando em considerações diferentes tipo de promoções já é mais comum. Não é estranho, em sessões de Coding Dojo, deparar-se com problemas como o do jogo de campo minado, avaliação de uma mão de poder ou conversão de números romanos para inteiros exatamente por esses serem mais próximos do dia-a-dia das pessoas.

Além de criar um laço mais próximo com o problema, a visão de problemas mais relacionados ao cotidiano também aumenta o grau de abstração. Ao abordar problemas reais e não meras impressões na tela resultantes de avaliações de números, como são compostos a maioria dos exercícios da aula de programação que pude vivenciar na UFF, a pessoa consegue abstrair-se de um mundo numérico para pensar em uma solução mais familiar para o problema, pois se trata de um problema concreto. Assim acredito que seja mais fácil remover a barreira invisível de todo desenvolvedor de pensar no código antes do algoritmo.

Esses pontos refletem o conceito do Construtivismo embutido no ambiente do Dojo. Esse conceito estabelece que o conhecimento não é adquirido sobre imposição de um meio qualquer, mas sim construído através das relações e respostas aos estímulos externos de um meio qualquer. Portanto, ter maiores laços com um problema repercute a visão de estimular a construção de um conhecimento.

Por fim, apesar do tipo e da escolha do problema ser algo essencial para o Coding Dojo, resolvê-lo não é essencial. Como dito anteriormente, aprendemos muito mais com questionamentos do que somente com as respostas e o Coding Dojo segue essa filosofia. Além disso, não se ater a resolver o problema consegue fazer com que as pessoas foquem no processo e possam tirar maior proveito e aprendizado do mesmo.

3.4 DINÂMICA

Como todo processo de ensino, o Coding Dojo também possui seus passos e ferramentas que devem ser utilizados para facilitar a troca de conhecimento entre os participantes. O intuito dessa sessão é explicar essas ferramentas e esses passos.

Vale dizer que existem dois tipos básicos de Coding Dojo: o *PreparedKata* e o *RandoriKata*. A diferença dos dois, basicamente, é que o primeiro é preparado previamente por uma pessoa, enquanto o segundo é construído coletivamente no início do encontro (FERNANDO, 2011). O *RandoriKata* é o formato mais utilizado exatamente por ser construído de maneira coletiva e não demandar uma organização prévia. É este tipo que será abordado nas próximas sessões do capítulo.

3.4.1 As Peças

A primeira exigência para se fazer uma sessão de Coding Dojo é a existência de um conjunto de peças essenciais. Primeiro, é necessário um ambiente físico, idealmente uma sala, para acontecer a sessão. Além disso, é necessário um projetor e um computador. O projetor, conectado ao computador, irá projetar o código que vai ser gerado durante a sessão. Depois de

toda a parte física preparada, um problema deve ser escolhido segundo as observações levantadas na parte em que os conceitos do Coding Dojo foram abordados. Dado que foi determinado um problema, escolhe-se uma linguagem de programação para desenvolver o problema. Sobre a linguagem existem alguns pontos a serem discutidos.

A linguagem deve fornecer uma maneira que os participantes possam testar o código que vai ser desenvolvido. Isso é necessário pois o Coding Dojo utiliza o princípio do Test Driven Development, conhecido como TDD, (FREEMAN, 2010a) para seguir com o desenvolvimento. O TDD é um processo para se desenvolver software que impõe um fluxo de desenvolvimento. Com ele, primeiro escreve-se um teste para garantir alguma lógica da aplicação para somente depois escrever o código que implemente aquela lógica. A ideia, como presente no nome, é a de que os testes guiem a maneira com que o código é desenvolvido.

O TDD é importante, porque, com os testes funcionando como validadores da aplicação, temos um feedback em tempo de implementação se o que está feito atende ou não as necessidades já descritas. Em um processo de ensino isso é importante, porque o aluno tem o resultado da sua experiência em um tempo muito curto – o tempo de escrever o código e executar os testes – e, com esse tempo curto, ele possui mais tempo para experimentar novas ideias e conseguir novas experiências com o objeto que é ensinado.

Portanto, no Coding Dojo temos o TDD não somente como um processo de desenvolvimento, mas também como um processo para a construção de experiência com o aluno. Assim torna-se mais complicado fazer um Coding Dojo com uma linguagem que não possibilite os seus desenvolvedores a testarem o seu código – já que adaptações terão que ser criadas. A ferramenta a ser escolhida é livre, basta apenas ser compatível com a linguagem.

Ainda sobre a escolha da linguagem, outro tópico a ser considerado é a diversidade. Sabemos que hoje existem dezenas de linguagens de programação. Cada uma tem seus pontos fortes, seus pontos fracos, seus objetivos, seus propósitos e filosofias. Como já mencionado anteriormente, no Coding Dojo a diversidade de pensamento é valorizada, pois é ela que gera diálogos, grandes responsáveis pela troca de conhecimento. Mas, além da diversidade do pensamento, a pluralidade de linguagens de programação também é encorajada por três pontos.

O primeiro ponto é que o desenvolvedor tende a sair da sua zona de conforto composta pelas linguagens que é especialista. O segundo ponto é que linguagens diferentes tendem a exprimir ideias e maneiras de manipular problemas diferenciados ou, até em um caso mais

drástico, fazerem uso de paradigmas de programação diferentes. O caso de Java e Haskel serve como um exemplo, pois o primeiro faz uso do paradigma de orientação a objetos e o segundo o de programação funcional. Então, utilizando linguagens diferentes, o desenvolvedor se expões mais para aprender algo novo. Por último, conhecer várias linguagens dá a possibilidade ao desenvolvedor de saber escolher melhor, dado um problema, qual a linguagem que melhor resolve aquela situação.

As últimas peças que compõem o Coding Dojo são as pessoas. Elas também se arrumam em papéis não fixos ao longo da sessão, diferentemente dos papéis do PBL. Como a linguagem de programação não é fixa, pode ser que seja escolhida uma em que a maioria dos participantes não tenha conhecimento. Para resolver esse problema é sempre aconselhável ter o chamado *Guru da Linguagem*. Ele fará o papel de ajudar os participantes a lidar com a linguagem. Assim, ele deverá explicar desde erros de sintaxe até conceitos, filosofias e funcionamento da linguagem.

Além do Guru, existem o piloto e o co-piloto. Esses papéis são importantes porque eles compõem o par utilizado na programação em par (WILLIAMS, 2000). A programação em par é um modelo de desenvolvimento em que dois programadores desenvolvem em conjunto, lado a lado, um mesmo código. Essa abordagem favorece alguns pontos importantes no processo de desenvolvimento como diminuir a ocorrência de erros, serem um nivelador de conhecimento ou propagar padrões para a escrita de código. Isso se dá pelo fato da maneira com que a programação em par modifica o relacionamento do desenvolvedor com o código.

O fluxo normal de diálogo do desenvolvedor com o código é de imposição. O desenvolvedor escreve o código e compila ou executa o mesmo. Em ponto nenhum existe alguma interferência de indagação do que está sendo feito. Entretanto, ao promover o desenvolvimento em par, esse fluxo de imposição torna-se mais inviável já que a solução deve satisfazer aos dois desenvolvedores. Portanto, o diálogo para a construção da implementação torna-se inerente e essencial ao processo.

No Coding Dojo, a divisão entre piloto e co-piloto respeita quem está digitando o código no momento. A ideia é que o piloto digite o código, enquanto o co-piloto completa o par auxiliando-o com a construção do código trazendo discussões com base no código produzido. Tanto o piloto quanto o co-piloto devem estar sempre explicando o que está sendo feito para que os outros participantes possam entender. Estes últimos participantes, por sua vez, formam a plateia. Enquanto o piloto e o co-piloto criam o código, a plateia acompanha o crescimento do mesmo através do que é projetado. Assim, todos têm a visão do que está sendo criado em tempo real.

Separando-se duas pessoas das demais e deixando as outras meramente observando o que está sendo feito parece não se diferenciar muito do modelo tradicional de ensino. Entretanto, como dito anteriormente, esses papéis não são fixos. Com o conceito de rodízio de pessoas essa visão tradicional é desfeita. O processo é estabelecer um período de tempo para a duração turno. Enquanto durar o turno, o par estará programando e a plateia observando. Após esse intervalo de tempo, temos uma alteração na configuração das pessoas. O piloto abandona o seu posto e o co-piloto assume seu lugar. O lugar do co-piloto, por sua vez, é assumido voluntariamente por alguém da plateia.

A intenção é que todos possam passar pelo menos por um turno para viverem a experiência. A coletividade do código, a explanação do que está sendo produzido, a modificação do código anterior, são esses os passos que reproduzem alguns os conceitos do desenvolvimento open-source e de interação entre as pessoas mencionados anteriormente. Focar no empenho nesse momento significa aumentar em muito a produtividade da sessão por estar lidando com pontos críticos do processo.

Dado que foi definida a infraestrutura necessária e os papéis das pessoas, é necessário explicitar a maneira com que o código é desenvolvido. Como dito anteriormente, no Coding Dojo fazemos uso do TDD de uma maneira bem gradual. Comumente essa visão é chamada de passos de bebê (ARTEM, 2007) e remete à visão de começar pelo teste julgado mais simples. No Coding Dojo, é utilizada uma notação abstrata para determinar o estado dos testes que remete a um sinal de trânsito. No caso dos testes estarem passando, é comum se dizer que o sinal está no verde. No caso contrário, de os testes não estarem passando, é comum se dizer que o sinal está no vermelho. Esses dois estados implicam no funcionamento da sessão dado que existem regras para cada um deles.

Quando o sinal está no vermelho, ou seja, durante o momento em que o par está desenvolvendo a solução para os testes escritos, a plateia não deve se manifestar. Mesmo sabendo que existe um erro de sintaxe, que a lógica utilizada está errada ou que alguém saiba uma solução mais rápida e mais simples, todos da plateia permanecem em silêncio para deixar que o par enfrente o problema. Isso está diretamente ligado ao conceito de PBL de fazer com que o aluno enfrente o problema para conseguir buscar uma solução. A analogia feita é a de como andar de bicicleta. A pessoa nunca vai aprender se sempre utilizar uma bicicleta com rodinhas de apoio. A pessoa está sendo privada de cair, está sendo privada da dúvida.

O interessante desse ponto é que, como essa visão de postura da plateia é compartilhada por todos, existe uma autoorganização dentro da mesma para que ela não interfira no trabalho que o piloto e o co-piloto estão fazendo. Entretanto, essa regra de silêncio

pode ser quebrada a qualquer momento caso alguém da plateia esteja com dúvida sobre qualquer assunto, desde a linguagem até o que está sendo feito. É comum dizer que durante a sessão não existem sinais para as dúvidas.

Enquanto no vermelho, a postura para o par é diferente. Eles devem explicar o que estão fazendo, o código que está sendo escrito e o raciocínio que estão seguindo. Além disso, para qualquer dúvida que eles tiverem, seja de implementação, conceito ou sintaxe da linguagem, eles podem perguntar para o guru ou para qualquer um na plateia, que tem o direito e dever de responder a dúvida. Afinal, não há como os pilotos prosseguirem caso tenham dúvidas em como fazer.

3.4.2 O Fluxo

Com o entendimento das peças e os seus papéis, podemos explicar o fluxo de uma reunião de Coding Dojo. Para isso, o fluxograma da Figura 1 será utilizado:

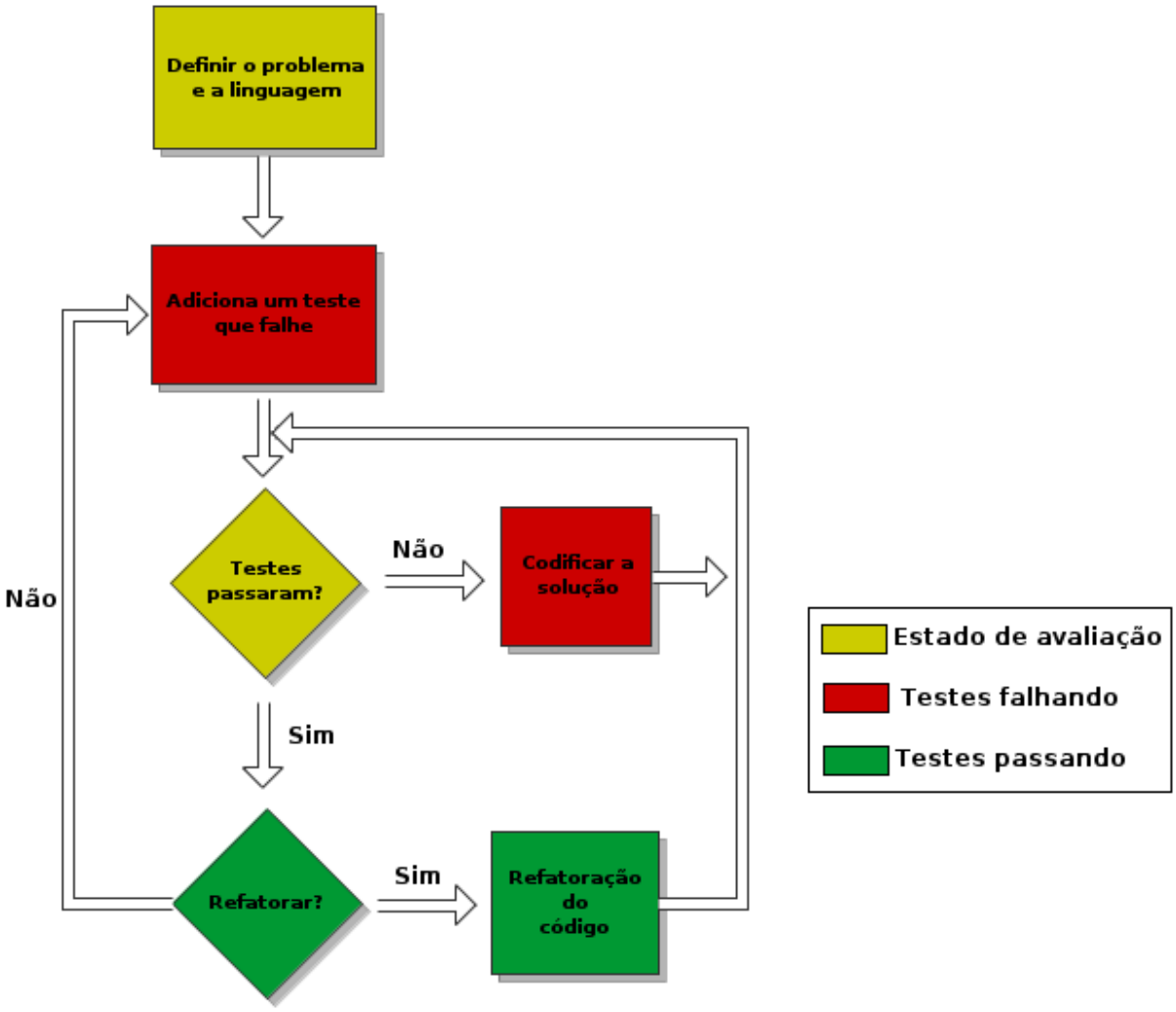

Fig. 1: Fluxograma Coding Dojo

Com o fluxograma podemos ver que o primeiro passo é a escolha do problema e da linguagem de programação a ser utilizada. Após essa escolha, inicia-se a codificação. Seguindo os princípios do TDD, primeiro é adicionado um teste que falhe para somente depois o código ser desenvolvido. Mas, antes de começar a desenvolver o código, sempre é necessário validar o estado da aplicação executando os testes. Essa validação pode seguir por dois fluxos, o dos testes estarem passando ou não estarem passando.

No caso dos testes não estarem passando, é necessário codificar a solução para atender somente as necessidades explicitadas pelos testes já escritos. É interessante ressaltar esse ponto para que não seja feito um esforço maior para resolver problemas do domínio que ainda não foram explicitados pelos testes. Após o código ser escrito é necessário avaliar se a solução implementada satisfaz os testes.

Após os testes serem executados e estarem passando, existe o momento em que o código pode ser refatorado. A refatoração é definida como um processo em que o código é alterado de uma maneira que não comprometa ou altere o seu comportamento externo mas que melhore a sua estrutura interna (FOWLER, 1999). Então, nesse momento, o código é alterado para que ele seja aprimorado. Entretanto, não é sempre que o código precisa ser refatorado.

Caso o código precise ser refatorado, após a refatoração ser feita, é necessário que os testes estejam passando. Os testes, nesse momento, garantem o comportamento externo da aplicação, então, devem continuar sendo respeitados. Caso o código não precise ser refatorado, um novo passo para a resolução do problema é dado e um novo teste é adicionado retornando ao fluxo inicial que é mantido por toda a sessão.

A sessão tem uma duração média de 2 horas. Após esse tempo é feita uma rápida reunião de retrospectiva. Essa reunião foi herdada do modelo de desenvolvimento ágil de software (KERTH, 2001). Existem dois intuitos principais com a retrospectiva. O primeiro é o de criar uma concepção global da experiência da sessão a partir da percepção individual de cada um. O segundo é o de reforçar a sensação de coletividade que o Coding Dojo transmite.

Para iniciar a reunião de retrospectiva, cada participante escreve em um papel tudo o que aconteceu durante a sessão que ele julgou positivo, e no outro lado, todos os pontos que ele julgou que podem ser melhorados. Após cada um escrever seus pontos de vista, são feitas duas rodadas em que cada um terá sua vez para falar. Na primeira rodada, cada participante falará sobre os pontos positivos que listou. Após todos falarem, inicia-se uma segunda rodada em que cada um fala os pontos a melhorar que foram listados. São feitas duas rodadas para que as pessoas consigam separar os pontos de uma natureza e de outra e focar separadamente em cada um, tornando uma avaliação geral mais fácil e menos confusa. Além disso, é esse um dos momentos em que cada um pode treinar ainda mais a sua expressividade.

Apesar de não fazer parte do processo como um todo, é uma consequência das relações que surgem dentro do encontro as pessoas permanecerem juntas após o término do encontro. É normal promoverem encontros em bares ou restaurantes, comumente chamado de pós-dojo. Assim, cria-se um ecossistema social em que as pessoas notam afinidades em comuns entre si e passam a conversar não mais só sobre o Coding Dojo, mas também sobre os interesses em comum. A partir daí deixam de ser meros conhecidos para construírem relacionamento que extrapolem o ambiente dos encontros.

Capítulo 4

PESQUISA

Com o intuito de quantificar e estudar analiticamente alguns dos argumentos levantados nesse projeto, uma pesquisa foi realizada através de um formulário distribuído às diversas comunidades que apoiam o Coding Dojo pelo país. A pesquisa foi hospedada no blog pessoal do autor desse projeto e durou duas semanas (FONTES, 2011). Foi iniciada no dia 25 de Abril de 2011 e encerrada no dia 09 de Maio de 2011. Após essas duas semanas, pudemos contar com a resposta de 174 pessoas que contribuíram com a pesquisa.

O formulário continha questões que levaram em consideração o aspecto social do Coding Dojo focando a interação dentro de comunidades de desenvolvimento de software, e o aspecto social, focando os potenciais do Coding Dojo citados anteriormente. O questionário foi dividido em quatro áreas que serão explicitadas abaixo.

A primeira área é referente à identificação e qualificação do entrevistado. Nessa área, além de nome e e-mail, foi perguntado se o entrevistado era profissional da área de tecnologia da informação ou estudante. Dos 174 entrevistados, tivemos 80 (46%) que se colocaram como profissionais, 44 (25.3%) estudantes e 50 (28.7%) que responderam que estudam e trabalham ao mesmo tempo, como pode ser visto na Figura 2.

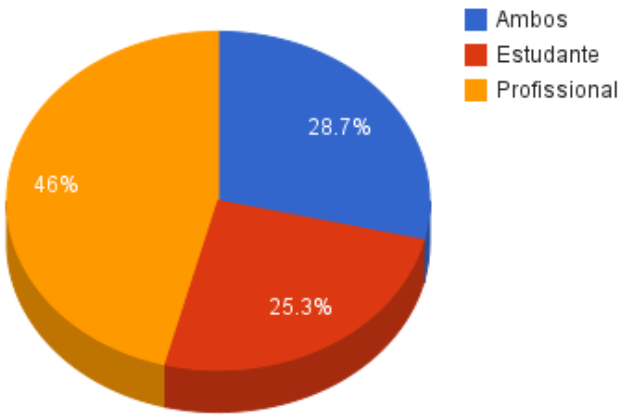

Fig. 2: Gráfico de Estudantes X Profissionais

Dos estudantes, 6 (6.4%) ainda não estão cursando uma faculdade, 16 (17%) entre o 1º e 3º período, 16 (17%) entre o 3º e 5º período e a maioria restante de 56 (59.6%) estão acima do 5º período. Portanto, podemos concluir que suas opiniões são bastante relevantes já que 87% são alunos que já possuem, pelo menos, 1 ano e meio de faculdade cursada.

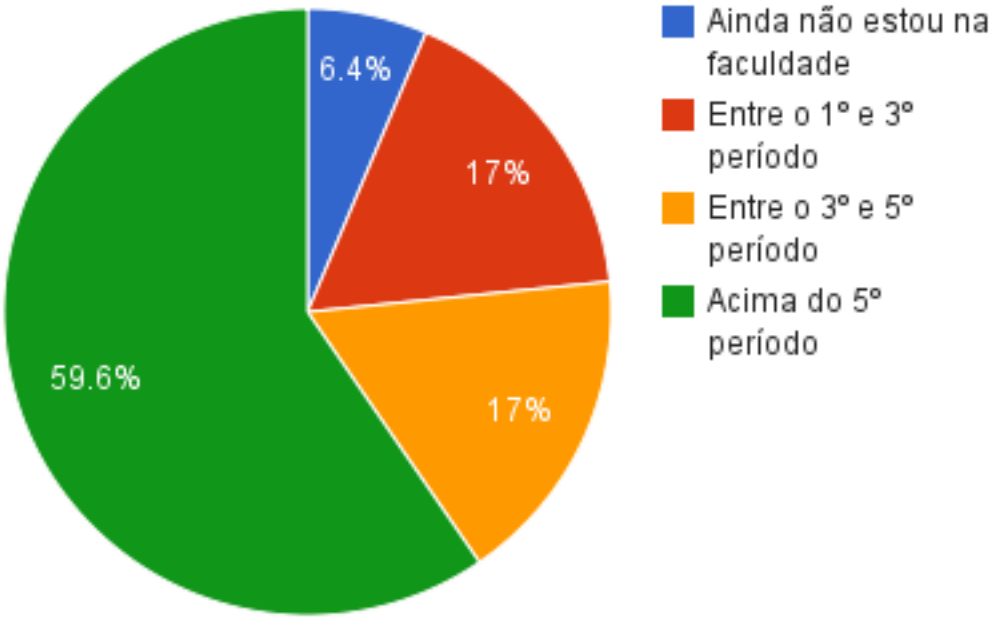

Fig. 3: Gráfico dos períodos dos estudantes

Dos profissionais, 20 (15.4%) com experiência de até 2 anos de desenvolvimento, 32 (24.6%) entre 2 e 4 anos, 39 (30%) entre 4 e 8 anos, 18 (13.8%) entre 8 e 12 anos e 21 (16.2%) com mais de 12 anos. Nessa questão, podemos ver uma distribuição um pouco mais normalizada com uma experiência geral próxima dos 4 anos de trabalho.

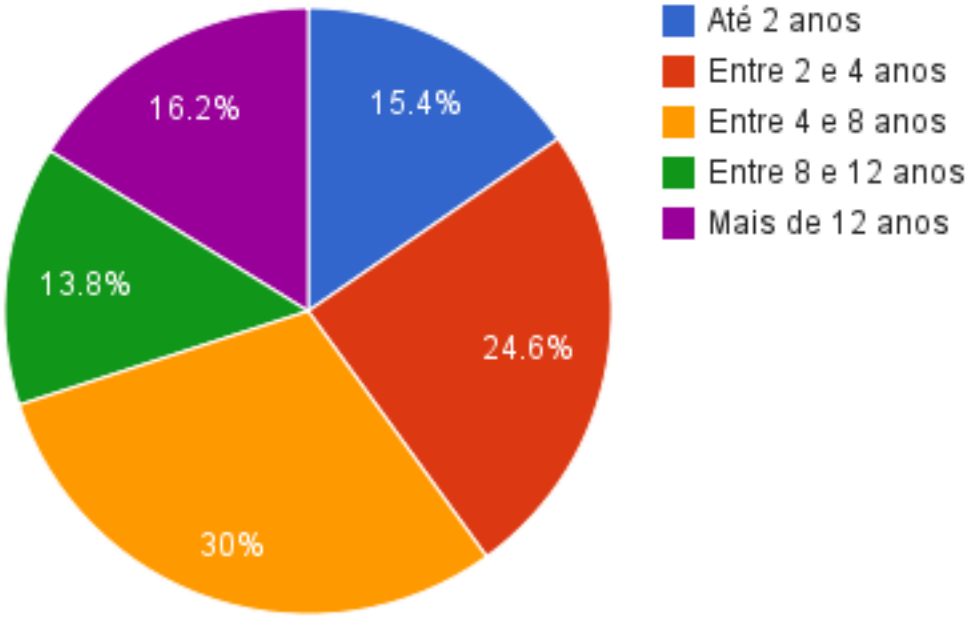

Fig. 4: Gráfico de experiência profissional

Para mapear a diversidade geográfica da pesquisa, uma pergunta opcional foi a cidade do entrevistado. De todos, apenas 15 (8.6 %) não responderam. Fazendo uma distribuição por estados, podemos detectar uma concentração maior de entrevistados no Rio de Janeiro com 90 (52.7%) pessoas, São Paulo com 21 (12.2%) e Rio Grande do Sul com 16 (9.3%). Depois dos três, temos outros estamos como Minas Gerais e Paraná com 8 (4.6%) cada, o Rio Grande do Norte com 6 (3.4%), Mato Grosso do Sul com 3 (1.7%), Espírito Santo com 2 (1.2%) e Distrito Federal, Goiás, Santa Catarina, Pernambuco e Bahia com 1 (0.6%) cada. Na Figura 5, por razões de espaço, estão sendo exibidas somente as cidades com maior número de participantes.

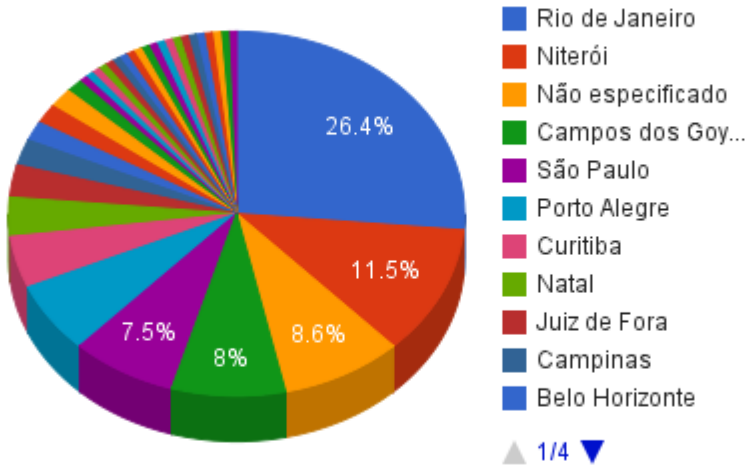

Fig. 5: Gráfico de cidades

Depois de mapear as características pessoais do entrevistado, partimos para o momento em que entendemos qual o grau de relacionamento dele com a prática do Coding Dojo. Essas perguntas são importantes para podermos mapear a experiência dos entrevistados com o Coding Dojo e também a maneira com que a prática é passada para as pessoas.

Sendo assim, a primeira pergunta realizada foi a de como o entrevistado conheceu o Coding Dojo. Quarenta e sete pessoas (27%) responderam que conheceram através de amigos que já participavam, 52 (29.8%) foram através de eventos de tecnologia, 24 (13.8%) em comunidades de desenvolvimento, 20 (11.5%) entre textos publicados e blogs e 19 (10.9%) por terem um Coding Dojo dentro de sua universidade. Por fim, 12 pessoas (6.9%) terem conhecido por outro meio. O interessante desse ponto é analisarmos que somando as pessoas

que conheceram por amigos, através de eventos e aquelas que possuem um encontro dentro da universidade, temos mais da metade das pessoas. Isso comprova que o maior vetor para trazer novos adeptos à prática é através do contato pessoal entre as pessoas.

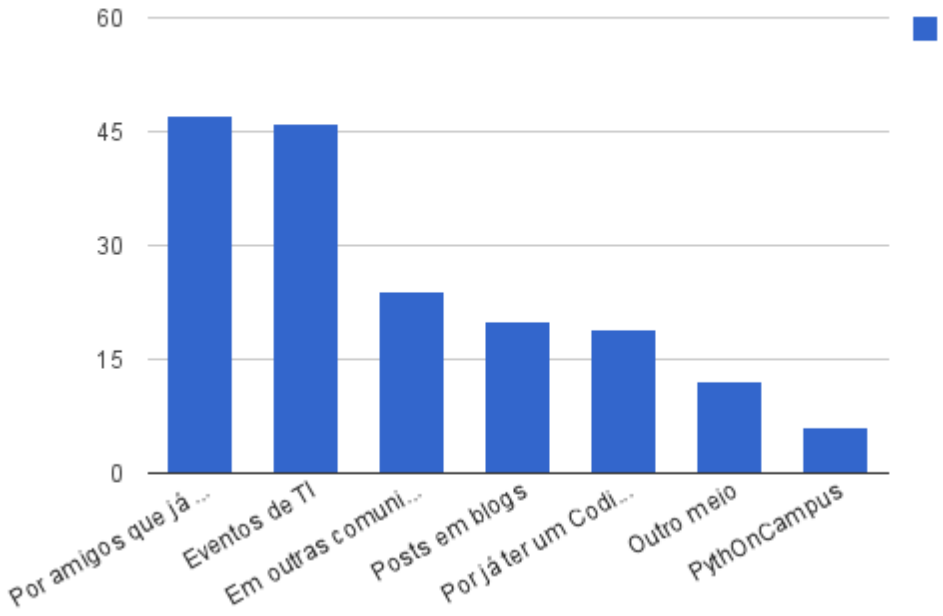

Fig. 6: Gráfico de como conheceu o Coding Dojo

Para medirmos o grau de envolvimento com o Coding Dojo, perguntamos qual o número de reuniões que o entrevistado já havia participado e também a frequência com que participa. Para a primeira pergunta, tivemos 71 (40.8%) participando de até 5 sessões, 31 (17.8%) entre 5 e 10 sessões, 13 (7.5%) entre 10 e 15 sessões, 12 (6.9%) entre 15 e 20 sessões e 47 (27%) em mais de 20 sessões. Sobre a frequência, tivemos 63 (36.2%) que participam bimestralmente, 39 (27.6%) mensalmente, 24 (13.8%) quinzenalmente e 48 (22.4%) semanalmente. Podemos ver então que a experiência dos entrevistados com o Coding Dojo é significativa, já que a maioria participou de um bom número de sessões e mantém assiduidade respeitável ao encontro.

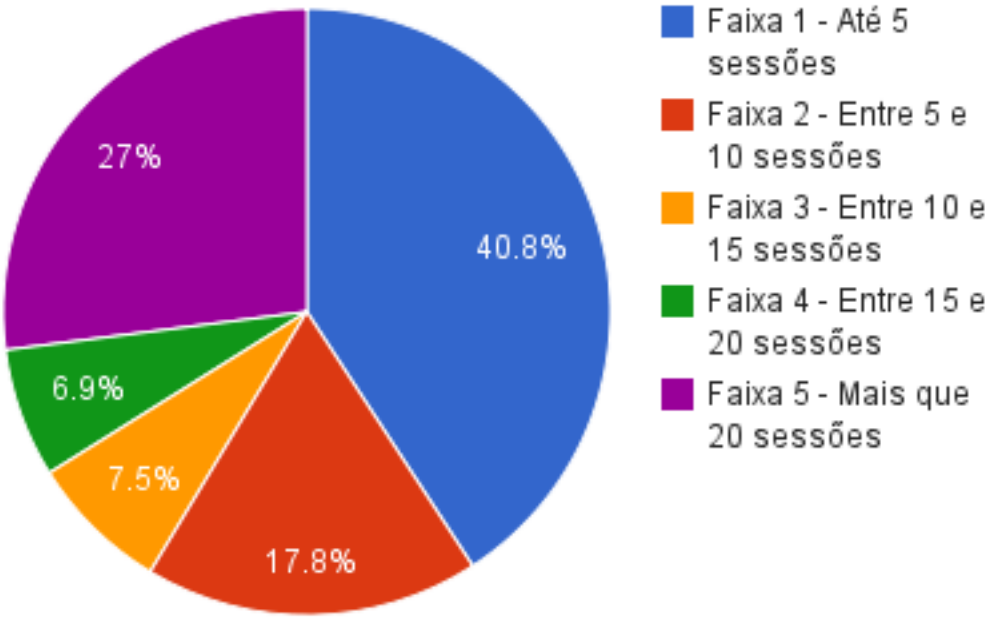

Fig. 7: Gráfico com número de participações

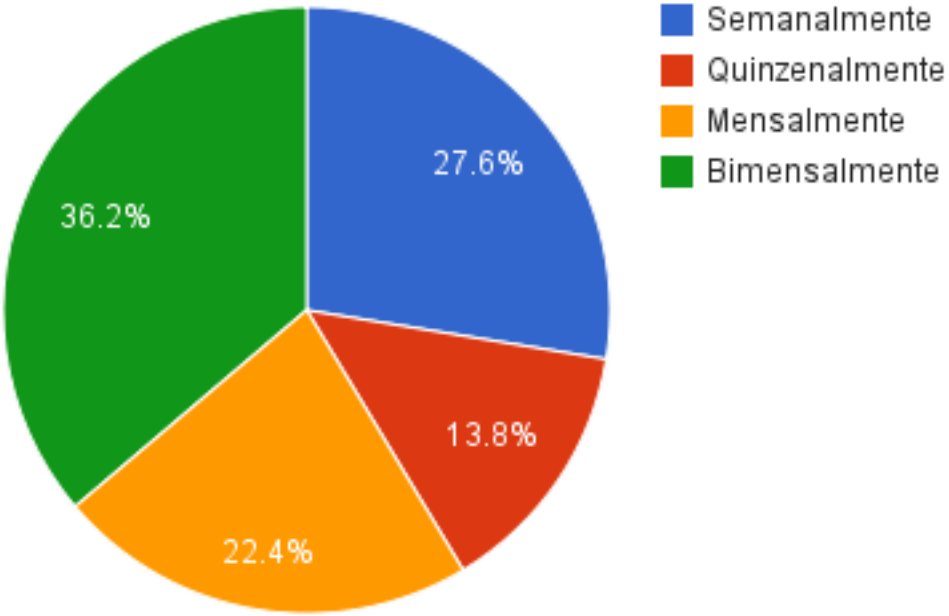

Fig. 8: Gráfico com frequência de participações

Para termos uma visão do que é considerado importante no Coding Dojo, principalmente para o ensino, foram feitas perguntas referentes a técnicas e pontos apresentados anteriormente e também a percepção de cada entrevistado sobre os potenciais da prática do Coding Dojo.

Para abordarmos esses tópicos, a primeira pergunta foi sobre quais pontos cada um julgava como importantes em um encontro. Vale dizer que essa pergunta não foi limitada para

apenas uma escolha, portanto, as porcentagens representadas para todas as respostas são referentes ao montante total de pessoas entrevistadas. Sendo assim, tivemos 136 pessoas respondendo que pessoas com conhecimentos diferentes é um ponto importante, 133 escolheram o TDD, 128 o Social Coding, 119 o Pair Programming, 108 a prática da refatoração constante, 101 tanto para o ambiente social quanto para o desenvolvimento utilizando o conceito de baby steps, 68 o foco nas pessoas, 68 a reunião de retrospectiva, 54 a percepção de progresso dada pelos testes e apenas 27 citaram os problemas escolhidos como ponto importante.

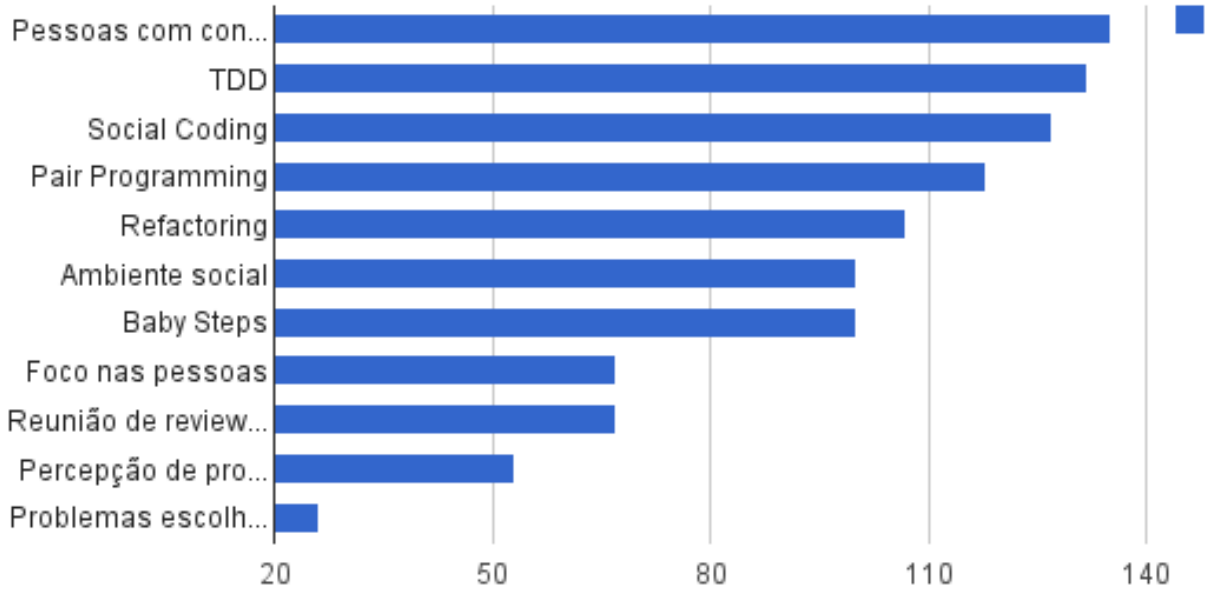

Fig. 9: Gráfico com os pontos importantes

A três perguntas seguintes consistiram em avaliarmos a importância dada ao Coding Dojo nos aspectos de formação profissional, fomentação de uma comunidade de desenvolvedores e como ferramenta de ensino, em um grau de 1 a 5, onde 1 é o grau de avaliação mais negativo e 5, o mais positivo. Para a primeira opção, a formação profissional, tivemos, 4 pessoas (2.3%) avaliando como grau 1, 15 (8.6%) como grau 2, 44 (25.3%) como grau 3, 50 (28.7%) como grau 4 e 61 (35.1%) como grau 5. No quesito fomentação de uma comunidade, teve-se 2 pessoas (1.2%) avaliando como grau 1, 5 (2.8%) como grau 2, 29 (16.7%) como grau 3, 43 (24.7%) como grau 4 e 95 (54.6%) como grau 5. No último ponto, ferramenta de ensino, tivemos 3 pessoas (1.8%) avaliando como grau 1, 4 (2.2%) como grau 2, 17 (9.8%) como grau 3, 52 (29.9%) como grau 4 e 98 (56.3%) como grau 5.

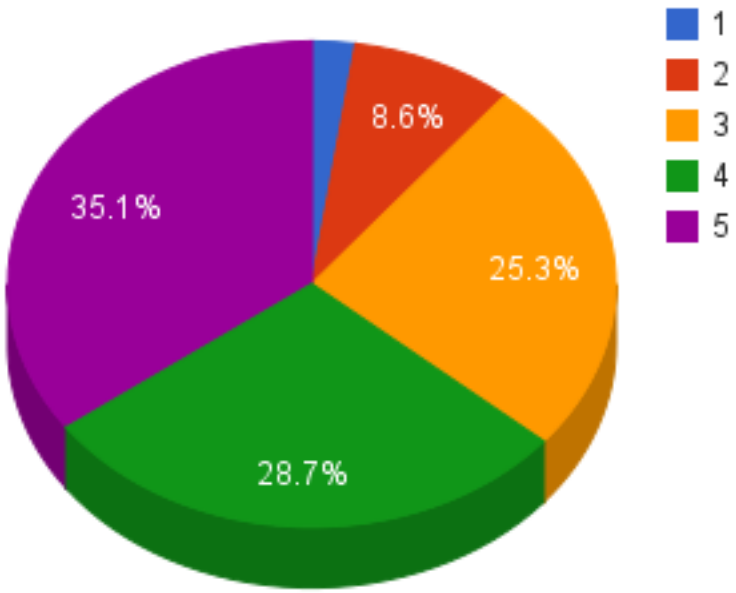

Fig. 10: Gráfico com a avaliação para formação profissional

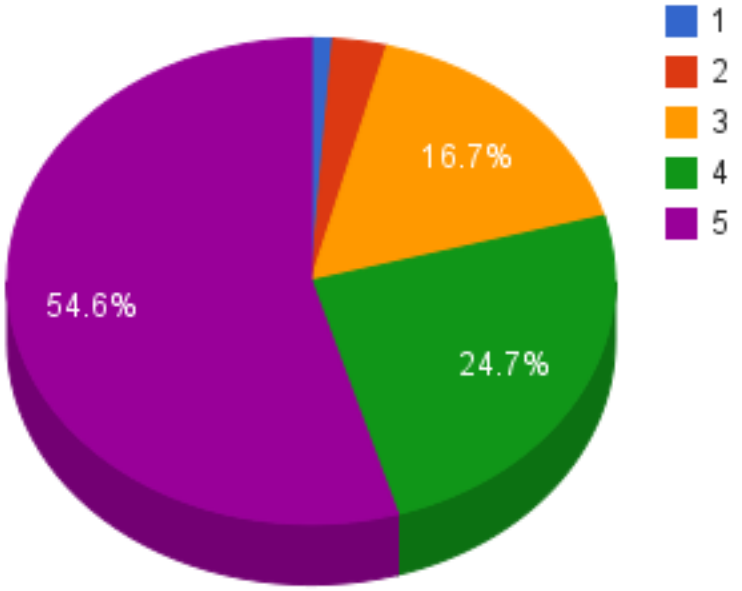

Fig. 11: Gráfico com a avaliação como fomentação de comunidades

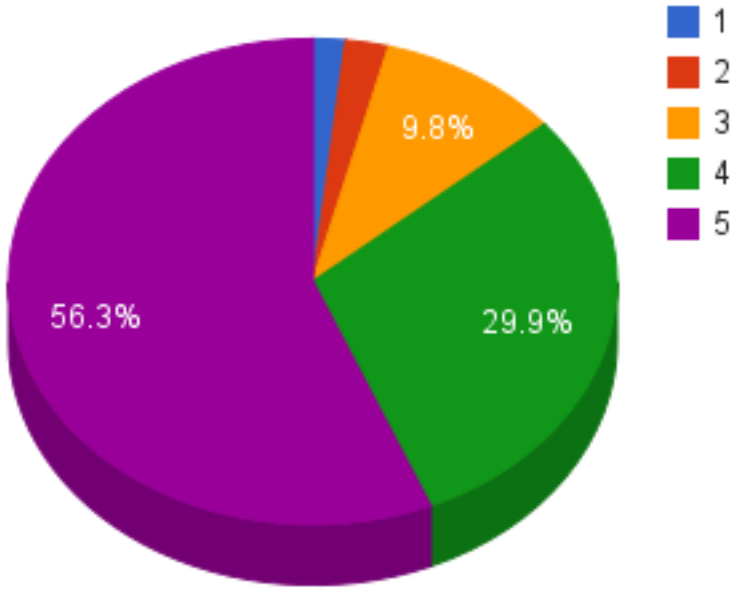

Fig. 12: Gráfico para avaliação como ferramenta de ensino

Dada essas três avaliações, podemos ver que a maior parte da percepção geral vê Coding Dojo como uma maneira muito boa, senão excelente, para fomentar o desenvolvimento de software em um grupo de pessoas. Isso impacta muito em como essas pessoas se relacionam e o que motivam elas a fazê-lo. Essa questão também será demonstrada com as próximas perguntas. Além disso, temos uma distribuição ainda mais otimista quando o assunto é o ensino, onde cerca de 86% dos entrevistados acreditam que temos em mão uma ótima ou até excelente ferramenta para ser utilizada de maneira didática.

Por fim, fazemos quatro perguntas para avaliar a ideia de que o perfil do participante de sessões de Coding Dojo é um perfil participativo, colaborativo e com o intuito de compartilhar o seu conhecimento com as outras pessoas. Assim, a primeira pergunta é sobre o número de comunidades de desenvolvimento de software que cada entrevistado participa. De todos os entrevistados, 29 (16.7%) disseram não participar de nenhuma comunidade, 16 (9.2%) disseram participar de apenas uma, 78 (44.8%) participam de uma a três, 26 (14.9%) de 3 a 5 e 25 (14.4%) participam de mais de 5 comunidades.

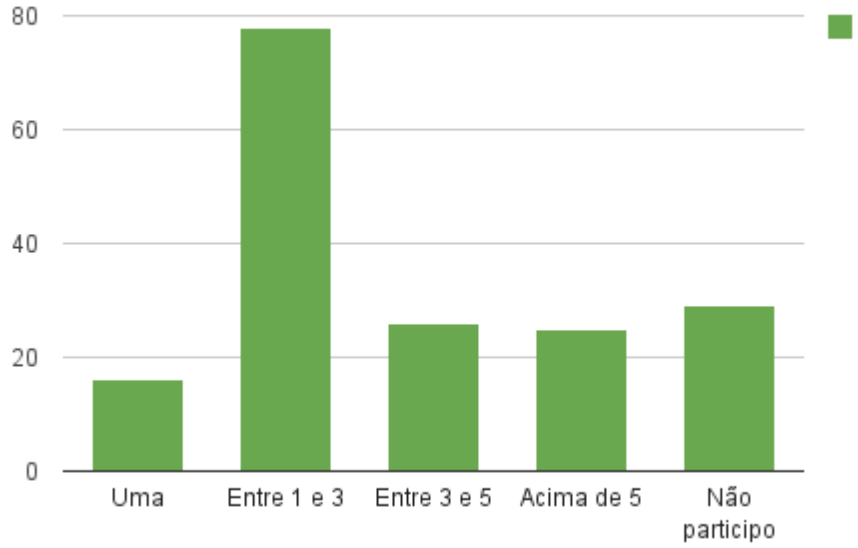

Fig. 13: Gráfico participação em comunidades

Somado a isso, perguntamos em quantos eventos de tecnologia ele participa por ano. 6 entrevistados (3.4%) disseram não participar de nenhum evento, enquanto 20 (11.5%) participam de pelo menos 1, 39 (22.4%) participam de 2 eventos em média, 37 (21.3%) de 3 em média, 16 (9.2%) de 4 em média e 56 dos entrevistados (32.2%) disseram participar, em média, em mais de 4 eventos por ano.

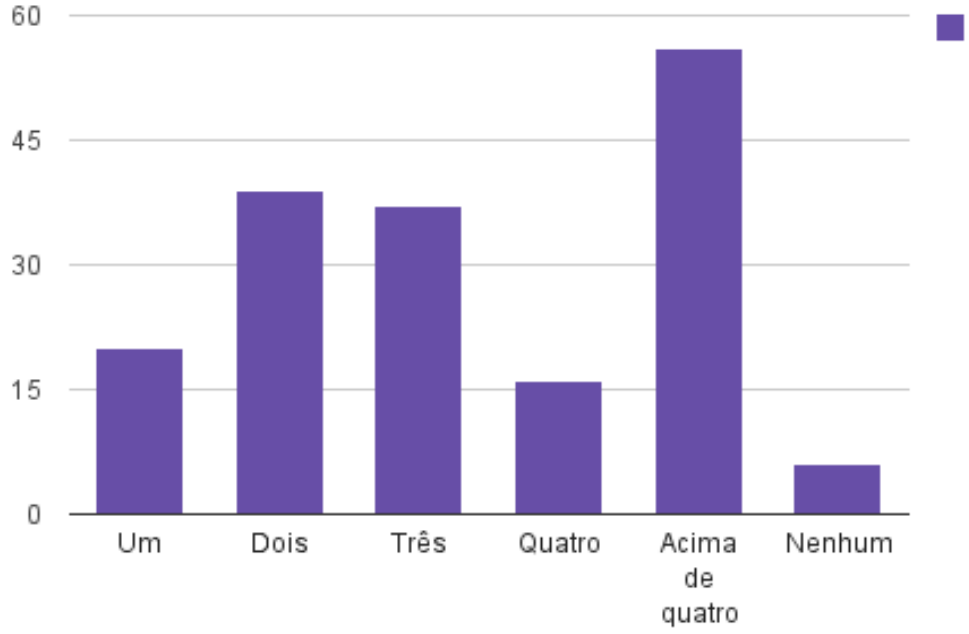

Fig 14: Gráfico com a participação em eventos

Avaliando conjuntamente esses dois dados, percebemos que o grau de participação em reuniões de pessoas, seja em eventos ou comunidades, é elevado, refletindo assim a característica participativa da amostra entrevistada. Outro ponto em que podemos apoiar isso é a na questão do uso e colaboração com o software livre. Dos entrevistados, temos que 168 (96.6%) utilizam software livre contra 6 (3.4%) que não utilizam. Mas, como apenas a utilização não é um critério forte para ser avaliado independentemente, temos que 111 dos entrevistados (63.8%) já colaboraram de alguma maneira com projetos open-source contra 63 (36.2%) que ainda não tiveram essa experiência.

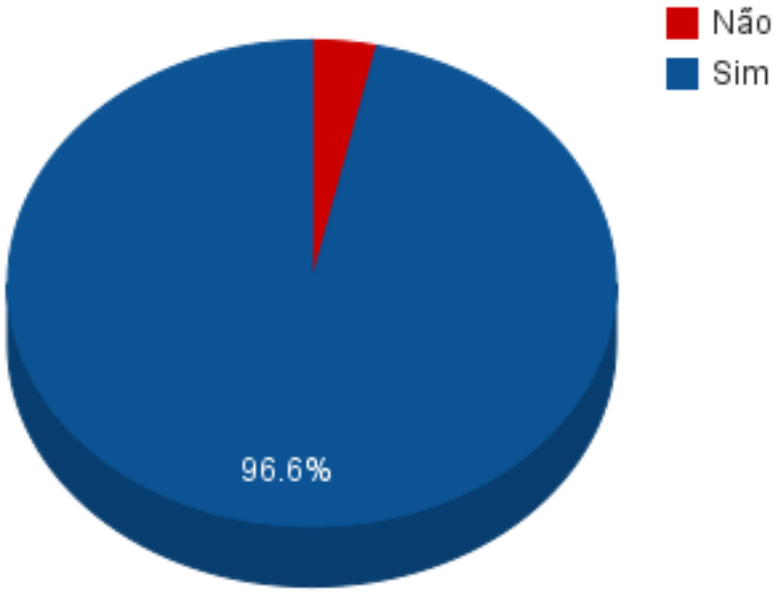

Fig. 15: Gráfico utilização de software livre

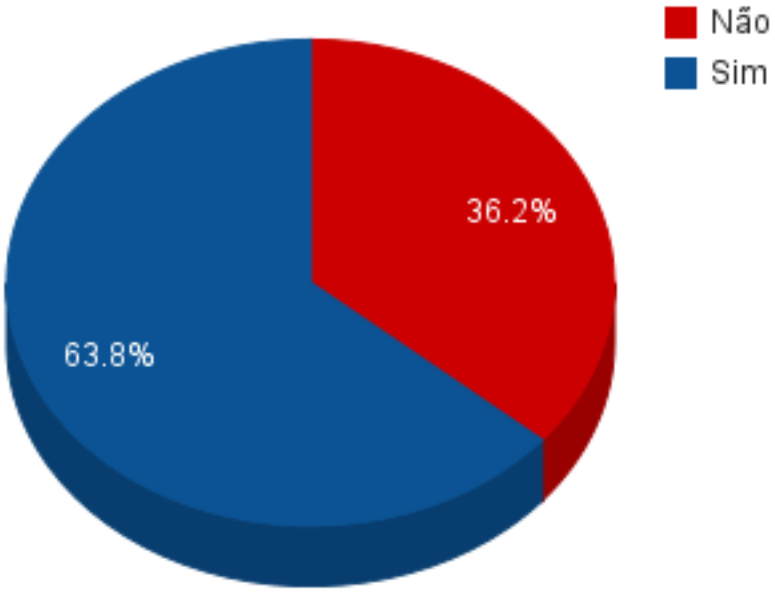

Fig. 16: Gráfico contribuição com software livre

Capítulo 5

O CODING DOJO NA UFF

Neste capítulo vamos abordar o Coding Dojo no contexto da Universidade Federal Fluminense. Primeiro será abordado a história para depois falarmos sobre duas consequências interessantes e diretas do encontro que é realizado na UFF.

5.1 A HISTÓRIA

Na Universidade Federal Fluminense, a prática do Coding Dojo começou quando o aluno do curso de Engenharia de Telecomunicações, Álvaro Justen, trouxe a experiência que ele havia vivido em um encontro da comunidade DojoRio (comunidade de praticantes do Coding Dojo no Rio de Janeiro). Isso foi no dia 22 de Outubro de 2009 (DOJORIO, 2009b). Desde essa data, o grupo de Coding Dojo que se encontra na UFF veio conquistando novos desafios e alavancou o seu patamar de ser apenas mais um encontro no estado do Rio de Janeiro para tornar-se um referencial de como implantar a prática de maneira sólida em outras universidades.

O começo dessa história se deu dentro da Sala de Seminários da Computação que era utilizada, de maneira não oficial, para que os encontros fossem realizados. O público variava entre 8 a 12 pessoas por encontro. Apesar de não haver uma variação grande no número de participantes, é importante dizer que esse número era praticamente composto pelos mesmos alunos da universidade. Esse fator foi fundamental que um grupo sólido de praticantes do Coding Dojo fosse cria dentro da universidade.

Após um começo ameno, o primeiro baque sofrido pelo grupo foi quando o uso da Sala de Seminários da Computação foi proibido para outras atividades que não fossem defesas de teses de pesquisa. Nesse momento o fato de o grupo de participantes da época ter sido formado de maneira sólida e também todos terem a prática do Coding Dojo como um grande interesse foi essencial para o encontro continuar existindo.

Fig. 17: Uma das primeiras sessões de Coding Dojo da UFF

Com a ajuda do Mário Mariani, na época, aluno do curso de Ciência da Computação da Universidade Federal Fluminense e participante do Coding Dojo, o laboratório do Núcleo de Tecnologia PROAC da universidade passou a ser utilizado para hospedar os encontros durante as férias de 2009 e início do ano de 2010. O impacto da mudança foi sobre o número de participantes que caiu um pouco, mas, novamente, isso foi interessante pois o fortalecimento das relações entre essas pessoas apareceria novamente como um fator importante na história mais a frente.

Em paralelo a essa mudança, o autor desta monografia e Andre Oliveira, na época também aluno do curso de Ciência da Computação da universidade, tentavam articular junto ao Instituto de Computação da Universidade Federal Fluminense uma possível sala que estivesse vaga no horário dos encontros para que pudesse ser utilizada. Na época, acreditava-se que era importante manter esse encontro perto dos alunos do curso. Era uma questão de expandir a experiência e os relacionamentos que estavam sendo construídos para que o maior número de alunos também pudesse viver o mesmo.

Nessa época, no meio do primeiro semestre de 2010, após muitas discussões sobre o futuro do Coding Dojo na universidade e sobre qual caminho que deveria ser adotado,

conseguiu-se o apoio do professor Leonardo Murta, do curso de Ciência da Computação. Com a ajuda de um professor, a sala foi disponibilizada para que os alunos a utilizassem durante o horário em que acontecia o encontro e desde então continua sendo utilizada frequentemente (DOJORIO, 2010a).

Após atingir a meta de trazer de volta o Coding Dojo para perto dos alunos, o grupo tinha um novo problema para resolver que era o de fazer com que os alunos soubessem e participassem do que estava acontecendo. Nessa época, novamente o autor desta monografia e o aluno Andre Oliveira receberam um convite das professoras Isabel Cafezeiro e Daniela Trevisan, ambas do curso de Ciência da Computação, para que preparassem palestras para os alunos da disciplina de Informática I que era ministrada para os novos alunos. O foco das apresentações eram as diversas comunidades de desenvolvimento de software existentes no Rio de Janeiro e o Coding Dojo. Como fruto dessas apresentações, dois pontos marcaram fundamentalmente essa história.

O primeiro foi a absorção dos novos alunos do curso com a ideia do Coding Dojo. A ligação foi tamanha que eles resolveram se reunir e começaram, por conta própria, uma sessão de Coding Dojo entre eles. A proposta era a de alinhar o conhecimento que eles iam adquirindo nas diversas turmas de programação que existiam e que eles cursavam. Afinal, como saber se eles estavam recebendo a mesma qualidade de informação se eles estavam em turmas diferentes? O autor desta monografia os auxiliou pelos primeiros meses fornecendo o que era necessário e servindo de guia pelos processos e práticas do encontro até o momento em que eles passaram a caminhar com as próprias pernas.

O segundo ponto marcante foi a invasão do antigo encontro na universidade pelos novos alunos do curso. Um público que antes variava entre 10 pessoas deu um salto para quase 30 em apenas uma semana. Essa invasão resultou em choque de cultura e conhecimento extremamente enriquecedor que resultou em sessões marcadas pelo ensino e pelo aprendizado (DOJORIO, 2010b).

Com esses dois acontecimentos, os professores começaram a perceber que havia algo acontecendo entre os alunos e os alunos estavam cobrando dos professores que eles entendessem e utilizassem o Coding Dojo na sala de aula. Por conta disso, alguns professores como a professora Vanessa Braganholo e o professor Dante Corbucci, começaram a procurar os alunos que participavam dos encontros na universidade para que estes os auxiliassem a adaptar a prática do Coding Dojo as suas aulas (CARVALHO, 2010).

O ápice desse processo de troca de experiências entre os alunos e os professores veio com um pedido de professora Alessandreia Julio, da Universidade Federal de Juiz de Fora, aluna de doutorado em Computação pela Universidade Federal Fluminense, para que o autor

desta monografia fosse mostrar aos alunos da Universidade Federal de Juiz de Fora o que estava acontecendo dentro da Universidade Federal Fluminense com o Coding Dojo. O resultado dessa viagem foi a criação de um novo encontro de Coding Dojo seguiu sendo mantido pelos alunos da própria universidade.

Ainda como reflexo do crescimento do número de participantes e pessoas envolvidas, no segundo semestre de 2010, o autor desta monografia foi novamente convidado pelas professoras Isabel Cafezeiro e Daniela Trevisan para falar aos novos calouros sobre o Coding Dojo. A reação foi tão positiva quanto a dos calouros do período passado. Eles também invadiram o antigo encontro que continuava a acontecer e também passaram a participar da sessão de Coding Dojo que os calouros do passado haviam criado.

Dado que a participação dos alunos havia sido consolidada, o grupo passou a estudar maneiras de potencializar ainda mais o processo do Coding Dojo. O grupo passou a analisar as sessões para identificar quais eram os problemas e os pontos que poderiam melhorar para o maior entendimento da dinâmica e dos conceitos de programação que surgiam nos encontros. Percebeu-se então que o conceito do estado dos testes, passando ou não passando, era uma barreira para os novatos entenderem de primeira. Foi então que, em uma madrugada, o autor desta monografia, junto a aluno Álvaro Justen, Eduardo Carvalho e Samir Cury, os dois últimos participantes da comunidade DojoRio, desenvolveram um semáforo para ser utilizado nos encontros. O semáforo foi desenvolvido utilizando o Arduino – microcontrolador eletrônico de código aberto – para fazer a comunicação entre o computador com o dojotools – ferramenta livre desenvolvida pelo Flávio Amieiro, membro da comunidade DojoRio, projetada para auxiliar sessões de Coding Dojo. Com o semáforo, o conceito de vermelho ou verde que representa o estado atual dos testes – falhando ou passando – que era subjetivo, agora passou a ser uma evidência física que tornava o entendimento mais fácil (JUSTEN, 2010).

Fig. 18: Bernardo Fontes e Álvaro Justen apresentando o semáforo no DevInRio 2010

Com a consolidação dos novos alunos no Dojo, os grupos de alunos mais antigos puderam soltar mais o controle dos encontros deixando, então, que os novatos se tornassem responsáveis por garantirem toda a infraestrutura para que a sessão se tornasse possível. Isso só pôde acontecer pela própria dinâmica do Coding Dojo de que não prevê um responsável, mas sim uma consciência de que é o coletivo que faz todo o sistema que envolve o Coding Dojo existente.

Em uma última iniciativa, o autor desta monografia e o aluno Thiago Garcia, tiveram a ideia de criar um vídeo com nomes importantes do mercado do desenvolvimento de software brasileiro em que cada um desse a sua opinião sobre a prática de Coding Dojo. Essa ideia surgiu poucos dias antes da semana de inscrição do primeiro semestre de 2011 do curso de Ciência da Computação e o ideal era apresentar para os novos alunos na semana de inscrição para motivá-los a também participar. Com a ajuda do Henrique Bastos, membro da comunidade DojoRio, os vídeos dos profissionais foram recebidos, agrupados e editados para compor o vídeo final.

Fig. 19: Coding Dojo na recepção aos calouros de 2011/1

O vídeo foi apresentado e o resultado foi gigantesco. Na própria semana de inscrição foi feita uma sessão de Coding Dojo mas, por causa do número de adeptos, teve que ser divida em duas sessões. O interessante é que, entre os futuros alunos, havia aqueles que nunca haviam visto uma linha de código sequer (BASTOS, 2011b). Como já era o esperado, eles continuaram a sua participação ao longo do período e o conceito de dojo trote, que foi criado nesse dia, foi repetido com os calouros do período seguinte.

É interessante analisar a história para podermos enxergar o caráter comunitário da mesma. Apesar de existirem ações pontuais de algumas pessoas, a história toda só pode ser construída através de uma experiência coletiva e não individual. Todo o intercâmbio de conhecimento e a evolução do conjunto só pode se dar através do compartilhamento.

5.2 CONSEQUÊNCIAS

A história da prática do Coding Dojo na UFF é longa e possuiu vários eventos marcantes, como foi visto anteriormente. Entre esses vários eventos, existem dois que valem a pena serem explorados e discutidos com mais atenção. Isso é válido por terem sido eventos que não se limitaram ao ambiente da universidade e extrapolaram para discussões que envolviam a academia e a comunidade que trabalha com computação em geral. Esses eventos foram decorrentes de encontros que aconteceram dentro da universidade e tornaram-se referência sobre Coding Dojo no aspecto nacional.

O primeiro acontecimento foi a publicação de “Apelo: parem de 'ensinar' Comp. I nas faculdades!” do Vinicius Teles em seu blog pessoal (TELES, 2010a). Nesse post, ele primeiramente introduz o seu ponto de vista alegando que, quando comparado a outros métodos mais prazerosos, o método tradicional de ensino mostra-se ineficiente, para compartilhar a experiência que ele viveu no Coding Dojo da UFF. Depois, ele aborda os pontos técnicos que são abordados e ensinados durante as sessões enumerando-os para fazer a comparação com o que é ensinado em uma aula de programação normal. O ponto principal do texto, é a análise dele sobre a utilização do Coding Dojo nas salas de aula, como pode ser visto neste trecho:

> *Em muitos cursos de computação os calouros têm alguma matéria do tipo "Comp. I" com duas aulas por semana, cada uma com duas horas de duração. Imagine se não houvesse aula, mas apenas Dojo. Então, teríamos duas sessões de Dojo por semana, durante um semestre inteiro. Se isso fosse adotado, tenho absoluta certeza de que os calouros chegariam no final do semestre sabendo programar mais e melhor que 90% daqueles que se formam em computação. E não só isso, eles saberiam programar mais e melhor que a maioria dos professores universitários que "ensinam" programação!*
>
> *Afinal, quantos professores dominam TDD e BDD, boas práticas de orientação a objetos, técnicas de refatoração, tudo isso em diferentes linguagens e ainda têm o hábito de programar em par e fazer retrospectivas? Garanto que poucos. E é aí que a minha ideia começa a ir por água abaixo.*
>
> *Porque para que ela fosse adotada, seria preciso, primeiro, vencer uma barreira cultural que me parece intransponível. A ineficiência do "cuspe e giz" aliada à insanidade das "provas" que não avaliam absolutamente nada, compõem uma muleta*

atrativa demais para deixar de ser usada. Mas, ainda que fosse possível vencer essa barreira, teríamos que lidar com um mar de professores que simplesmente não sabem programar bem.

O ponto de vista é relativamente extremista a partir do momento que não envolve somente a utilização do Coding Dojo e vai mais além quando ele trata as questões da qualificação dos professores das universidades e também do processo de avaliação das mesmas. Ainda assim, esse texto teve uma grande repercussão e gerou uma grande discussão acerca do tema tanto na comunidade acadêmica quanto no mercado.

O mais importante desse texto foi o fato de ter sido o primeiro a trazer esse tipo de discussão à tona de uma maneira duradoura. Isso é comprovado pelo número de comentários que foram feitos e pelo número de conversas que foram iniciadas em diversas listas e grupos na internet acerca da publicação. O interessante foi que ele serviu para evidenciar problemas no sistema de ensino tradicional e no próprio sistema do Coding Dojo para que ele possa ser utilizado por um professor durante as suas aulas.

Dentro da UFF, essa discussão foi tratada durante muito tempo e, como já citado anteriormente, o resultado foi a da aproximação dos professores com a prática do Coding Dojo e o início da utilização do mesmo como ferramenta de ensino.

O outro acontecimento derivado do Coding Dojo na UFF foi a utilização do modelo que foi criado nas outras universidades. O modelo de criação de um ambiente de ensino pelos próprios alunos e para os próprios alunos. Isso pode ser verificado pelo número de palestras que os participantes do grupo da UFF realizaram ao longo do tempo. Foram realizados eventos para propagar o modelo realizado na UFF em universidades de diversas cidades como Petrópolis, Juiz de Fora, São Paulo, Curitiba e também em outras do Rio de Janeiro e Niterói (DOJORIO, 2011a, 2011b).

Outro padrão de valorização da cultura que foi criada dentro da universidade é o interesse da mesma no processo. Com o passar do tempo, tornou-se recorrente o número de apresentações para os alunos sobre o Coding Dojo sendo feitas durante as aulas a convite dos próprios professores. A valorização dessa cultura e dessa comunidade construída também serviu para solidificá-la e conseguir exportá-la para outras universidades.

Um exemplo principal de referência dessa cultura foi o dojo trote e a sua repercussão (BASTOS, 2011a). Encontrou-se no dojo trote o ponto intermediário entre o trote tradicional e o chamado trote solidário. Quebrar o paradigma de que o veterano é alguém superior no primeiro dia é essencial para criar um relacionamento entre o grupo de veteranos e o grupo calouros que não seja baseado nos atalhos para provas e/ou matérias. A integração entre as

pessoas não se dá através de uma festa ou com potes de tinta e pincéis, mas sim através da troca de conhecimento e, principalmente, dúvidas entre os mais novos. Por fim, a conscientização de que é necessária a criação de uma comunidade de confiança logo no início da faculdade é uma lição não só para o período em que ela durar, mas também para a vida, já que esse conceito pode ser abstraído para outras situações. Então, o processo do dojo trote satisfaz tanto o âmbito social como o âmbito humano desejado pelos trotes.

Além do que foi mencionado anteriormente, o dojo trote auxilia os alunos a reduzir o susto do impacto com o novo e cria um ambiente que transpareça mais confiança à eles para fazerem perguntas. Muitos dos alunos ingressados nos cursos de computação não sabem o que é programar e nunca tiveram contato com a programação. Mostrar para esses alunos o que é um código no primeiro momento deles na faculdade é muito importante para já fazer despertar uma possível curiosidade de buscar mais sobre o tema. Não limitando somente a isso o ambiente menos institucional do que o da sala de aula motiva o aluno a fazer questionamentos que possivelmente ele não faria em uma sala de aula por sentir-se inibido com o professor ou com os novos colegas.

O processo do dojo trote foi uma experiência tão enriquecedora para todas as partes que, como citado na história, foi repetido com os calouros do período seguinte. Mas, como era de se esperar, essa prática não se limitou somente à UFF. Outras universidades de outras cidades como, por exemplo, Petrópolis e Campos dos Goytacazes, no estado do Rio de Janeiro, também adotaram a prática do dojo trote com resultados muito similares aos que ocorreram na UFF.

A utilização desse modelo de propagação de troca de conhecimento mostra-se novo ainda e, possivelmente, não tão maduro. O fluxo normal de expansão do público e do nome de uma universidade é através dos seus professores e alunos orientados, e dos projetos que são desenvolvidos por eles e apresentados em congressos e eventos gerais. Na lógica apresentada anteriormente, a expansão se dá pelos próprios alunos apresentando para outros um modelo que esses julgam interessante e que veio a funcionar na universidade em que estão. Afinal, um modelo que potencializa a capacidade de aprendizado e ensino é de total interesse da maioria dos alunos e da maioria dos professores.

Capítulo 6

PROPOSTAS PARA O USO DO CODING DOJO NA SALA DE AULA

Nesse capítulo, abordaremos alguns pontos para servirem como propostas para a utilização do Coding Dojo na sala de aula. Essas propostas são derivadas de experiências resultadas de utilização própria do Coding Dojo como atividade complementar ao ensino para alunos calouros do curso de Ciência da Computação no ano de 2010 e também de sugestões que foram dadas para que professores do curso pudessem utilizar a ferramenta dentro da sala de aula.

O processo de introdução do Coding Dojo na sala de aula pode ser feito de maneira gradual. Para isso, acredito que o formato do *PreparedKata* seja mais interessante pois ele não se distancia tanto do modelo que temos hoje. Nesse modelo, não existe a troca de papéis entre os participantes e o problema já é preparado anteriormente com soluções graduais. A não ser pela troca de papéis e pelo problema já previamente escolhido e com uma evolução resolvida, o processo é igual ao do *RandoriKata* (FERNANDO, 2011). Nesse formato o apresentador, no caso o professor, irá apresentando o problema, os testes e os passos para resolvê-lo de maneira gradual e, caso seja possível por questão de tempo, chegar à solução final.

Apesar de parecerem iguais, é importante explicitarmos que o processo atual de ensino de programação e o processo apresentado no parágrafo anterior, o *PreparedKata*, são diferentes. Ainda que os papéis do professor e do aluno se mantenham, o propósito final é diferente. Enquanto no primeiro existe a preocupação com a resposta ao problema, no segundo a preocupação é com o estudo da evolução do problema. Existe a preocupação com a forma com que o problema é apresentado garantido que os seus passos estão sendo respeitados, que o TDD está sendo utilizado, que o código e as dúvidas que surgem com o crescimento da resolução estão sendo discutidas. Então, ele apresenta as ferramentas e os princípios coletivos de um Coding Dojo sem romper com a estrutura atual de ensino.

Por garantir essa estrutura, esse modelo possui menos barreiras para ser implementado e é mais fácil enxergá-lo como um primeiro passo em direção ao uso do Coding Dojo. Entretanto, propor a migração para o modelo em que exista a alternância de papéis é importante por este fornecer uma experiência de interação direta do aluno com o problema proposto, como foi discutido nos capítulos anteriores. Sendo assim, acredito que o ideal, o que

gera menos gasto de energia na mudança, é a aplicação do Coding Dojo utilizando o *PreparedKata* mas com o foco em preparar os alunos e o professor para a utilização do *RandoriKata*, ou seja, para um modelo em que ambas as partes possam compartilhar de maior liberdade de experimentação. Sobre a manipulação do Coding Dojo, é válido o debate das questões que serão explanadas nos parágrafos abaixo.

Um dos questionamentos a serem levantados em utilizar o Coding Dojo em uma aula de programação é se o propósito de um curso de Ciência da Computação é atingido. Quando falamos de um curso como este que envolve além da parte tecnológica a parte matemática, existe uma intenção de capacitar os alunos a poderem atuar na área científica também. Então, o foco não é somente ensinar a programar, mas também passar todos os conceitos que estão embaixo daquela linguagem e da maneira que o aluno desenvolva o código para que ele não só tenha a capacidade de aplicá-los mas também saber quando são necessários.

Sendo assim, é frequente uma primeira argumentação de que o Coding Dojo prevê somente a parte do ensino da programação para o aluno. Entretanto, com a programação é possível experimentar a parte conceitual através de exemplos. Então, a primeira sugestão de adaptação é definir o tema a ser abordado. Por exemplo, suponhamos que seja uma aula de uma disciplina que aborde a Unidade Lógica Aritmética de um processador. Por que não explicitar o funcionamento da mesma abstraindo a sua lógica interna para um processo em que o aluno possa de fato construir uma ULA através da criação de código?

A questão nesse ponto é não limitar o assunto somente ao seu domínio. Gerar abstrações do assunto parece interessante porque aproxima o aluno do problema. Ao fazer isso, o aluno constrói seu próprio conhecimento a respeito do tema e reapresenta o problema aos colegas e ao professor, com suas palavras, com elementos do seu domínio. Ocorre então um movimento de realimentação em que todos 'reaprendem', ou seja, aprendem de uma outra forma. Poder interagir com esse conhecimento através de uma abstração representada em código agrega mais experiências para o aluno. Esse código, por sua vez, deve poder ser construído de maneira evolutiva para que os alunos possam fazê-lo dentro do processo do Coding Dojo. Assim, teremos um problema definido e implementado em uma linguagem, o que, por definição, já satisfaz alguns dos itens necessários para a realização do Coding Dojo.

Dado que a abstração do problema foi feita, precisamos também saber como vamos testar o código que será produzido. Dependendo da complexidade da abstração criada para o código, a própria ferramenta de testes utilizada já pode ser necessária para validar o funcionamento do código que estiver sendo implementado. Mas existem casos em que o domínio pode ser mais complexo e que a validação utilizando somente as capacidades que ferramenta de testes possui não atenda essa necessidade.

Esse problema pode ser resolvido pensando-se na representação do tema a ser discutido. Por exemplo, supondo que em uma aula de estrutura de dados o tema seja a construção de árvores binárias. Por ser uma estrutura de dados mais complexa, a validação da construção nó a nó pode tornar-se exaustiva. Para isso, podemos criar uma maneira de representar uma árvore binária como, por exemplo, com uma cadeia de caracteres. Poderia fazer parte do processo o professor e o os alunos criarem, juntos, métodos a serem utilizados durante a sessão que fossem responsáveis por montar a representação das árvores em uma cadeia de caracteres que seria, então, avaliada pelos testes.

Essa primeira solução pode parecer muito simples e pode não atender em alguns casos. Uma outra abordagem possível, seria a criação de uma camada adicional entre a ferramenta de testes utilizada e o domínio da matéria. Assim, os testes desenvolvidos pelos alunos ao invés de utilizarem a ferramenta de testes da linguagem, utilizariam essa nova camada que foi desenvolvida.

É interessante notarmos o papel do professor em um Coding Dojo dentro de uma sala de aula. Devemos ter em mente que o professor deve atuar como um facilitador, auxiliando a construção do conhecimento, e não como dono da solução. Por ser um ambiente de experimentação do que está sendo aprendido, os alunos estarão sujeitos a erros e a caminhos não idealizados pelo professor. Mas, nessa hora, é interessante que este tenha em mente que o caminho do erro também é um caminho válido dado o momento em que, a partir deste caminho, a solução também poderá ser revelada. Então, é importante o professor ter em mente que existem muitos caminhos a serem seguidos e que ele deve atuar como um guia conduzindo os alunos pelo caminho desejado e, quando tomado um caminho diferente, ajudá-los a retornar ao anterior. Ao agir como um bloqueio aos caminhos do erro, o professor estará privando o aluno da experimentação, do real aprendizado, e estará apenas utilizando um processo mecanizado para reforçar o conteúdo da disciplina.

Existe no modelo atual a necessidade da avaliação do aluno, pelo professor, com uma nota. Ou seja, é necessário quantificar o quanto aquele aluno aprendeu ou não sobre a matéria. Como eu disse no início do capítulo, delegar ao Coding Dojo essa responsabilidade é algo que vai além do intuito coletivo embutido na sua prática. Isso se dá, principalmente, pelo formato de avaliação em que o individual tende a ser mais valorizado. Portanto, acredito que as sessões durante as aulas funcionariam melhor quando utilizadas como ferramentas que potencializem a assimilação do conteúdo pelos alunos e não como um novo modelo de ensino completo.

Apesar do modelo de avaliação ser individual, é perigoso limitar o formato do ensino ao individual. Como apresentado nos capítulos anteriores, o Coding Dojo frequentemente

refere-se ao relacionamento interpessoal e a comunidade que o constrói. Então, é necessário criar uma própria comunidade dentro da sala de aula em que todos sejam respeitados. Isso se dá com a construção de uma consciência coletiva que aproxime os laços entre os alunos e entre os alunos e o próprio professor. No final, todos devem se enxergar como parte de um grande todo comum.

Como conclusão, temos que a utilização do Coding Dojo na sala de aula depende tanto de uma adaptação do processo como nas pessoas. Adaptar o domínio da matéria à um modelo cabível no Coding Dojo e a formação de uma pequena comunidade composta por alunos e professor fornecem uma base firme para a utilização da prática.

Capítulo 7

CONCLUSÃO

A necessidade de aprimoramento do processo de ensino deve ser uma preocupação constante em todos os profissionais que trabalham nesta área. Precisamos enxergar a troca de conhecimento como o principal vetor de construção de uma educação sólida. Fomentar a vivência, experiências, a integração entre as pessoas com o trabalho em grupo e a curiosidade pelo novo, faz parte da base que viabiliza essa construção. O Coding Dojo mostra-se como um processo que atende essas questões e pode ser encarado como uma possível grande ajuda na formação de novos desenvolvedores de software ou pesquisadores da área da computação.

Existem novos potenciais do Coding Dojo que podem ser explorados e podem ser temas de trabalhos futuros. Por exemplo, é interessante a maneira com que as pessoas envolvidas mantém uma atenção constante no projeto. É fácil perceber que existe um trabalho social além do trabalho técnico envolvido no Coding Dojo. Nesse trabalho desenvolvi uma análise sob o ponto de vista da educação, da troca de conhecimento, mas acredito que um questionamento envolvendo a perspectiva social do encontro seja bastante válido.

Um outro projeto a ser pensado é a extrapolação do Coding Dojo para outras áreas como, por exemplo, áreas humanas. Sabemos que a dinâmica do encontro é um processo que pode ser adaptado para outras situações. A lógica de criar um problema, resolvê-lo da maneira simples e depois resolvê-lo de uma maneira mais correta não se limita à área de programação. Portanto, seria interessante podermos estudar maneiras de implantar o Coding Dojo, por exemplo, no ensino fundamental, na construção de textos jornalísticos ou, até mesmo, em processos seletivos de empresas, e avaliar qual foi o seu desempenho.

Sobre avaliação de desempenho, o possível primeiro trabalho que venha depois desta proposta de utilização do Coding Dojo no ensino de programação é o estudo dos seus resultados. Analisar de maneira empírica os seus benefícios utilizando grupos de pesquisa. Por exemplo, podemos medir qual o desempenho do Coding Dojo quando o assunto for orientação a objetos. Separa-se um grupo que seja instruído utilizando-o e o outro que seja da maneira tradicional. Feito isso, cria-se um processo de avaliação prático e teórico que possa identificar possíveis falhas em ambos os sistemas.

Por ser uma prática nova – THOMAS (2007a) propôs os seus primeiros *katas* em 2003 – é provável que existam, além das já mencionadas anteriormente, várias outras situações em que o Coding Dojo possa ser aplicado, testado e validado que ainda não foram pensadas.

Por fim, é importante ter em mente que este é apenas mais um processo para potencializar o ensino. O Coding Dojo auxilia a resolver alguns dos problemas que enxergamos hoje no modelo de ensino das faculdades, mas ainda existem outros que não são tratados por ele. Enxergar o Coding Dojo como a resposta é criar a mesma ilusão que se tem hoje com o sistema tradicional. Precisamos sempre ter em mente que a mudança no sistema de ensino é um processo evolutivo que é composto pelo trabalho de várias pessoas utilizando várias ferramentas. Nesse contexto, o Coding Dojo mostra-se como uma ferramenta muito válida para ser utilizada.

REFERÊNCIAS BIBLIOGRÁFICAS

ALMEIDA, Raphael de. Evento no CEFET-RJ durante as férias. 2011. Disponível em: http://groups.google.com/group/dojo-rio/browse_thread/thread/94d549a76c3bbe3f/94a842baf804936f?show_docid=94a842baf804936f Acesso: 30/11/2011

ARTEM. Baby Steps. 2007. Disponível em http://agilesoftwaredevelopment.com/baby-steps Acesso: 05/12/2011

BASTOS, Henrique. Dojo Trote virou febre. 2011a. Disponível em: http://groups.google.com/group/dojo-rio/browse_thread/thread/76ff1af7da4a5f46/ae319ed50b546f9c?show_docid=ae319ed50b546f9c Acesso: 30/11/2011

BASTOS, Henrique. Trote na UFF = DojoRio + Educação 2.0. 2011b. Disponível em: http://henriquebastos.net/2011/03/02/trote-na-uff-dojorio-educacao-2-0/ Acesso: 30/11/2011

BOSSAVIT, Laurent. CodingDojoPrinciples. 2011. Disponível em: http://codingdojo.org/cgi-bin/wiki.pl?CodingDojoPrinciples Acesso 15/11/2011

BOSSAVIT, Laurent. Dojo. 2005. Disponível em: http://bossavit.com/dojo/ Acesso: 15/11/2011

BOSSAVIT, Laurent. Object Dojo. 2004. Disponível em http://web.archive.org/web/20071031005628/http://www.bossavit.com/pivot/pivot/entry.php?id=207 Acesso: 15/11/2011

CARVALHO, Rafael. DojoUFF – Turma de Prog. 2!. 2010. Disponível em: http://dojorio.wordpress.com/2010/06/22/dojouff-turma-de-prog-2/ Acesso: 15/11/2011

CASTAÑON, Gustavo Arja. Construtivismo Social: A ciência sem sujeito e sem mundo. 2009

CUKIER, Daniel. Coding Dojo. 2009. Disponível em: http://blog.locaweb.com.br/tecnologia/coding-dojo/ Acesso: 05/12/2011

DOJORIO. Coding DojoTreinamento para programadores utilizando TDD (Desenvolvimento Orientado a Testes). 2009a. Disponível em: http://apoie.org/Dojo.html Acesso: 15/11/2011

DOJORIO. Dojo dos calouros em Petrópolis. 2011a. Disponível em: http://dojorio.wordpress.com/2010/08/30/dojo-dos-calouros-em-petropolis/ Acesso: 30/11/2011

DOJORIO. DojoRio Em Niterói Foi Sensacional. 2009b. Disponível em: http://dojorio.wordpress.com/2009/10/24/dojo-rio-em-niteroi-foi-sensacional/ Acesso: 15/11/2011

DOJORIO. Dojo@IFF Especial de Ano Novo (Letivo). 2011b. Disponível em: http://dojorio.wordpress.com/2011/03/24/dojoiff-especial-de-ano-novo-letivo/ Acesso: 30/10/2011

DOJORIO. DojoRio@Niterói - 13/05/2010. 2010a. Disponível em: http://dojorio.wordpress.com/2010/05/16/dojorioniteroi-13052010/ Acesso: 15/11/2011

DOJORIO. DojoRio@Niterói - 20/05/2010. 2010b. Disponível em: http://dojorio.wordpress.com/2010/05/24/dojorioniteroi-%E2%80%93-20052010/ Acesso: 15/11/2011

FERNANDO, Thales. Coding DOJO (CoDOJO) – O Manual de Sobrevivência. 2011. Disponível em http://blog.oxylabtech.com.br/?tag=laurent-bossavit Acesso: 15/11/2011

FREEMAN, S. e PRYCE, N. Growing Object-Oriented Software, Guided By Tests, Chapter 1. 2010a

FREEMAN, S. e PRYCE, N. Growing Object-Oriented Software, Guided By Tests, Chapter 2. 2010b

FONTES, Bernardo. Pesquisa Coding Dojo. 2011. Disponível em: http://www.bernardofontes.net/blog/pesquisa-coding-dojo-2/ Acesso: 15/11/2011

FOWLER, Martin. Refactoring: Improving the Design of Existing Code. 1999

JUSTEN, Álvaro. Semáforo do Coding Dojo Rio 2010. Disponível em: http://blog.justen.eng.br/2010/10/semaforo-do-coding-dojo-rio.html Acesso: 15/11/2011

KERTH, N. L.. Project Retrospectives: A Handbook for Team Reviews. 2001.

NEVILLE, A.J. Problem-Base/d Learning and Medical Education Forty Years On. Medical Principles and Practice, 18, 1-9. 2009

PIAGET, J. Psicologia e Epistemologia: por uma teoria do conhecimento. Rio

de Janeiro: Forense - 1973

REHM, James. Problem-Based Learning: An Introduction. 1998. Disponível em: http://www.ntlf.com/html/pi/9812/pbl_1.htm Acesso: 15/11/2011

ROSA, Paulo Ricardo da Silva. Instrumentação Para o Ensino de Ciências - Capítulo 3: A Epistemologia Genética de Piaget e o Construtivismo. 2010

SATO, D. T., CORBUCCI, H., e BRAVO, M. V. Coding dojo: an environment for learning and sharing agile practices. AGILE Conference, 0:459–464. 2008

SATO, Danilo. Dojo: Treino para Programadores. 2007. Disponível em: http://www.dtsato.com/blog/2007/08/27/dojo-treino-para-programadores/ Acesso: 15/11/2011

TELES, Vinícius. Apelo: parem de "ensinar" Comp. I nas faculdades!. 2010a. Disponível em: http://blog.improveit.com.br/articles/2010/05/28/apelo-parem-de-ensinar-comp-i-nas-faculdades Acesso: 15/11/2011

TELES, Vinícius. Parem de ensinar Comp. I nos cursos de computação. 2010b. Disponível em: http://groups.google.com/group/dojo-rio/browse_thread/thread/621bca4265b09eb6/a92e74d158d4d840 Acesso: 15/11/2011

THOMAS, Dave. Code Kata--How It Started. 2007a. Disponível em: http://codekata.pragprog.com/2007/01/code_katahow_it.html Acesso: 15/11/2011

THOMAS, Dave. Code Kata. 2007b. Disponível em: http://codekata.pragprog.com/2007/01/code_kata_backg.html Acesso: 15/11/2011

THOMAS, Dave. Kata, Kumite, Koan, and Dreyfus. 2007c. Disponível em: http://www.codekata.com/2007/01/kata_kumite_koa.html Acesso: 15/11/2011

WILLIAMS, L., KESSLER, R. R., CUNNINGHAM, W. e JEFFRIES, R. Strengthening the Case for Pair-Programming. 2000

WOOD, Diana F. ABC of learning and teaching in medicine: Problem based learning. 2003. Disponível em: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1125189/ Acesso: 15/11/2011