XV ENCONTRO NACIONAL DE ENGENHARIA E DESENVOLVIMENTO SOCIAL
12 a 14 de novembro de 2018
Alagoinhas- BA, Brasil

Software Livre e Metodologias Participativas – Desafios de uma disciplina na graduação e no mestrado

Celso Alvear
celsoale@gmail.com
Núcleo Interdisciplinar para o Desenvolvimento Social / Universidade Federal do Rio de Janeiro – Rio de Janeiro, Brasil.

Pedro Braga
pedrohcb@ufrj.br
Universidade Federal do Rio de Janeiro – Rio de Janeiro, Brasil.

Antonio Claudio Gómez de Sousa
ac@poli.ufrj.br
Universidade Federal do Rio de Janeiro – Rio de Janeiro, Brasil.

RESUMO

Este artigo tem como objetivo relatar a experiência da disciplina Software Livre e Metodologias Participativas, ministrada no Departamento de Engenharia Eletrônica (DEL/POLI) da UFRJ desde 2011. Esta disciplina que originalmente é optativa para alunos de cursos de Engenharia Eletrônica, de Computação, do curso de Ciência da Computação, a partir de 2016 também se tornou optativa para o Mestrado Profissional em Tecnologia para o Desenvolvimento Social. Ao longo desses anos, foram 6 turmas já finalizadas, 4 na graduação, uma combinando mestrado e graduação e uma apenas com alunos do mestrado. A disciplina é composta por uma primeira metade teórica, e outra com foco em trabalhos práticos de desenvolvimento de software a partir da demanda de grupos sociais. Como principais resultados, a disciplina gerou softwares que estão em uso, teve como desdobramento alguns projetos de extensão e permitiu que os alunos vislumbrassem outras possibilidades profissionais através das reflexões sobre software livre, métodos ágeis, design participativo e tecnologias sociais.

PALAVRAS-CHAVE: Ensino de Engenharia; Engenharia de Software; Software Livre; Design Participativo; Ciência, Tecnologia e Sociedade; Métodos Ágeis

INTRODUÇÃO

Os cursos de computação são conhecidos por terem uma formação muito tecnicista, com pouco diálogo com as ciências humanas. Isso se dá, em geral, por conterem uma grade curricular com muitas disciplinas específicas da área de computação, com forte cunho matemático e quantitativas, e poucas disciplinas de outras áreas do conhecimento, com abordagens humanísticas, qualitativas e reflexivas.

Dentro desses cursos, uma das áreas de conhecimento é a Engenharia de *Software*, criada por volta dos anos 1970, dentro de um contexto de grandes projetos de software que necessitavam técnicas e métodos para gerenciar sua complexidade, que envolvia muitos recursos, pessoas e instituições diferentes. Muitos desses projetos aconteciam em ambientes de grandes corporações ou em ambientes militares, e o paradigma que mais se desenvolveu na época foi o Cascata (ou *Waterfall*), que segue uma lógica linear, na qual o processo de desenvolvimento é dividido em uma sequência de etapas que se iniciam uma após o término anterior.

A Engenharia de *Software* trata de aspectos da gestão do desenvolvimento de software, e portanto possui uma perspectiva interdisciplinar, já que envolve muitos aspectos para além dos ditos técnicos. Porém, pelo perfil dos cursos de computação apresentados acima, muitas vezes a Engenharia de Software é apresentada de forma superficial, focando muito mais nos aspectos ditos técnicos, mas com pouca formação na parte de interação com os usuários, na parte de levantamento de requisitos, interação humano-computador e em todos os aspectos interdisciplinares da Engenharia de Software.

Por outro lado, recentemente, no cenário internacional, a Engenharia de Software vem mostrando preocupação crescente com aspectos interdisciplinares, com a questão das interfaces e das técnicas de levantamento de requisitos com usuários diversos e multiculturais (SOMMERVILLE, 2016; PRESSMAN, 2016). Além disso, pela disseminação cada vez maior da informática entre usuários leigos, é fundamental o desenvolvimento de técnicas de interação que permitam melhor compreensão das demandas de usuários que não possuem conhecimentos em informática e têm dificuldade de expressar seus desejos em um linguajar técnico. Essa é uma das grandes preocupações do Design Participativo (SCHULLER & NAMIOKA, 1993).

Ao mesmo tempo, o uso de novos paradigmas da Engenharia de *Software* como os Métodos Ágeis vem crescendo, a partir da adoção destes por muitas empresas. Estes métodos permitem uma maior interação com usuários e a possibilidade de diminuir erros no projeto do sistema. Em um contexto de desenvolvimento web, com mudanças cada vez mais rápidas, os métodos ágeis tendem a ser uma metodologia que produz resultados com mais qualidade e menor retrabalho (HIGHSMITH et al, 2001).

Outra tendência é o crescimento do *software* livre. Através da disponibilização do código-fonte, esse paradigma permite que muitos possam contribuir na melhoria do software, garantindo assim sua qualidade por uma validação coletiva (STALLMAN, 2012). Hoje o mercado de software livre concorre de igual pra igual com o mercado de software proprietário, como é o caso do sistema operacional Linux e do servidor web Apache, o mais usado para hospedar sites no mundo inteiro (GREENSTEIN & NAGLE, 2014). Outro exemplo é o software WordPress, responsável por 27% de todos os sites do mundo (PRIMO, 2017, p. 27).

Neste artigo, descreveremos a disciplina *Software* Livre e Metodologias Participativas, oferecida pelo Departamento de Engenharia Eletrônica da UFRJ, que aborda as questões mencionadas acima e busca novas formas de ensino de Engenharia. Na Seção 2, descreveremos a metodologia de ensino-aprendizagem, incluindo seus conceitos filosóficos e os métodos de trabalho e avaliação. Em seguida, descrevemos na Seção 3 os eixos temáticos cobertos no programa. A Seção 4 apresenta dois dos projetos trabalhados em sala de aula e a Seção 5 discute alguns dos resultados alcançados na disciplina e os desafios para o futuro.

METODOLOGIA

O objetivo da disciplina é analisar metodologias participativas de desenvolvimento de software, apresentar a filosofia do *software* livre como uma forma de desenvolvimento coletivo de *software*, e refletir sobre as possibilidades de desenvolver um outro tipo de tecnologia que possa contribuir diretamente para o desenvolvimento social.

Para isso, o percurso de estudo e a avaliação de desempenho dos estudantes segue as seguintes ações: (1) Elaboração de resenhas críticas[1] dos textos de apoio aos temas; (2) Presença e participação nos debates e atividades em sala; e (3) Projeto envolvendo levantamento de requisitos, modelamento e desenvolvimento de software livre utilizando metodologias participativas.

A disciplina normalmente era desenvolvida em 30 aulas, sendo duas aulas por semana de duração de 2 horas cada[2]. Na turma de 2016, na qual tinham alunos de graduação e mestrado juntos, mudamos para 15 aulas com quatro horas por semana (e como o mestrado tem um período trimestral, combinamos com os alunos do mestrado para participarem da disciplina seguindo o periodo da graduação semestral). Por fim, na turma de 2017,

1 Nas ultimas duas turmas, experimentamos usar reações (250 palavras) ao invés de resenhas criticas (1000 palavras), de forma a diminuir a carga de trabalho e dar mais tempo para os alunos desenvolverem os projetos.
2 O programa da disciplina, contendo a previsão de conteúdo das aulas e a bibliografia completa pode ser acessado em http://bit.ly/1fmLyhy.

exclusivamente para alunos de mestrado, foram 12 aulas de 4 horas, uma vez por semana.

Outro elemento importante é que a disciplina é dada por uma dupla de professores em sala de aula, o que permite a exposição de divergências, ajudando na desconstrução de uma ciência neutra e do professor como alguém que apresenta verdades. Além disso, a disciplina é dada em uma sala que possui uma grande mesa no centro e uma bancada que ocupa todas das três paredes da sala com em torno de 25 computadores. Na quarta parede tem um grande quadro branco e é onde pode se projetar o datashow quando necessário. Podemos dividir a disciplina em 3 tempos pedagógicos, que serão descritos a seguir:

TEMPO-TEORIA

Na primeira metade da disciplina, que chamamos tempo-teoria, as aulas são orientadas por debates acerca de textos selecionados previamente. É requisitada a entrega de resenhas críticas dos textos antes do debate, como forma de estimular a leitura. As aulas são iniciadas pelos comentários dos alunos sobre texto. A partir destes comentários, os professores buscam facilitar o debate e aprofundar as questões trazidas.

Os textos são divididos em quatro eixos: (1) Engenharia de *Software*; (2) *Software* Livre; (3) Design Participativo; e (4) Estudos em Ciência, Tecnologia e Sociedade. Em cada uma das quatro primeiras aulas é debatido um texto introdutório sobre cada um destes temas e, a partir da quinta aula, cada um dos temas é aprofundado sequencialmente.

Ainda no tempo-teoria, se inicia a transição para o tempo-prática. São apresentadas algumas possibilidades de desenvolvimento de software dentro da perspectiva da disciplina, a partir de demandas de movimentos sociais ou de órgãos públicos que desenvolvam alguma ação voltada para o desenvolvimento social. A turma é dividida em grupos de no máximo 10 integrantes e cada grupo (Time) seleciona uma demanda para ser trabalhada. A forma dos grupos trabalharem é inspirada na metodologia *Scrum* (SCHWABER & SUTHERLAND, 2013), apresentada em mais detalhes no item 3.1 deste artigo. O foco do trabalho deve ser o levantamento de requisitos e modelagem do sistema demandado, mas sempre que possível busca-se desenvolver pelo menos as funcionalidades essenciais para o demandante.

TEMPO-PRÁTICA

A partir da segunda metade da disciplina o foco é quase exclusivamente no projeto[3]. Os grupos usam o horário para se

3 Ao longo dos anos, os alunos pediram pra iniciar o quanto antes o projeto, então antes da metade já vamos intercalando aulas teóricas e prática.

reunir, planejar o desenvolvimento do sistema, para documentar os requisitos levantados e para fazer as modelagens do sistema. Além disso, podem ocorrer reuniões com os demandantes no horário de aula na sala, no local do demandante com parte do grupo, ou em muitos casos essas reuniões ocorrem fora do horário de aula a partir da agenda desses demandantes.
A cada fim de um ciclo (*Sprint*), que dura de três a quatro aulas, o grupo faz uma apresentação rápida de 15 minutos sobre os resultados atingidos naquele ciclo e planejam o próximo ciclo. Em cada ciclo, dois a três estudantes assumem o papel de coordenação do grupo (*Scrum Masters*) e são responsáveis por garantir a coordenação de tarefas e uma boa comunicação dentro do grupo entre uma aula e outra. Por fim, no último dia da disciplina, os demandantes são convidados e os alunos apresentam e entregam os resultados do trabalho para que estes possam se apropriar deles e dar continuidade.

TEMPO-AVALIAÇÃO

Ao se tentar romper com os paradigmas tradicionais e bancários da educação (FREIRE, 2005), um dos maiores desafios certamente se coloca no momento da avaliação. O maior motivo é que neste ponto, a autonomia da relação educador-educando em sala de aula se rompe para dar uma resposta objetivo às estruturas de ensino. O professor de uma disciplina formal precisa traduzir em uma nota o resultado de uma relação de 4 meses com os estudantes.

Como forma de contornar esta exigência e dar novos sentidos à avaliação, buscou-se nesta disciplina uma forma de ponderar 3 aspectos para compor a nota final de um/a estudante:

- A avaliação do/a próprio/a educando/a sobre seu desempenho;
- A avaliação dos demais educandos/as sobre seu desempenho;
- A avaliação dos educadores sobre o seu desempenho.

Todas estas avaliações são feitas de forma subjetiva e objetiva. Portanto, além da composição numérica, cada estudante reflete sobre o processo de forma escrita, e também recebe uma resposta subjetiva sobre sua atuação. Além destas três avaliação, há a ainda a avaliação dos/as estudantes sobre os professores. Esta avaliação, no entanto, é anônima e não compõe a nota final.[4]

É necessário ressaltar o caráter pedagógico da autoavaliação e da avaliação dos pares. Mais do que um simples julgamento, o tempo-avaliação busca uma reflexão final sobre o processo

4 Há um modelo de avaliação que é entregue a um representante dos alunos, que junto com a turma podem ajustar conforme acharem melhor. Este é passado pra todos por um formulario web no qual só esse representante tem acesso a todas as respostas. Esse aluno consolida essas respostas e apresenta na ultima aula, na qual é aberto um debate de avaliação da disciplina, dos professores e da turma.

vivenciado, de forma a sistematizar os resultados do método de ensino e ressignificar a nota final de uma disciplina. Essa forma de avaliação também é importante pois esses alunos costumam estar nos últimos períodos, prestes a se formar e se inserir no mercado de trabalho, e essa avaliação serve como uma orientação para conhecerem seus pontos fortes e suas necessidades de melhoria em relação a sua atuação em uma equipe de trabalho.

Além disso, também buscamos ressignificar o processo de aprovação/reprovação, que tradicionalmente é diretamente ligado à nota final. No início das aulas, os estudantes são avisados de que serão aprovados caso cheguem ao final da disciplina participando de todas as aulas e atividades. Devido ao forte caráter participativo das atividades, consideramos que, ao chegar ao final do processo, podemos considerar que o estudante vivenciou ativamente a experiência proposta e, portanto, será "aprovado".

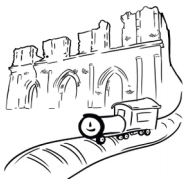
EIXOS TEMÁTICOS DA DISCIPLINA

Nesta seção, serão descritos os eixos temáticos da disciplina. A escolha dos eixos se deu no processo de construção do curso, e cada eixo se define em oposição ao ensino tradicional na Engenharia. O eixo de Engenharia de *Software* busca trazer a visão geral do desenvolvimento, ao contrário do tradicional foco em técnicas de programação. O eixo de S*oftware* Livre contrasta com a lógica proprietária do conhecimento, enquanto o eixo de design participativo contrapõe a visão tecnocrática normalmente reforçada. Finalmente, o eixo Ciência Tecnologia e Sociedade (CTS) busca romper a lógica da tecnologia neutra e situar a atuação dos engenheiros e engenheiras na sociedade.

INTRODUÇÃO À ENGENHARIA DE *SOFTWARE*

A Engenharia de *Software* tem por objetivo tratar problemas de grande complexidade na Tecnologia da Informação. Para isso, utilizam-se técnicas que buscam estabelecer processos, métodos, técnicas e ferramentas para a redução da complexidade no desenvolvimento e facilitar as manutenções de sistemas de software.

> A experiência inicial na construção desses sistemas mostrou que o desenvolvimento informal de software não era suficiente. Projetos importantes apresentavam, algumas vezes, anos de atraso. O *software*, cujo custo superava as previsões, não era confiável, era difícil de manter e seu desempenho era insatisfatório. O desenvolvimento de software estava em crise. Os custos de hardware estavam caindo, enquanto os custos de software aumentavam rapidamente. Novas técnicas e métodos eram necessários para controlar a complexidade inerente aos grandes sistemas de *software*. (Sommerville, 2007 p. 04)



Muitas das técnicas estão ligadas a trabalhos em equipe, de maneira a otimizar a construção de algoritmos e soluções integradas para a resolução de um problema maior. O objetivo é desenvolver metodologias de criação para que todos os desenvolvedores façam as modificações e facilitem o entendimento da equipe. Portanto, a Engenharia de *Software* busca, principalmente, pela Qualidade de *Software*, que não somente está relacionado a uma ótima experiência de usuário, mas também relacionada a construção do *software* e sua estrutura e facilidade de modificação. Esta motivação está relacionada principalmente pelo *software* poder sofrer mudanças durante seu processo de construção.

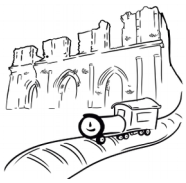
Um dos recursos de Engenharia de Software utilizados na Disciplina de *Software* Livre e Metodologias Participativas é a utilização de Métodos Ágeis, ou seja, um conjunto de ferramentas que facilita o desenvolvimento incremental do *Software*, para que os objetivos da Engenharia de *Software* sejam mantidos. A utilização dos Métodos Ágeis está ligada principalmente à construção do *Software* e não se preocupa em desenvolver documentações muito extensas. Em geral, o programa é dividido em pequenos blocos funcionais, que vão se adaptando conforme as demandas dos usuários.

As ferramentas de Software utilizadas na disciplina de *Software* Livre são *Scrum*, com o objetivo de organização e gerenciamento de projetos, e *Extreme Programming* (XP), que, em conjunto, tornam o desenvolvimento do software mais prático, aliado a técnicas de Programação em Par, em que uma dupla possa focar mais na programação, e Desenvolvimento Orientado a Testes, para testar de maneira automatizada todas as possibilidades que o código deve atender (KNIBERG, 2007). O *Scrum* trabalha com alguns conceitos como o trabalho em pequenas equipes autogerenciáveis (chamados de Times, com 7 a 10 membros), coordenadores por um *Scrum Master*, responsável por ajudar a equipe em suas dificuldades e com um desenvolvimento em ciclos curtos de 2 a 4 semanas conhecidos como *Sprint*.

SOFTWARE LIVRE

O *Software* Livre tem como sua base o compartilhamento do conhecimento tecnológico e, hoje, é defendido por uma comunidade muito grande de pesquisadores acadêmicos, cientistas, ha*ckers* e demais defensores do movimento. O *Software* Livre vai contra uma tendência da economia capitalista, que tem agregado valor na utilização de *softwares* dos quais há uma alta dependência de utilização na sociedade, a partir do modelo de Licenças de Uso de *Software*. Para que um *Software* seja considerado livre, ele deve possuir quatro tipos de liberdade (SILVEIRA, 2004):

1. Uso – É a liberdade de utilizar o software para qualquer propósito definido pelo usuário;



2. Modificações – Fazer modificações do software para seu determinado interesse e ter seu código-fonte aberto;
3. Cópias e Redistribuições – Permitir que a modificação do software feita pelo usuário no código-fonte seja replicada para qualquer pessoa que possua o mesmo interesse;
4. Aperfeiçoamento – Permitir a otimização e aperfeiçoamento feito por terceiros no software.

> Richard Stallman, presidente da *Free Software Foundation* (Fundação do *Software* Livre), costuma comparar o software a uma receita de bolo. Ambos são um conjunto de instruções. Um *software* diz ao computador o que este deve fazer. Uma receita diz à pessoa as quantidades de cada ingrediente, a ordem em que devem ser misturados e outras orientações. Imagine se as pessoas fossem impedidas de trocar receitas? Ou se fossem proibidas de melhorar a receita que conseguiram de sua mãe ou de seu vizinho? (Silveira, 2004, p. 09).

Dada as quatro características fundamentais, o *Software* Livre possui grande relevância a comunidade: Assim como livros e qualquer outra forma de aquisição de conhecimento, a difusão de um software livre permite um maior desenvolvimento de novas ferramentas a partir da liberdade dada pelo compartilhamento do *software*. O uso de um *Software* Livre é permitido para todo e qualquer fim e esta é a característica principal.
Existem diversas comunidades de Software Livre espalhadas pelo mundo, compartilhando projetos na internet e trabalhando em conjunto para aprimorar e criar softwares. Com milhares de participantes, as comunidades estão se tornando cada vez mais sólidas. A mais conhecida é a *GNU/Linux*, que desenvolve distribuições de sistemas operacionais. Geralmente, os usuários participantes das comunidades utilizam seu tempo livre para contribuir nos meios de comunicação, ou até mesmo disseminando sobre o uso do *software* em questão.

> […] Dividem-se aqui, mais uma vez, duas esferas de trabalhos que podem estar voltados a uma determinada distribuição: primeiro, aquelas atividades que estão ligadas diretamente ao software, tais como desenvolvimento, programação, tradução, documentação etc.; segundo, todo e qualquer trabalho indireto que faça alusão à distribuição, sendo os mais comuns a divulgação, o incentivo ao uso, a promoção de eventos, palestras e seminários ou a postagem de conteúdos em fóruns e listas de discussões, tirando dúvidas, trocando informações, direcionando os principiantes. (Machado, 2009. p.34).

No Brasil, as comunidades possuem alta representatividade, com participações em fóruns, listas de discussão, e-mails e redes sociais, principalmente. Segundo Machado (2009), um exemplo é o *software* Fedora que possui uma das comunidades com maior representatividade na América Latina e no Brasil, com embaixadores em diversos estados brasileiros, além de ser muito

ativa. Quando novas versões do Fedora são lançadas, as traduções para "Português do Brasil" são lançadas rapidamente[5]. O objetivo da Disciplina ao utilizar o conceito de *Software* Livre é analisar os tipos de metodologias participativas no desenvolvimento de software, além de trazer debates sobre o uso e criação de softwares livres como forma de desenvolvimento coletivo e sobre suas Comunidades. Criando, a partir disso, projetos utilizando metodologias participativas.

DESIGN PARTICIPATIVO

O *Design* Participativo (*Participatory Design* ou PD) é uma metologia de desenvolvimento de software que busca aproximar todas as partes envolvidas, incluindo os clientes, no processo de desenvolvimento. No *Design* Participativo, os demandantes do software são convidados a cooperar com os desenvolvedores, idealmente participando de todas as etapas da produção do sistema, desde o levantamento dos requisitos até a validação do produto final. Dessa forma busca-se que o *software* atenda a todas as demandas dos clientes.

O objetivo da disciplina ao adotar essa metodologia é estimular o estudante a ir a campo e trabalhar junto com um demandante real para projetar um software que atenda às necessidades do problema levantado por este. Os alunos, dessa forma, entram em contato com a realidade do demandante, saindo do ambiente familiar da sala de aula e tendo de buscar ver o problema em questão sob a ótica do outro, que em geral é alguém com pouco conhecimentos técnicos de programação. Alvear (2014), em sua tese de doutorado, explica as vantagens do *Design* Participativo:

> No caso de sistemas de informação para movimentos sociais, estamos falando de um público que costuma ter pouca experiência com sistemas de computador. Nesse sentido, elementos de PD como o uso de cenários, jogos, colagens e design em papel podem ajudar a quebrar um pouco o medo e a visão de que sistemas de computador são elementos muito distantes da vida deles.
>
> Além disso, a premissa de que não existe uma "realidade" objetiva a ser levantada e modelada é o ponto de partida fundamental para construir, junto com esse público, um sistema de informação que se ajuste às suas necessidades. A entrevista formal e estruturada, técnica mais comum do desenvolvimento tradicional para levantar os requisitos de um sistema, está longe de ser suficiente para esse público. É necessário conhecer seu ambiente, vivenciar sua realidade, entender seus problemas, para, depois, utilizar-se de dinâmicas, principalmente coletivas, para desenvolver esses sistemas.

5 Na verdade, essas comunidades não são assim tão globais como se dizem e muitas vezes fica relegado ao desenvolvedores do terceiro mundo atividades consideradas menores, como desenvolvimento de plugins, tradução, e divulgação (PRIMO, 2017). Essa dinânica é muito similar com a reflexão sobre a separação entre produção e reprodução dos estudos de gênero e tecnologia, que também ocorrem nas comunidades de software livre (BUSTOS, 2010).

> O uso de protótipos e do design by doing (ou design in use) é muito importante para permitir que esses requisitos surjam de forma mais dinâmica. Muitas necessidades e demandas desses usuários surgirão ao longo do uso, ao perceber as possibilidades e os limites da tecnologia. É provável que, a partir do uso desses sistemas, essas pessoas comecem a usar mais computadores e internet, o que lhes fará ter mais clareza sobre o que eles querem ou gostariam. (ALVEAR, 2014, p. 136)

Dessa forma, mais do que os métodos e ferramentas do PD, prioriza-se que os alunos atuem a partir dos princípios do PD: (1) As ferramentas são para facilitar o trabalho das pessoas, e não substituí-las; (2) os trabalhadores sabem o que é melhor para si; (3) a percepção e o sentimento dos usuários sobre as tecnologias são tão importante ou mais que a tecnologia em si; e (4) os sistemas computacionais são parte de um contexto de um ambiente de trabalho ou da forma das pessoas se relacionarem (SCHULER & NAMIOKA, 1993, p. 11).

ESTUDOS EM CIÊNCIA, TECNOLOGIA E SOCIEDADE (CTS)

Os estudos em Ciência, Tecnologia e Sociedade (CTS) são um ramo do estudo das ciências que tratam de como valores sociais, político e culturais configuram a construção de fatos científicos e artefatos tecnológicos, e como, de maneira análoga, a produção científica e a inovação tecnológica afetam a sociedade, definindo novas relações sociais e condições de vida.
Ao adentrar nos estudos CTS durante a disciplina, permite-se ao estudante o debate de tópicos como o questionamento da neutralidade científica, o fetichismo da tecnologia, a linearidade do progresso e a problematizar o modelo de difusão tecnológica. Essa experiência em sala de aula portanto contrasta com a visão hegemônica dentro do ensino da engenharia que Ivan da Costa Marques expõe em seu artigo “Engenharias brasileiras e a recepção de fatos e artefatos”:

> O mito da universalidade e da neutralidade da ciência pura é transferido em parte para a engenharia no momento em que a formação do engenheiro o induz a acreditar que haja e que ele possa prover uma solução puramente técnica para a construção de um artefato (bem ou serviço) que lhe seja solicitada. Ensina-se aos estudantes de engenharia, explicita ou implicitamente, que ao profissional cabe cuidar da parte “técnica” do artefato tecnológico. Estabelece-se uma divisão entre o “técnico” e o “social” ou “político”, e cabe ao engenheiro tratar aquela parte que se pretende independente das condições sociais locais e que por isto como que paira acima ou pelo menos separada delas. No entanto, de modo geral, qualquer projeto de engenharia envolve tomar decisões. E qualquer decisão, qualquer escolha no projeto de um artefato, privilegia uns e desfavorece outros. Não se pode escapar disto. Não há, pelo menos não há mais, universalidade e neutralidade (MARQUES, 2005, p. 3)

Nesse contexto apresenta-se também o conceito da Tecnologia Social, que trata da reflexão do desenvolvimento de tecnologias

voltadas principalmente para o desenvolvimento social, em contraposição as tecnologias convencionais. Segundo Dagnino (2008, p. 54-55), “A Ciência e Tecnologia gerada sob a égide de determinada sociedade e, portanto, construída de modo a ela funcional, está de tal maneira ‘‘comprometida’’ com a manutenção desta sociedade que não é passível de ser utilizada por outra sociedade.” Dessa forma, a Tecnologia Social estaria voltada principalmente para os trabalhadores e não o capital, sendo apropriada para pequenos empreendimentos, organizações informais, empreendimentos de economia solidaria e outros grupos associativos (DAGNINO, BRANDÃO e NOVAES, 2004).

Esse eixo acabou se tornando transversal ao longo das turmas. Atualmente trabalhamos apenas com um texto introdutório e depois vamos trabalhando essa temática de forma transversal e interdisciplinar ao longo dos outros três eixos.

RESULTADOS

Até o momento da escrita deste trabalho, 6 turmas foram finalizadas (2011/1, 2012/1, 2014/2, 2015/1, 2016/2 e 2017/2). É importante ressaltar que a turma de 2016/2 foi a primeira a contar com alunos de mestrado, que a cursaram junto dos da graduação, e que a de 2017/2 foi a primeira a ser oferecida exclusivamente para o mestrado. A Tabela 1 mostra um resumo:

Ano/ Semestre	Nr de Estudantes	Projeto(s)	Cursos Atendidos
2011/1	6	Sistema de Plenária Virtual (http://sisplev.sourceforge.net/)	Eng. de Computação (Graduação)
2012/2	10	Sistema de Fontes Jornalísticas (https://bitbucket.org/celsoale/slmp-fontes)	Eng. de Computação (Graduação)
2014/2	20	Riob.us Relatórios (https://github.com/RioBus/analytics)/ Sistema de Gestão do NIAC (https://github.com/NandoFire/plataforma-niac-slmp)	Eng. de Computação e Ciência da Computação (Graduação)
2015/1	18	Riob.us Relatórios / Sistema de Apoio à Comissão de Direitos Humanos da Alerj	Eng. de Computação e Ciência da Computação (Graduação)
2016/2	10	Nandu – sistema de linha do tempo para apoio didático a professores de história (https://github.com/SLMP/linhaDoTempo)	Eng. de Computação e Ciência da Computação (Graduação) + Mestrado Profissional em Tecnologia para o Desenvolvimento Social
2017/2	8	Sistema de transparência de dados abertos para professores da educação básica	Mestrado Profissional em Tecnologia para o Desenvolvimento Social

Tabela 1- Número de Estudantes e cursos inscritos nas Disciplinas nos semestres oferecidos

Nas quatro primeiras turmas exclusivas para estudantes da graduação que vinham de áreas da computação, foram realizadas entrevistas com movimentos sociais, órgãos públicos e grupos populares que apresentaram suas demandas. Os alunos especificaram, modelaram e desenvolveram sistemas que foram entregues a essas organizações e, em alguns casos, se tornaram projetos de extensão dando prosseguimento com melhorias e suporte a esses sistemas.

Em 2016/2, a disciplina teve a primeira experiência em combinar alunos de graduação com alunos do Mestrado Profissional em Tecnologia para o Desenvolvimento Social. Devido ao menor número de alunos, a turma inteira trabalhou em um único projeto: o desenvolvimento de um software para uso em salas de aula da educação básica por professores de história para a criação de linhas do tempo interativas. O produto final foi

batizado de *Ñandu,* disponível para download no *GitHub* (https://github.com/SLMP/linhaDoTempo).

Em 2017/2, em decorrência de incompatibilidades nos horários e calendários da graduação e da pós-graduação decidiu-se por abrir a turma apenas para alunos do mestrado. Essa turma teve duas particularidades: (i) foi composta majoritariamente (6 alunos dentre os 8) por professores da educação básica; e (ii) apenas um dos alunos tinha experiência prévia com programação. Dessa forma além dos tópicos tradicionalmente ensinados na disciplina, conforme já descrito previamente no presente artigo, também tivemos que introduzir os alunos na programação. Para isso foram ensinadas as linguagens de programação *Logo* e *Python*, essa última sendo usada no projeto final da disciplina, que fora um *software* para *smartphones* que realizava a leitura de bases de dados abertas e produzia gráficos. O público-alvo desse produto seriam professores da educação básica que usariam a ferramenta para propor políticas públicas para as secretarias de educação do estado e municípios.

Para além dos produtos de software produzidos na disciplina, um resultado importante são as avaliações dos professores e da metodologia empregada na disciplina. Abaixo estão as médias dessa avaliação realizada pelos alunos da disciplina[6][7]:

Ano/ Semestre	Apresentaram uma programação clara da disciplina (Objetivos, estratégias, conteúdos, recursos, material bibliográfico, sistema de avaliação).	Estratégias adotadas em aula (Metodologia didática) facilitaram a aprendizagem dos alunos.	Em resumo, considerando inclusive as suas qualidades e fraquezas, avalie os professores de um modo geral.
2012/2	8,8	8,1	8,6
2014/2	8,6	8,5	9,4
2015/1	8,8	8,3	9,2
2016/2	8,8	8,4	8,8
2017/2	8,0	7,8	8,8

Tabela 2: Médias das avaliações quantitativas da disciplina pelos alunos

6 Em 2011/1 não foi realizada essa avaliação, por isso essa turma está ausente da tabela abaixo.
7 Cada índice é resultado da média entre as avaliações de todos os alunos da turma. Para cada item, cada aluno deu uma nota de 0 a 10.

De forma geral, houve uma queda na avaliação quando se juntaram os alunos da graduação com o mestrado, que ficou mais evidente na avaliação qualitativa feita na ultima aula da turma de 2016. O perfil dos alunos da graduação é de alunos com uma formação técnica bem forte (alunos dos ultimos períodos de cursos de computação, muitos com experiência de estágios em empresas) que buscam aprofundar seus conhecimentos para atividades consideradas mais “sociais”, como um maior contato com usuários e demandantes, através de métodos e técnicas variadas de levantamento de requisitos. Já os alunos do mestrado em Tecnologia para o Desenvolvimento Social, vem de formação de cursos de Humanas, e buscam entender como podem usar Tecnologias da Informação para contribuir em suas pesquisas e trabalhos.

Dessa forma, concluímos pela necessidade de separação em duas disciplinas, deixando a SLMP para a graduação, da forma como era dada até 2015. Por outro lado, para alunos do mestrado profissional, criaremos uma disciplina com perfil um pouco mais técnico, sobre introdução a programação e ferramentas de software. Dessa forma, o objetivo será instrumentalizá-los para poderem criar apps e sites simples, para trabalhar e analisar com banco de dados públicos e contribuir com analises quantitativas em suas pesquisas.

CONSIDERAÇÕES FINAIS

Neste artigo, descrevemos a disciplina *Software* Livre e Metodologias Participativas, oferecida pelo Departamento de Engenharia Eletrônica da UFRJ. Destacamos seu caráter questionador do ensino tradicional de Engenharia, e sua vertente de ensino voltado à extensão.

A experiência de combinar a disciplina na graduação e no mestrado, como realizado em 2016, provocou a dificuldade de lidar com as especificidades de cada um dos grupos, como um dos alunos disse na sua avaliação final: “Penso que deveria haver uma preocupação maior em considerar a diferença de conhecimento dos alunos sobre o conteúdo da disciplina.” Essa dificuldade se deu devido ao fato dos alunos vindos do mestrado não possuírem em sua maioria experiência com programação, enquanto que os alunos da graduação já possuem, em decorrência de disciplinas eletivas como essa serem cursadas após o término do ciclo básico dos cursos.

Por outro lado, os alunos de mestrado, em virtude de se tratar de um mestrado profissional, já possuem contato com algum campo para criação do projeto final. Em uma turma com um perfil relativamente homogêneo como a que cursou a disciplina em 2017 essa aproximação prévia com o campo auxilia no processo de levantamento de requisitos e na aplicação do *Participatory Design.*

Os desafios a enfrentar em direção a uma verdadeira transformação do currículo da Engenharia são profundos. A criação desta disciplina busca abordar uma pequena parte deste desafio, ao relacionar a engenharia com o desenvolvimento social e questionando os paradigmas da propriedade do conhecimento, da tecnocracia e da neutralidade da ciência. É necessário, no entanto, que este tipo de iniciativa ultrapasse os conteúdos curriculares opcionais e a iniciativa pessoal de professores engajados e se insira nos cursos de forma estrutural. Com este relato, esperamos contribuir para este debate e inspirar novos métodos no ensino de Engenharia.

REFERÊNCIAS

ALVEAR, C, A, S. **Tecnologia e participação: Sistemas de informação e a construção de propostas coletivas para movimentos sociais e processos de desenvolvimento local.** Tese (Doutorado) - Programa de Engenharia de Produção, Universidade Federal do Rio de Janeiro (PEP/COPPE/UFRJ), Rio de Janeiro, 2014.

BUSTOS, Tania Pérez. Reflexiones sobre una etnografía feminista del Software Libre en Colombia. Rev. Estud. Fem., Florianópolis , v. 18, n. 2, p. 385-405, Aug. 2010 . Available from <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0104-026X2010000200006&lng=en&nrm=iso>. access on 20 Feb. 2018. http://dx.doi.org/10.1590/S0104-026X2010000200006.

DAGNINO, R. **Neutralidade da ciência e determinismo tecnológico: Um debate sobre a tecnociência**. Campinas: Editora Unicamp, 2008.

DAGNINO, R.; BRANDÃO, F.C.; NOVAES, H.T. Sobre o marco analítico conceitual da tecnologia social. Em: LASSANCE Jr. et al. **Tecnologia Social – uma estratégia para o desenvolvimento**. Rio de Janeiro: Fundação Banco do Brasil, 2004.

FREIRE, P. (2005). Pedagogia do oprimido. Rio de Janeiro: Paz e Terra.

GREENSTEIN, S., NAGLE, F. Digital dark matter and the economic contribution of Apache. **Research Policy**, v. 43, n. 4, p. 623-631, 2014.

HIGHSMITH, J. **Manifesto for agile software development**. Disponível em http://agilemanifesto.org, Acessado em 20 de dezembro de 2013. 2001.

KNIBERG, H. **Scrum e XP direto das Trincheiras: Como nós fazemos Scrum.** InfoHQ. 2007.

MACHADO, M. B. Distros e comunidades a dinâmica interna de Debian, Fedora, Slackware e Ubuntu. Em: AGUIAR, V. M., MACHADO, M. **Software livre, cultura hacker e o ecossistema da colaboração.** 1ª edição. São Paulo: Momento Editorial, 2009.

MARQUES, I. C. Engenharias brasileiras e a recepção de fatos e artefatos. Em: LIANZA, S. e ADDOR, F. (Ed.). **Tecnologia e**

desenvolvimento social e solidário. Porto Alegre: Editora da UFRGS, 2005.
PRESSMAN, R. S. **Engenharia de software**. 8a. edição. McGraw Hill Brasil, 2016.
PRIMO, R. S. **O discurso do global nas comunidades de software livre: estudo de caso do WordPress**. Dissertação (Mestrado) - Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro (PESC/COPPE/UFRJ), Rio de Janeiro, 2014. Disponível em: *https://rodrigo.utopia.org.br/files/Dissertação.pdf*
SCHWABER, K., SUTHERLAND, J. Guia do Scrum – Um guia definitivo para o Scrum: As regras do jogo. 2013. Disponível em https://www.scrum.org/Scrum-Guide. Acessado em 20 de dezembro de 2013.
SCHULLER, D. NAMIOKA. A. **Participatory Design: Principles and Practices**. Hillsdale NJ: Lawrence Erlbaum Associates. 1993.
SILVEIRA, S, A. **Software livre: a luta pela liberdade do conhecimento** – 1a edição. São Paulo: Editora Fundação Perseu Abramo, 2004.
SOMMERVILLE, I. **Engenharia de Software.** 8a ediçao. São Paulo: Pearson Addison-Wesley Editora, 2007.
SOMMERVILLE, I. Software Engineering. 10th edition. Pearson Education Limited, 2016.
STALLMAN, R. M. **Free Software, Free Society: Selected essays of Richard M. Stallman**. Boston, MA: GNU Press, 2002.
THIOLLENT, M. **Metodologia de Pesquisa Ação**. 7a edição (1985 – 1ª edição). São Paulo: Cortez Editora, 1996.