Software Livre e Metodologias Participativas - ensino e extensão em uma disciplina da Engenharia

Área Temática: Formação em Engenharia e Novas Possibilidades

Lucas Carvalho[1], Fernando Rodrigues[2], Rodrigo Ferreira[3], Pedro Braga[4], Alan Tygel[5], Celso Alvear[6], Rodrigo Primo[7]

[1] *Universidade Federal do Rio de Janeiro – UFRJ – Rio de Janeiro – RJ – llagocarvalho@gmail.com*
[2] *Universidade Federal do Rio de Janeiro – UFRJ – Rio de Janeiro – RJ – fernandorodrigues@poli.ufrj.br*
[3] *Universidade Federal do Rio de Janeiro – UFRJ – Rio de Janeiro – RJ – rodrigo.s.f@poli.ufrj.br*
[4] *Universidade Federal do Rio de Janeiro – UFRJ – Rio de Janeiro – RJ – pedrohcb.ufrj@gmail.com*
[5] *Universidade Federal do Rio de Janeiro – UFRJ – Rio de Janeiro – RJ – alantygel@gmail.com*
[6] *Universidade Federal do Rio de Janeiro – UFRJ – Rio de Janeiro – RJ – celsoale@gmail.com*
[7] *Universidade Federal do Rio de Janeiro – UFRJ – Rio de Janeiro – RJ – rodrigosprimo@gmail.com*

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 atualmente é optativa para alunos de cursos de Engenharia Eletrônica, de Computação e do curso de Ciência da Computação. Ao longo desses anos, foram três turmas já finalizadas e a quarta turma está em andamento. 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

1 Introdução

Os cursos de computação são conhecidos por terem uma formação muito técnica, 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 1970/1980, 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, 2007; PRESSMAN, 2011). 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 um 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).

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.

2 Metodologia de Ensino-Aprendizagem

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 de 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 é desenvolvida em 30 aulas, sendo duas aulas por semana de duração de 2 horas cada[1]. 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. Podemos dividir a disciplina em 3 tempos pedagógicos, que serão descritos a seguir:

2.1 Tempo-Teoria

Na primeira metade da disciplina, que chamamos tempo-teoria, as aulas são orientadas por debates acerca de texto 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.

2.2 Tempo-Prática

A partir da segunda metade da disciplina, as aulas ocorrem em um laboratório de informática e são aulas práticas. Os grupos usam o horário para se 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

[1] O programa da disciplina, contendo a previsão de conteúdo das aulas e a bibliografia completa pode ser acessado em http://bit.ly/1fmLyhy.

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 5 a 10 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.

2.3 Tempo-Avaliação

Ao se tentar romper com os paradigmas tradicionais e bancários da educação, 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.

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

3 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 software 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.

3.1 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.4)

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*.

3.2 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, Hackers 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. 9).

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 novos 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 Fedora são lançadas, as traduções para "Português do Brasil" são lançadas rapidamente.

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.

3.3 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 da produção 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.
>
> 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: (i) As ferramentas são para facilitar o trabalho das pessoas, e não substituí-las; (ii) os trabalhadores sabem o que é melhor para si; (iii) a percepção e o sentimento dos usuários sobre as tecnologias são tão importante ou mais que a tecnologia em si; e (iv) os sistemas computacionais são parte de um contexto de um ambiente de trabalho ou da forma das pessoas se relacionarem (SCHULER; NAMIOKA, 1993, p. xi).

3.4 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; NOVAES, 2004).

4 Dois casos práticos

A seguir, descrevemos dois projetos trabalhados durante a edição de 2014/2 e 2015/1 da disciplina.

4.1 Riob.us/Relatórios

A ideia inicial surgiu durante uma discussão em sala na qual se sugeriu a criação de um sistema que pudesse ser usado pela população para controlar a qualidade do serviço de ônibus. A partir daí os alunos começaram a desenvolver um sistema com licença de Software Livre de relatórios para gerar dados que servissem de base para pressionar os gestores públicos por melhorias nesse serviço, assim como para realizar análises sobre a mobilidade urbana na cidade do Rio de Janeiro. Esse sistema foi pensado a partir do Rio Bus (http://riob.us), um Software Livre criado por dois alunos do curso de Ciência da Computação da UFRJ.

O Rio Bus é um serviço de monitoramento dos ônibus do Rio de Janeiro que utiliza como base de informação dados disponibilizados publicamente pela prefeitura, contendo a posição em tempo real (atualizada a cada minuto) dos ônibus da cidade. Contando com cerca de 20 mil acessos por mês, sua principal aplicação é estimar o tempo de espera pelo próximo ônibus da linha desejada pelo usuário. Ele é escrito na linguagem de programação JavaScript, linguagem também adotada nos códigos do sistema de relatórios.

Como a maioria dos integrantes do grupo não estavam familiarizados com JavaScript, os alunos organizaram alguns *Coding Dojos* (quando um grupo se reúne para treinar técnicas e metodologias de desenvolvimento de software através da solução de um pequeno desafio), entretanto a forma mais eficaz de aprendizado foi a troca de conhecimento entre os mais e menos experientes, que se concretizou tanto dentro quanto fora da sala de aula. Os estudantes também precisaram aprender a utilizar algumas ferramentas amplamente empregadas no desenvolvimento de software, como o Git (sistema de controle de versão distribuído e de gerenciamento de código fonte) e o Trello (ferramenta colaborativa para gerenciamneto de projetos). A comunicação entre os integrantes se desenvolveu por meio do Facebook, por um grupo de e-mail e pelo próprio Trello, ao passo que a comunicação com os demandantes se realizou apenas por email e Facebook.

O primeiro demandante do projeto foi o gabinete do deputado estadual Eliomar Coelho, responsável pela CPI dos ônibus do Rio de Janeiro. O diálogo surgiu da necessidade por dados concretos que pudessem embasar os questionamentos feitos pela comissão da CPI. Já na turma seguinte os encontros com o assessor se mantiveram, porém os alunos buscaram outros demandantes com o intuito de obter diferentes pontos de vista. Os novos demandantes foram o Fórum de Mobilidade Urbana, grupo que se dispõe a traçar diagnósticos e apontar soluções para os problemas relacionados à área, e um professor do Programa de Engenharia de Transporte da COPPE/UFRJ (PET) que atualmente tem sua pesquisa voltada para criação de indicadores de mobilidade urbana no Rio de Janeiro.

Como resultado, no semestre de 2014/2 foram desenvolvidos os seguintes relatórios: linhas sem ônibus circulando, linhas por quantidade de ônibus, ônibus por faixa de velocidade, ônibus com GPS desatualizado e ônibus sem linha. Enquanto que no semestre de 2015/1 estão sendo implementadas as demais funcionalidades:

distância média entre os ônibus de uma linha, quilômetros percorridos por um ônibus, quantidade de ônibus em funcionamento por horário e tempo médio de espera por um ônibus em um ponto. Outro objetivo da segunda turma é de melhorar a interface, além de resolver alguns problemas de acesso ao sistema.

Os alunos se apropriaram do projeto, apesar da curta participação de apenas um período letivo, e o que se percebe é que grande parte do envolvimento dos integrantes do grupo se justifica pelo retorno que o projeto dá à população e pela experiência prática de desenvolver um sistema real.

4.2 Núcleo Interdisciplinar de Ações para a Cidadania (NIAC/UFRJ)

Criado em 2006, o Núcleo Interdisciplinar de Ações para Cidadania da UFRJ (NIAC) é um programa de extensão que articula as áreas de Direito, Psicologia e Serviço Social, com o objetivo de desenvolver diferentes ações em Direitos Humanos e acesso à Justiça. Um dos objetivos principais do NIAC, entre muitos, é garantir defesa dos direitos fundamentais, além da promoção, difusão e educação em direitos humanos às pessoas vulneráveis às dinâmicas da violência. A maior parte das pessoas atendidas pelo mesmo, tem sua moradia no Complexo da Maré[2], embora o NIAC esteja aberto para qualquer pessoa que o procure.

Esse mesmo núcleo de extensão contactou o professor da disciplina para ajudar na resolução de algumas de suas dificuldades. Com a grande quantidade de casos, fica difícil armazená-los para depois consultá-los, existindo um arquivo físico onde os casos ficam armazenados. Assim, estes só podem ser consultados por uma pessoa de cada vez, além de imprevistos acontecem, como o sumiço temporário de alguns casos impossibilitando sua consulta até reencontrarem. Por fim, fica difícil fornecer mensalmente as estatísticas de casos atendidos (bairros onde as pessoas atendidas moram, renda mensal etc).

Porém, um dos grandes problemas para a implantação de um sistema de apoio estava na resistência de alguns membros da equipe do NIAC a uma digitalização dos casos, pois alguns preferiam continuar guardando-os em papel, mesmo que para isso ainda continuassem a sofrer os problemas acima citados. Mas, depois de mobilizações internas e também do apoio da disciplina de Software Livre e Metodologias Participativas (SLMP), eles começaram a se abrir para essa ideia.

Assim, na turma de 2014/2 foi iniciado o desenvolvimento de um sistema web que apoiasse no cadastro, busca e acompanhamento dos casos, com a participação da equipe do NIAC. Uma das preocupações foi garantir quesitos de segurança de informação, pois existem casos com informações muito sensíveis que não podem ser vistos por ninguém fora do NIAC, e em alguns casos informações que só podem ser vistas pela área que fez o atendimento. Esse sistema foi entregue em fevereiro de 2015, e por ainda não atender a todos os requisitos que eles desejavam, decidiu-se iniciar um projeto de extensão com a cessão de uma bolsa por parte do NIAC para o Núcleo de Solidariedade Técnica da UFRJ (SOLTEC/UFRJ), como forma de continuar o desenvolvimento do sistema.

---

[2] O Complexo da Maré é um conjunto de favelas que é vizinho ao Campus principal (Ilha do Fundão) da UFRJ, com população em torno de 130.000 habitantes, segundo o Censo Demográfico do IBGE de 2010.

No incio de 2015, decidiu-se também que o NIAC não seria um possível caso da disciplina nesse ano, já que em função de algumas resistências internas ao sistema, seria melhor avançar no sistema entregue através da ação do bolsista de extensão, para só depois de implementada uma versão com sucesso, repensar a a entrada de uma nova turma no NIAC. Assim, neste ano o foco será implementar um sistema que atenda as necessidades mínimas do NIAC, capacitar seus membros para uso e acompanhar a apropriação do sistema por parte deles.

5 Resultados

Até o momento da escrita deste trabalho, 3 turmas foram finalizadas (2011/1, 2012/1 e 2014/2) e uma esta em andamento (2015/1). A Tabela 1 mostra um resumo:

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

Ano/Semestre	Número de Estudantes	Projeto(s)	Cursos Atendidos
2011/1	6	Sistema de Plenária Virtual	Eng. Eletrônica e de Computação
2012/2	10	Sistema de Fontes Jornalísticas	Eng. Eletrônica e de Computação, Eng. de Computação e Informação
2014/2	20	Riob.us Relatórios / Sistema de Gestão do NIAC	Eng. Eletrônica e de Computação, de Computação e Informação, Controle e Automação e Ciência da Computação
2015/1	18	Riob.us / Sistema de Apoio à Comissão de Direitos Humanos da Alerj	Eng. Eletrônica e de Computação, de Computação e Informação, Controle e Automação e Ciência da Computação

Na primeira turma, o projeto realizado foi de um sistema de Plenárias Virtuais (http://sisplev.sourceforge.net/). Foram realizadas entrevistas com movimentos sociais, que colocaram a necessidade de sistemas que facilitem a organização popular através de plenárias online, em que participantes não precisem se deslocar. Desta forma, economiza-se tempo e dinheiro. O projeto foi especificado, mas não chegou a ser implementado. Tentou-se adaptar um software já existente (Big Blue Button), contudo o tempo para entender o código dele não foi suficiente.

A segunda turma recebeu a demanda da Superintendencia Geral de Comunicação Social da UFRJ (SGCOMS), e ao final foi implementado o sistema de fontes jornalisticas que atualmente está em uso pelo órgão (https://bitbucket.org/celsoale/slmp-fontes).

Em 2014/2, devido ao maior número de alunos, trabalhou-se em dois times. Um deles desenvolveu o sistema Riob.us/relatórios (https://github.com/RioBus/analytics) e o outro o sistema de gestão do NIAC (https://github.com/NandoFire/plataforma-niac-slmp), descritos na seção anterior. Ambos os sistemas se transformaram em projeto de extensão do Núcleo de Solidariedade Técnica (SOLTEC/UFRJ), que dispõe de um bolsista para cada projeto responsável por seguir melhorando, dando manutençao e dialogando com os demandantes.

Por fim, na turma de 2015/1, ainda em andamento, um dos times está continuando o desenvolvimento do sistema Riob.us/Relatórios e o outro está desenvolvendo um sistema apoio a Comissão de Defesa dos Direitos Humanos e Cidadania da Assembléia Legislativa do Estado do Rio de Janeiro (CDDHC/ALERJ).

6 Conclusões

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.

Os desafios a enfrentar em direção à 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.

7 Referências Bibliográficas

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.

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.

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**. 7a. edição. McGraw Hill Brasil, 2011.

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.

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.