Reforçando a Comunicação com Uso de uma Ferramenta de Software Livre em Ensino de Engenharia de Software

Débora M. B. Paiva , Alexandre M. Manduca , Marco Aurélio G. Silva , Leonardo J. Quemello , Rosana T. V. Braga , Renata P. M. Fortes *

[1]Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo
Caixa Postal 668, São Carlos, SP

debora,magsilva,rtvb,renata@icmc.usp.br

michetti,leonardo@grad.icmc.usp.br

Abstract. *The gap that exists between software industry requirements and academic background has motivated the introduction of new experiences in Computer Science courses. In this paper, we present the planning, operation and results obtained from the application of a practical exercise developed by Software Engineering students. The main goal was to propose an activity in which communication among group members could be perceived as an essential element to accomplish the desired results and to fulfill the activity completely. It was noticed that students recognized the importance of communication in the proposed activity context and that they concentrated their efforts in making it effective.*

Resumo. *A lacuna existente entre as exigências da indústria de software e a formação acadêmica tem motivado a realização de experiências inovadoras em cursos de Ciência da Computação. Neste artigo, apresenta-se o planejamento, a execução e os resultados obtidos com a realização de um trabalho prático desenvolvido por estudantes de uma disciplina de Engenharia de Software. O principal objetivo foi propor a realização de uma atividade em que a comunicação entre os membros dos grupos pudesse ser percebida como um elemento fundamental para alcançar os resultados desejados e cumprir a atividade plenamente. Foi observado que os estudantes reconheceram a importância da comunicação no contexto da atividade proposta e se esforçaram para torná-la efetiva.*

1. Introdução

Atualmente muitas organizações estão empenhadas em melhorar a qualidade de seus processos e de seus produtos de software. Pesquisas realizadas pelo SEI (*Software Engineering Institute*) sobre o número de organizações que buscam certificação SW-CMM (*Capability Maturity Model for Software*) [**?**] e CMMI (*Capability Maturity Model Integration*) [**?**] têm demonstrado que o interesse da comunidade de software em relação à qualidade tem aumentado de forma significativa desde 1987, quando as pesquisas iniciaram. Com isso, torna-se crescente também a busca por profissionais qualificados, conscientes da importância em aplicar boas práticas de Engenharia de Software no desenvolvimento dos projetos.

*Os autores agradecem à CAPES e FINEP (MCT/FINEP/CT-INFO 01/2003) pelo suporte financeiro

No entanto, frequentemente são relatadas na literatura diferenças entre a formação acadêmica dos estudantes e as exigências do mercado de trabalho em relação ao desenvolvimento de software. É muito comum que gerentes de projetos desenvolvidos em indústrias expressem as dificuldades dos profissionais em trabalhar como membros de grupos e de gerenciar seus trabalhos individuais de forma produtiva [**?**]. De acordo com McCracken [**?**], estas e outras dificuldades têm acarretado sérios problemas no desenvolvimento de software em contexto industrial mas eles acreditam que é completamente viável que tais dificuldades comecem a ser tratadas em ambiente acadêmico.

Algumas propostas têm sido implementadas em cursos de graduação e de pós-graduação com o objetivo de tentar implantar um ambiente de ensino que favoreça o crescimento da maturidade dos estudantes para Engenharia de Software [**?**]. Em geral, essas propostas enfatizam não somente o aprendizado de aspectos técnicos da disciplina mas, principalmente, de aspectos sociais do desenvolvimento de um projeto de software, tais como cooperação e comunicação entre desenvolvedores. Além disso, é fundamental que os estudantes estejam preparados para assumir posição crítica frente a novas metodologias e tecnologias e saibam identificar como agregar valor aos projetos em desenvolvimento utilizando-se dos recursos disponíveis.

Em desenvolvimento de software livre, é comum que os indivíduos trabalhem em grupo e de forma geograficamente dispersa. Esses grupos utilizam ferramentas constituídas por um repositório de dados que pode ser acessado, via rede, por todos os desenvolvedores e por usuários finais. Durante o desenvolvimento de projetos desse tipo, elementos como comunicação e colaboração são fortemente explorados. Portanto, é comum que as ferramentas utilizadas pela comunidade de software livre ofereçam mecanismos que favoreçam os aspectos sociais do desenvolvimento de software. Por exemplo, o CVS (*Concurrent Versions System*) [**?**], utilizado para controle de versões de artefatos, permite que o desenvolvedor associe a cada versão um *log* descrevendo (ou comunicando) alterações realizadas e informações gerais sobre aquela versão. A ferramenta Bugzilla [**?**], utilizada para controle e gerência de alterações de software, possui o recurso de troca de mensagens entre os desenvolvedores, facilitando a discussão sobre a resolução de *bugs* registrados.

No contexto da disciplina de Engenharia de Software, oferecida aos estudantes de Ciência da Computação do ICMC-USP, foi proposto um trabalho prático envolvendo o uso da ferramenta Bugzilla (descrita na Seção **??**). Os objetivos deste trabalho foram:

1. oferecer ao estudante a oportunidade de participar de um projeto desenvolvido de forma distribuída, em que a comunicação entre os membros da equipe pudesse ser valorizada;
2. aprimorar a formação dos estudantes em relação ao cumprimento da atividade de controle de alterações;
3. apresentar ao estudante um cenário de uso da ferramenta Bugzilla, de forma que ele pudesse interagir neste cenário, utilizar a ferramenta, fazer críticas em relação ao seu uso e analisar os potenciais benefícios em utilizá-la em outras atividades de Engenharia de Software.

Como atividade a ser cumprida, foi solicitado que os estudantes tratassem determinadas alterações em código fonte desenvolvido por estudantes de outra disciplina, utilizando a ferramenta Bugzilla como apoio e simulando um ambiente de desenvolvimento distribuído de software.

Com o desenvolvimento deste trabalho, foi interessante notar como os estudantes valorizaram a comunicação entre os membros do grupo (após superarem uma resistência inicial) como um recurso importante de apoio ao desenvolvimento do projeto. Do ponto

de vista de ensino de Engenharia de Software, a experiência realizada foi importante, pois os estudantes puderam vivenciar uma situação prática que permitiu o aprimoramento do aprendizado de conceitos técnicos e, ao mesmo tempo, serviu como uma primeira oportunidade para que eles pudessem reconhecer a importância em desenvolver habilidades essenciais (por exemplo, a comunicação) para o desenvolvimento de projetos de software.

O trabalho proposto aos estudantes está no contexto do projeto SAFE (*Software Engineering Available for Everyone*) [**?**, **?**], que tem como principal objetivo desenvolver uma infra-estrutura que permita a integração de ferramentas de software livre de apoio às atividades de Engenharia de Software. Em particular, os resultados obtidos com a realização deste trabalho prático contribuíram para o projeto no sentido de indicar potenciais usos da ferramenta Bugzilla em relação às diversas práticas de Engenharia de Software.

Na Seção **??** são apresentadas experiências registradas na literatura sobre ensino de Engenharia de Software. Na Seção **??** é apresentada uma visão geral da ferramenta Bugzilla, da disciplina de Engenharia de Software e dos elementos considerados na definição do trabalho prático proposto. Na Seção **??** são apresentados os resultados obtidos com a realização do trabalho e na Seção **??** são apresentadas as conclusões e pesquisas futuras.

2. Ensino de Engenharia de Software em cursos de graduação

As propostas e as experiências relacionadas ao ensino de Engenharia de Software geralmente estão baseadas na premissa de que é fundamental que os estudantes possuam excelente formação técnica e também desenvolvam habilidades para lidar com *pessoas* e com *processos*.

Silva et al [**?**] apresentam os resultados que têm sido obtidos com a mudança de paradigma experimentada no ensino de Engenharia de Software. Segundo eles, esta mudança tem sido benéfica à formação dos estudantes. O paradigma de orientação a objetos e a linguagem UML foram substituídas pela adoção do processo *Extreme Programming* (XP) para guiar o desenvolvimento de trabalhos práticos. Com isso, foi observado que os alunos se concentraram menos à aprendizagem de linguagens de modelagem e programação e se dedicaram mais à aprendizagem e vivência de conceitos como cooperação, reuso e gerência de configuração, dentre outros. Foram constatados também resultados relevantes na aprendizagem e prática dos conceitos de Engenharia de Software, pois os alunos passaram a participar e questionar mais ativamente nas aulas.

Begosso e Filgueiras [**?**] relatam os resultados obtidos com a implantação de um ambiente de maturidade em Engenharia de Software em um curso de graduação em Ciência da Computação. Para a definição do ambiente foi feito um relacionamento entre os objetivos de aprendizagem cognitivos, de habilidade e de atitudes [**?**] com as áreas chave do nível 2 do modelo CMM e as áreas de conhecimento do SWEBOK [**?**]. Em seguida foram relacionadas as disciplinas do curso com os objetivos de aprendizagem, permeando o conteúdo das disciplinas com práticas de maturidade. O ambiente foi experimentado durante dois anos e foi possível observar, segundo os autores, que o processo de implantação do ambiente de maturidade foi bem sucedido, revelando que realmente houve crescimento da maturidade para a Engenharia de Software entre os alunos.

Gnatz et al [**?**] descrevem a abordagem prática utilizada no ensino de Engenharia de Software na Universidade de Munich. O objetivo é oferecer aos estudantes uma experiência de desenvolvimento de um projeto de software em que elementos comuns do contexto industrial são considerados, por exemplo, discussões com o cliente sobre re-

quisitos inconsistentes. São enfatizados também aspectos sociais do desenvolvimento de software, como a comunicação entre membros de um grupo. Os objetivos dos professores são ensinar aos estudantes (1) o significado e a importância de trabalho em grupo e (2) como desenvolver um projeto de software cumprindo todo o ciclo de vida, desde a engenharia de requisitos até a entrega do produto final. Durante o curso é simulado um ambiente de desenvolvimento real de projetos. Assim, os estudantes se reunem frequentemente com o cliente para negociar requisitos, definir prioridades, executar testes de aceitação, etc. São utilizadas diversas ferramentas de software livre, como CVS, JUnit, Eclipse e Bugzilla. Com essa experiência, os estudantes têm percebido a importância e as dificuldades na comunicação entre os desenvolvedores e com os clientes. Além disso, eles geralmente indicam que é muito válida a atividade de desenvolvimento de software considerando-se fatores comuns do ambiente industrial.

O estudo dos trabalhos apresentados nesta seção foi importante principalmente por dois motivos:

- estimulou a realização do trabalho prático apresentado neste artigo, pois os resultados das experiências indicaram que realmente é possível oferecer ao estudante um ambiente ou um contexto que valorize o desenvolvimento de habilidades importantes, a partir da proposta e execução de atividades relativamente simples;
- indicou elementos fundamentais, por exemplo, formas de incentivar a comunicação, que foram bastante úteis durante a realização do trabalho apresentado.

3. Utilização da Ferramenta Bugzilla em um Curso de Graduação

3.1. Visão Geral da Ferramenta Bugzilla

Bugzilla é uma ferramenta de controle e gerência de alterações de software que permite o registro, a organização e a visualização de *bugs*. O controle de alterações permite que grupos de desenvolvedores ou indivíduos efetivamente organizem o tratamento aos problemas e mudanças em seus produtos. Ferramentas como Bugzilla podem ser utilizadas para apoiar a fase de manutenção de software, proporcionando maior visibilidade de informação sobre o produto, comunicação direcionada e registro histórico das mudanças e da evolução da base de código. O objetivo da ferramenta é servir como apoio na coordenação do trabalho, que envolve alterações em software visando a garantia de qualidade, o aumento na produtividade e a redução no tempo dedicado ao trabalho.

Bugzilla é um software livre distribuído sob a licença MPL (*Mozilla Public License*), desenvolvido originalmente na linguagem TCL. Posteriormente foi escrito em Perl, linguagem na qual se mantém até hoje. É um software implementado como aplicação web, na qual os usuários utilizam um navegador (*browser*) para acessar sua interface. Atualmente é utilizado em vários projetos, tanto livres quanto proprietários. Como exemplos de organizações e projetos que já utilizaram a ferramenta ou a utilizam, pode-se citar: NASA, Motorola, Redhat, Conectiva, GNOME, KDE, e IBM.

A ferramenta possui vários conceitos, recursos e utilidades que incluem: *bugs* e suas propriedades, mudanças de estado e ciclo de vida dos *bugs*, contas de usuário, produtos e componentes, administração do sistema, comunicação direcionada sobre alterações, arquivos anexos, grupos, dependências entre *bugs*, interface de consulta e integração com correio eletrônico. Cada software adicionado à Bugzilla é considerado um produto, e cada produto é dividido em componentes. Os componentes são os módulos de desenvolvimento do software.

Todo usuário cadastrado na ferramenta Bugzilla possui certas permissões para acesso e utilização de algumas de suas funcionalidades. As permissões de um novo usuário são selecionadas pelo administrador do sistema. De acordo com as permissões que cada usuário possui, delimitam-se níveis diferentes de usuários, por exemplo: proprietários de componentes e administrador.

Durante o desenvolvimento ou manutenção do software, qualquer sugestão de melhorias, adição de novas funcionalidades ou erros encontrados são relatados na Bugzilla através da criação de um novo *bug*. Todo *bug* possui várias características que devem ser informadas. Preferencialmente, deve-se ter conhecimento das várias condições nas quais o *bug* ocorreu, para preencher corretamente os valores de suas características. Essas características são, por exemplo: produto, componente, resumo, versão, plataforma, prioridade, severidade, votos, comentários, status e resolução.

Na ferramenta Bugzilla são estabelecidos "estados" de ciclo de vida para cada *bug*, que são: *Unconfirmed*, *New*, *Assigned*, *Resolved*, *Verified*, *Closed* e *Reopened*. Em um determinado momento, um *bug* somente pode estar em um estado. O estado inicial de um *bug* é sempre *Unconfirmed* ou *New*. O ciclo de vida de um determinado *bug* é o conjunto de todos os estados pelos quais ele passou.

3.2. Disciplina de Engenharia de Software no ICMC-USP

O principal objetivo da disciplina de Engenharia de Software oferecida nos cursos de graduação do instituto é fornecer uma visão geral do processo de desenvolvimento de software e de técnicas que podem ser utilizadas em cada fase do ciclo de vida do software. O conteúdo programático abrange, além das fases do ciclo de vida, conceitos fundamentais sobre gerenciamento de projetos e garantia de qualidade de software. A disciplina é oferecida em um semestre para estudantes do segundo ano do curso de graduação em Ciência da Computação e possui carga horária de 60 horas.

Como pré-requisito para cursar a disciplina de Engenharia de Software, os estudantes devem ter cursado anteriormente:

- disciplinas do ciclo básico como Matemática Discreta, Geometria Analítica, Cálculo I, II e III e Física I e II;
- disciplinas de Introdução à Ciência da Computação I e II e Estruturas de Dados I e II, nas quais são abordadas as linguagens de programação C e Pascal.

A disciplina de Engenharia de Software foi oferecida no segundo semestre de 2004 a duas turmas (denominadas A e B neste artigo), totalizando 94 alunos. Foram abordadas as fases de engenharia de requisitos, análise, projeto, teste e manutenção de software. A metodologia orientada a objetos foi enfatizada. Outros temas como gerenciamento de projetos (envolvendo principalmente estimativas de prazos e custos) e garantia de qualidade de software (gerenciamento de configurações, Norma ISO 15504 e CMMI) também foram apresentados. Durante todo o curso, os estudantes cumpriram as fases de um ciclo de vida de software tradicional para o desenvolvimento de um projeto de software, utilizando o Processo Unificado [**?**], cujo tema foi distribuição de carga didática em escolas ou universidades, isto é, um sistema para alocação das disciplinas oferecidas para as diversas turmas de alunos aos professores responsáveis, verificando-se critérios como quantidade de horas/aula, horários e área especializada requerida. Inicialmente, em uma tarefa realizada "em campo", os alunos entrevistaram secretários de instituições de ensino envolvidos com a tarefa de alocar professores e disciplinas aos horários disponíveis do curso. Com isso, os estudantes puderam compreender as principais funcionalidades que um sistema que automatize esta tarefa deve conter. Em cada fase do ciclo de vida os projetos foram continuados e os artefatos referentes foram elaborados.

Foram desenvolvidos também trabalhos práticos envolvendo a técnica de pontos por função, o modelo Cocomo [**?**] e métodos de avaliação de usabilidade de software. A ferramenta Versionweb [1] [**?**, **?**, **?**] foi utilizada pelos estudantes para controle de versões dos artefatos produzidos durante o curso.

Nota-se que existe tradicionalmente um certo preconceito dos alunos em relação à disciplina de Engenharia de Software, por possuir muito conteúdo teórico. Assim, buscou-se aumentar o número de trabalhos práticos e o uso de ferramentas, de forma a motivar os estudantes, apresentar situações reais que ocorrem quando o software é desenvolvido em ambiente industrial e melhor prepará-los para o desenvolvimento de software em contexto industrial. Acredita-se que iniciativas como essa sejam fundamentais para tentar diminuir o *gap* existente entre o ambiente universitário e o mercado de trabalho, uma vez que as vivências reais de fato propiciam uma experiência do empirismo inerente à área de Engenharia de Software.

3.3. Projeto desenvolvido utilizando a ferramenta Bugzilla

Para o desenvolvimento do trabalho, os estudantes foram divididos em 21 grupos de quatro membros e 2 grupos de 5 membros. Para promover a criação de um cenário distribuído de desenvolvimento de software (para o qual a ferramenta Bugzilla é especialmente útil), foram incluídos em um mesmo grupo dois estudantes de cada turma, considerando-se a ordem alfabética dos nomes para distribuição. Por exemplo, os dois primeiros estudantes da lista da turma A formaram grupo com os dois primeiros estudantes da lista da turma B, e assim sucessivamente. Com isso, notou-se que em aproximadamente 80% dos grupos, os membros não se conheciam ou tinham pouco contato, o que é bastante comum quando projetos são desenvolvidos de forma distribuída. Eles foram incentivados a desenvolver os projetos de forma colaborativa, dividindo tarefas e registrando dúvidas, comentários e decisões tomadas, sem realizar encontros presenciais. A própria ferramenta Bugzilla foi utilizada para promover a comunicação entre os membros do grupo, através do recurso de troca de mensagens.

A ferramenta Bugzilla foi apresentada aos estudantes no último mês do curso, quando todos os tópicos já haviam sido ministrados. Foi oferecido treinamento que englobou as características, funcionalidades e uso da ferramenta. O treinamento foi ministrado em sala de aula e teve duração de 2 horas para cada turma. Não houve exercícios práticos.

A ferramenta pode ser utilizada de duas formas diferentes em relação ao acesso aos dados pelos usuários: os usuários podem acessar os dados de todos os produtos do repositório ou podem acessar apenas os dados referentes ao produto com o qual estão envolvidos. Neste trabalho, os estudantes tiveram acesso apenas aos dados do seu grupo de trabalho, por ter sido considerada a abordagem mais interessante para o objetivo de ensino.

O trabalho proposto aos estudantes foi a realização de alterações em código fonte gerado por estudantes de outra turma que estavam cursando a disciplina de Introdução à Ciência da Computação (ICC). Os estudantes da turma de ICC implementaram, como um de seus trabalhos práticos, nove funções utilizando a linguagem C, com o objetivo de exercitar comandos de repetição, de condição e de impressão. A cada um dos grupos de Engenharia de Software foram atribuídos códigos fontes desenvolvidos por diferentes grupos da disciplina de ICC.

Poderia ter-se optado por fazer gerenciamento de alterações nos próprios arte-

[1] esta ferramenta foi desenvolvida com o objetivo de fornecer um modo fácil e eficiente de controle de versões de arquivos. É disponibilizada uma interface web por meio da qual os usuários executam os principais comandos do CVS

fatos elaborados pelos alunos em outros projetos da disciplina, que consistiam basicamente de modelos de análise e projeto. No entanto, considerou-se mais desafiador fazer manutenção em código alheio, principalmente para motivar a comunicação entre os membros do grupo.

Foi solicitado aos estudantes de Engenharia de Software que realizassem alterações simples no código fonte disponível. Essas alterações foram:

1. Inserir os protótipos das funções em um arquivo com a extensão .h
2. Agrupar os códigos das funções em, no máximo, quatro arquivos com a extensão .c
3. Incluir a determinação da soma dos números ímpares em uma das funções desenvolvidas em que, originalmente, o objetivo era determinar a soma apenas dos números pares de uma sequência de n números inteiros.

Essas alterações foram definidas com o objetivo de auxiliar a simulação de erros detectados e que deveriam ser tratados. Para a resolução de cada um dos ítens, os estudantes precisaram inserir o código fonte das funções desenvolvidas pelos alunos de ICC na ferramenta Bugzilla, criar um *bug* e utilizar o processo de resolução de *bugs* intrínseco à Bugzilla para efetuar as alterações solicitadas.

Outro fator que motivou a proposta deste trabalho prático aos estudantes foi a possibilidade de que eles cumprissem uma atividade de manutenção de software, após terem cumprido diversas atividades do ciclo de desenvolvimento de software durante a disciplina. Conforme observado por Gnatz et al [**?**], é difícil mostrar aos estudantes a importância da boa documentação e do código bem escrito e documentado. A importância dessas práticas torna-se evidente quando a manutenção precisa ser realizada. No entanto, cursos de Engenharia de Software geralmente focalizam o desenvolvimento de um novo projeto de software, em detrimento de atividades de manutenção. Portanto, neste trabalho, os estudantes tiveram a oportunidade de atuar em um contexto em que puderam observar como a qualidade dos artefatos produzidos e o cumprimento de boas práticas de Engenharia de Software impactam a fase de manutenção.

4. Execução das Atividades Propostas e Resultados Obtidos

Primeiramente foi realizada a configuração da ferramenta Bugzilla (de forma que os estudantes pudessem acessar apenas os dados de seu grupo de trabalho) e a divisão dos estudantes em grupos. Em seguida, foi apresentado o treinamento sobre a ferramenta e os trabalhos práticos puderam ser iniciados.

Pôde ser notada resistência dos estudantes em relação ao estabelecimento inicial da comunicação nos sete primeiros dias. Como a maioria deles não se conhecia (eles possuíam apenas os emails uns dos outros) esperava-se que eles tentassem se comunicar assim que os trabalhos fossem explicados e que o treinamento fosse oferecido, com o objetivo de começar a definir tarefas, pois eles possuíam apenas 21 dias para aprender a usar a ferramenta na prática, dividir tarefas, definir regras para o desenvolvimento de um trabalho realizado de forma distribuída e cumprir as atividades propostas. Uma parcela dos estudantes (16,66%) desistiu de desenvolver o trabalho proposto. Apesar de não terem sido questionados diretamente sobre os motivos pelos quais desistiram, supõe-se que os desafios que surgem quando um projeto é desenvolvido de forma distribuída tenha sido um dos fatores determinantes. Outra suposição é de que a realização de trabalhos práticos com parceiros desconhecidos também era uma novidade, pois até então os trabalhos em grupo sempre tinham seus membros previamente conhecidos.

Quando a ferramenta começou a ser utilizada, os estudantes tiveram dificuldade em cumprir o processo de criação, resolução e fechamento de um *bug* (ciclo de vida de um *bug*). A maioria dos estudantes procurou esclarecer dúvidas com o professor responsável pela disciplina e com o monitor. Nesta fase, foi observado que algumas mensagens foram trocadas entre os membros dos grupos. Para cada projeto, no mínimo quatro mensagens eram referentes ao uso da ferramenta Bugzilla. Nesse caso, a comunicação oral e escrita foi utilizada como ferramenta em uma situação em que a equipe foi exposta a uma tecnologia desconhecida (aproximadamente 90% dos estudantes não conheciam a ferramenta) e dispunha de pouco tempo para aprendê-la.

Para o desenvolvimento dos projetos, quando a comunicação havia se tornado uma atividade mais natural entre os membros, o conteúdo das mensagens era relacionado principalmente, a atividades cumpridas, pendências, dúvidas e decisões tomadas. Em média, foram trocadas 22 mensagens entre os membros de cada grupo. Considerando-se que as atividades de manutenção propostas foram bastante simples, a quantidade e o conteúdo das mensagens indicaram que, de forma geral, os estudantes se esforçaram para vencer a resistência inicial em estabelecer a comunicação com os colegas em prol da resolução do problema apresentado. Na Figura **??** são apresentadas telas da ferramenta Bugzilla em que podem ser visualizados dados referentes a um *bug* e exemplos de mensagens que foram trocadas entre os estudantes. Outros valores que dimensionam os *bugs* gerados pelos estudantes e as mensagens trocadas entre eles são apresentados na Tabela **??**.

Figura 1: Exemplo de telas da ferramenta Bugzilla

Tabela 1: Valores relacionados as atividades realizadas pelos estudantes

Item	Valor
Número de grupos	23
Número de *bugs*	117
Número de mensagens trocadas	506
Número médio de mensagens trocadas por grupo	22
Número médio de mensagens trocadas por *bug*	4,3

Ao final do cumprimento das atividades propostas, os estudantes foram questionados em relação à importância de uma ferramenta como a Bugzilla para o desenvolvimento de software e sua adequação para auxiliar o cumprimento de outras atividades além daquela que foi experimentada. Ao propor essas perguntas, o objetivo foi promover a reflexão crítica dos estudantes, para que eles se posicionassem de forma favorável ou contrária em relação ao uso de determinada tecnologia no contexto de Engenharia de Software como um todo. As questões foram respondidas individualmente. Nas sub-seções seguintes são apresentados resumos das respostas obtidas e algumas discussões.

4.1. Questão 1: Quão importante você considera a utilização da ferramenta Bugzilla em relação aos tópicos de Engenharia de Software apresentados na disciplina?

De forma geral, os estudantes consideraram muito importante a utilização da ferramenta. Eles observaram a importância em utilizar uma ferramenta que favoreça a comunicação e o registro de decisões que são tomadas quando o desenvolvimento ocorre de forma distribuída. Alguns estudantes mencionaram que um aspecto muito positivo da realização do trabalho prático realizado foi a familiarização com um ambiente de desenvolvimento

mais real. Eles gostariam que atividades como essa fossem mais comuns no curso. Apenas 5% deles expressaram que não compreenderam o objetivo da ferramenta e, portanto, não souberam fazer a avaliação.

Foi mencionado que o trabalho se tornou mais organizado quando a ferramenta foi utilizada, em comparação com outros trabalhos acadêmicos que foram desenvolvidos em grupo. Foi observado também aumento de produtividade, pois não foi preciso marcar reuniões para discutir tarefas a serem realizadas. A própria ferramenta auxiliou bastante em relação ao gerenciamento do trabalho. Outra observação dos estudantes foi a possibilidade de melhorar a reutilização de código quando uma ferramenta como a Bugzilla é utilizada, pois melhorias propostas por um desenvolvedor são rapidamente conhecidas por outros desenvolvedores e a documentação do software torna-se mais completa.

Com as respostas obtidas, foi possível observar o interesse de uma parcela dos estudantes (aproximadamente 50% deles) em utilizar a ferramenta em outros trabalhos desenvolvidos em grupo, de forma a experimentar mais suas funcionalidades e aprender mais (exercitando) sobre o gerenciamento de uma equipe com o apoio de uma ferramenta. Este resultado foi considerado bastante interessante, pois indicou a disponibilidade dos estudantes em mudar a forma como estavam acostumados a trabalhar (característica que tem sido bastante valorizada no mercado de trabalho).

4.2. Questão 2: Em quais atividades de Engenharia de Software além daquela que você realizou (modificação de código fonte) você julga que Bugzilla seria útil? Por que?

As respostas obtidas para esta questão enfatizaram atividades relacionadas à qualidade de software. Foram destacadas as melhorias que são obtidas na fase de teste, pois muitas pessoas testam o software ao mesmo tempo e se candidatam para corrigir os *bugs* encontrados. Foi observado por 68% dos estudantes que seria muito interessante que houvesse integração entre a ferramenta Bugzilla e uma ferramenta de controle de versões, de forma que houvesse uma associação entre *bugs* corrigidos e versões originadas.

Aproximadamente 25% dos estudantes registraram também a possibilidade de uso da ferramenta Bugzilla nas atividades de análise de requisitos e projeto de software. As atividades podem ser iniciadas por alguém do grupo e o trabalho é refinado pelos integrantes. Todos podem discutir detalhes do projeto e aceitar ou rejeitar os modelos que são gerados. O cliente pode acompanhar essas discussões e auxiliar no esclarecimento de dúvidas dos projetistas usando a ferramenta Bugzilla.

Como lições aprendidas com a realização deste trabalho prático, destacam-se:

- O treinamento enfocou principalmente os aspectos teóricos da ferramenta, isto é, as características que ela oferece para fazer o gerenciamento das alterações. Não foi enfatizada a parte prática, o que fez com que os alunos tivessem o primeiro contato com a ferramenta somente durante o experimento em si. Sugere-se que o treinamento seja estendido para incluir um exemplo completo de como criar um produto, incluir *bugs* e simular o gerenciamento desse *bug* até o final do seu ciclo de vida. Isso evitaria alguns dos problemas relatados nesta seção.
- A ausência de um manual da ferramenta em português e que fosse apresentado de forma mais didática foi considerada uma dificuldade. Quando os alunos se deparavam com algum problema, tinham que recorrer à documentação da Bugzilla, disponível on-line em inglês. Embora a maioria dos alunos de graduação domine o inglês, um manual em português e mais explicativo (com mais exemplos) teria sido útil para motivá-los. Por outro lado, com a realização deste trabalho, os alunos tiveram a oportunidade de aprender termos técnicos utilizados comumente em

computação de uma forma que, individualmente, eles puderam assimilar os conceitos relacionados aos termos e procurar suas próprias traduções e interpretações.
- O tempo total gasto para o planejamento e correção dos trabalhos realizados pelos alunos foi de aproximadamente 45 horas. Outro valor que auxilia na quantificação do esforço gasto durante a correção dos trabalhos se refere ao número de requisições de leitura de dados efetuadas ao servidor, que foi de aproximadamente 200 requisições. Assim, considerando-se que os cursos oferecidos a turmas de graduação geralmente possuem curta duração e o professor responsável precisa se preparar para ministrar aulas sobre diversos temas, observou-se que foi fundamental a colaboração de outras pessoas, como monitor da disciplina e estudantes que já conheciam em profundidade a ferramenta utilizada, para que o trabalho prático pudesse ser realizado.

Finalmente, alguns comentários sobre os resultados alcançados em relação aos objetivos deste trabalho são destacados:

- **Objetivo 1** (*quanto a oferecer ao estudante a oportunidade de participar de um projeto desenvolvido de forma distribuída, em que a importância da comunicação entre os membros da equipe pudesse ser valorizada*): a maneira como a atividade foi planejada ofereceu a oportunidade de desenvolvimento de um trabalho de forma distribuída; os dados coletados demonstram que a maioria dos alunos usufruíram dessa oportunidade. Certamente, a baixa complexidade das funções nas quais a manutenção foi realizada não permitiu discussões longas (e até mesmo o exercício de um ciclo de vida mais diversificado para os *bugs*). No entanto, o esforço dos estudantes para entender os recursos de troca de mensagens oferecidos pela Bugzilla e o número total de mensagens trocadas demonstraram o interesse dos mesmos em desenvolver o trabalho de uma forma diferente daquela com a qual estavam acostumados.
- **Objetivo 2** (*quanto a aprimorar a formação dos estudantes em relação ao cumprimento da atividade de controle de alterações*): em geral, os estudantes cumpriram corretamente a tarefa solicitada, demonstrando o entendimento do processo de controle de alterações. Como resultado, a nota média dos estudantes foi 8,06 neste trabalho.
- **Objetivo 3** (*quanto a apresentar ao estudante um cenário de uso da ferramenta Bugzilla, de forma que ele pudesse interagir neste cenário, utilizar a ferramenta, fazer críticas em relação ao seu uso e analisar os potenciais benefícios em utilizá-la em outras atividades de Engenharia de Software*): os resultados do questionário demonstram que eles estão conscientes da importância do controle de alterações e que eles souberam analisar os benefícios e os problemas da ferramenta que utilizaram em um contexto mais amplo, considerando as diversas práticas de Engenharia de Software. Os dados coletados na própria ferramenta Bugzilla (por exemplo, aqueles apresentados na Tabela **??**) demonstram que eles de fato utilizaram os recursos disponíveis e, portanto, as afirmações apresentadas como respostas do questionário foram decorrentes da experiência prática dos estudantes, ou seja, não expressaram simplesmente "aquilo que o professor deseja ouvir".

5. Conclusões

Com o desenvolvimento do trabalho prático apresentado neste artigo, os estudantes tiveram a oportunidade de refletir sobre a importância da comunicação durante o desenvolvimento de software, cumprir na prática o processo de controle de alterações de um software e avaliar os benefícios e os problemas de uma ferramenta de software livre que oferece suporte à prática de Engenharia de Software.

Estimulando o uso da Bugzilla, que é uma ferramenta livre, em um curso de Engenharia de Software, pretende-se propiciar ao aluno a experiência de utilizar uma ferramenta que pode ser moldada às suas necessidades. Caso em seu futuro profissional o aluno se depare com outras ferramentas adquiridas pela empresa, poderá compará-las com as ferramentas livres utilizadas na graduação, aguçando seu espírito crítico. Se ferramentas similares livres estiverem disponíveis, o aluno pode vir a indicar a sua substituição na empresa, contribuindo para a diminuição de custos.

Além disso, tendo experimentado ferramentas cooperativas durante o curso de graduação, espera-se que o aluno ganhe experiência no trabalho em grupo, especialmente no caso em que a equipe está distribuída geograficamente, e não sofra tanto impacto quando iniciar sua vida profissional.

Um problema observado na realização deste trabalho prático foi o fato de que não foi possível analisar os *logs* de acesso do Apache [2]. Assim, não foi possível verificar se os alunos estavam utilizando diferentes máquinas (analisando o IP das máquinas que acessaram o servidor) para realizar o trabalho, estabelecendo, de fato, um ambiente distribuído. Caso este trabalho seja replicado, uma alternativa para resolver o problema é configurar o servidor de forma que dados mais detalhados sobre requisições para criação e alteração de *bugs* (por exemplo, o IP da máquina solicitante) sejam armazenados.

Para experimentos futuros, planeja-se incluir, entre as alterações solicitadas aos alunos, manutenções do tipo corretiva no software. Isso poderá ser feito inserindo-se propositalmente erros no código fonte, fazendo com que os alunos tentem encontrar esses erros por meio de técnicas de teste abordadas em aula. Assim, após encontrar o erro eles utilizarão a Bugzilla para gerenciar a correção.

[2] servidor web utilizado na máquina na qual a ferramenta Bugzilla estava instalada