Ensino de Engenharia de Software: Relato de Experiências

Lyrene Fernandes da Silva[1], Julio Cesar Sampaio do Prado Leite[1], Karin Koogan Breitman[1]

[1]Departamento de Informática – Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)

{lyrene, Julio, karin}@inf.puc-rio.br

Resumo. *Este artigo relata a experiência adquirida com um método de ensino aplicado à disciplina Princípios de Engenharia de Software oferecida pela PUC-Rio aos alunos de Engenharia de Computação. O método visa despertar o interesse dos alunos em engenharia de software, apresentando os aspectos teóricos e oferecendo uma experiência prática da aplicação destes conceitos. Nesta experiência prática adotamos uma filosofia de desenvolvimento cooperativo, calcado no processo Extreme Programming (XP).*

Abstract. *This paper presents our experience teaching Software Engineering. The proposed method aims to motivate the students in this discipline. We present the theoretical aspects and explore them in a study case. In this experience we use a cooperative development philosophy based on the Extreme Programming process (XP).*

1. Introdução

Há um consenso entre os docentes de disciplinas de informática a cerca de que é necessário trabalhos práticos para que os alunos consigam assimilar o conhecimento transferido [Tomayko87][Schneider03][Stevens01]. No caso da disciplina Engenharia de Software estes estudos de caso normalmente são realizados de maneira a enfatizar a aprendizagem de aspectos tecnológicos tais como linguagens de especificação, métodos e ferramentas de apoio. Desta forma, os aspectos sociais do desenvolvimento, tais como cooperação e comunicação entre os desenvolvedores, deixam de ser trabalhados [Hazzan02].

A disciplina Princípios de Engenharia de Software (PES) é oferecida semestralmente aos alunos do curso de Engenharia de Computação. Seu foco é apresentar e vivenciar os principais conceitos e práticas utilizadas e estudadas na área de Engenharia de Software, fornecendo uma visão geral de como desenvolver software sistematicamente. O método utilizado para ensino de PES apresentado neste artigo aborda os dois aspectos, técnico e social. Para isto, utilizamos o processo de desenvolvimento XP modificado e alguns métodos de apoio.

O processo XP é um método ágil de desenvolvimento. Em geral, métodos ágeis enfatizam aspectos humanos do desenvolvimento de software [Bergin04]. Como definido no manifesto para desenvolvimento ágil de software: *"Individuals and interactions over processes and tools; Working software over comprehensive documentation; Customer collaboration over contract negotiation; Responding to change over following a plan"* [Cf. Manifesto for Agile Software Development at http://agilemanifesto.org/].

O XP tem despertado interesse de educadores em engenharia de software e suas práticas aplicadas em diversos cursos [Bergin04]. Em [Schneider03], há uma análise de como o uso de XP em estudos de caso em engenharia de software impede e/ou ajuda a atingir os objetivos educacionais [SEEK] e as dificuldades encontradas em implantar o método no ambiente acadêmico. O método aqui apresentado mostra como contornamos estas dificuldades e como minimizamos os efeitos das desvantagens do XP para o ensino. Nosso objetivo é mostrar nossa experiência em tornar a disciplina PES mais prática sem causar detrimento à aprendizagem dos conceitos de Engenharia de Software.

O restante do artigo está organizado como a seguir: na Seção 2 apresentamos os desafios e a problemática referente ao ensino desta disciplina; na Seção 3 descrevemos o método de trabalho utilizado para contornar os problemas identificados e relatados na Seção 2; os resultados obtidos e melhorias na qualidade de ensino desta disciplina são apontados na Seção 4; na Seção 5 descrevemos alguns trabalhos relacionados; na Seção 6 apresentamos nossas conclusões e trabalhos futuros que visam melhorar o método de ensino aqui apresentado.

2. Contexto da Experiência

O corpo discente da disciplina PES é formado em sua maioria por alunos que estão em média no sétimo período de Engenharia de Computação. Esta disciplina representa o primeiro contato com o desenvolvimento de software como uma atividade de engenharia. Antes desta, os alunos cursam disciplinas referentes ao ciclo básico de Engenharia (Cálculo, Física, Química dentre outras) e as disciplinas Introdução a Ciência da Computação, Programação em Ponto Grande e Estrutura de Dados. Assim, eles vêem o desenvolvimento de software como uma atividade, predominantemente, de programação.

Em PES temos o objetivo de apresentar os conceitos desenvolvidos pela Engenharia de Software [Sommerville01]. *S*eu objetivo é: "*Conscientizar o aluno sobre o papel da Engenharia de Software na produção de sistemas que envolvam a produção ou compra de software. A ênfase é demonstrar que software é um produto, mas com características diversificadas dos produtos normalmente produzidos pelas engenharias clássicas. Mostrar ao aluno alguns dos métodos, técnicas e ferramentas desenvolvidas para apoiar a produção de software dentro dos níveis de qualidade/custo exigidos pelo cliente*" [Ementa]. Sua ementa é: "*Qualidade de software; Requisitos, especificação, projeto, implementação, teste e verificação de grandes sistemas de software; Estudo e uso de metodologias e ferramentas; Programação em grupo"* [Ementa].

Além de expor as informações a respeito deste domínio temos o intuito de fazer com que os alunos construam seu próprio conhecimento, incentivando-os a buscar outras fontes de informação e ajudando-os a criticar o conhecimento adquirido. A disciplina tem duração de um semestre com aproximadamente 66 horas de aula. Segmentamos o tempo do curso em duas partes, na primeira, cerca de 30 horas de aula, apresentamos os principais pontos teóricos que embasam a engenharia de sistemas de software, na segunda parte do tempo, 32 horas, dedicamos ao desenvolvimento de um software. No entanto, o curso assume que na parte prática os alunos dedicarão ao trabalho em média quatro horas semanais, extra-classe. Quatro horas do curso são dedicadas a duas avaliações.

O desenvolvimento de um software visa fazer com que os alunos possam vivenciar uma situação parecida com a real e assim identificar as dificuldades e a necessidade de sistematizar a atividade de construção de software, seja ele um software de pequeno ou

grande porte, simples ou complexo e que para obter sucesso (entenda sucesso como a aplicação da engenharia de maneira a favorecer e não retardar o desenvolvimento de um software) é necessário escolher as técnicas mais adequadas a cada problema e considerar os recursos disponíveis.

PES é oferecida desde 1998. Até 2001 o trabalho prático era baseado no paradigma da orientação a objetos, e as linguagens UML e Java eram utilizadas para desenho da arquitetura e implementação, respectivamente. Os grupos para o trabalho prático eram compostos, em média, por 3 alunos. Excelentes trabalhos foram desenvolvidos neste tempo e estão disponíveis em [PES].

A partir de 2002 passamos a adotar o processo Extreme Programming para guiar o desenvolvimento. Desde então, a infra-estrutura de implementação passou a ser voltada para WEB baseada nas linguagens PHP, HTML e XML e banco de dados MySQL ou PostgreSQL. Desta forma os alunos se prenderam menos à aprendizagem de linguagens de modelagem e programação e se dedicaram mais à aprendizagem e vivência de conceitos como cooperação, reuso, gerência de configuração, dentre outros e suas dificuldades. Como o processo XP é mais dinâmico que os tradicionais, permite que os integrantes das equipes de desenvolvimento desempenhem diferentes papéis e experimentem trabalhar em grupos maiores, cenário real durante o desenvolvimento de software.

Estamos vivenciando o quarto semestre desta experiência. Constatamos neste tempo que alcançamos resultados relevantes na aprendizagem e prática dos conceitos de engenharia de software visto que os alunos passaram a participar mais ativamente nas aulas e a questionar sobre os mais variados aspectos da engenharia de software, muitas vezes por sentirem que o processo de desenvolvimento adotado não trata a contento tais aspectos.

Para tornar estes aspectos mais explícitos para os alunos, precisavamos escolher um estudo de caso (processo, ambiente e produto) que os fizesse vivenciar as dificuldades sociais do desenvolvimento (como a comunicação) sem deixar de ensinar aspectos técnicos de modelagem e implementação. Então, consideramos que o estudo de caso devia conter várias falhas para que os alunos questionassem e construíssem uma visão crítica sobre as técnicas de engenharia de software, lembrando sempre que não existe uma "bala de prata" [Brooks87]. Na próxima Seção descrevemos como contornamos os desafios relatados aqui.

3. Método de Trabalho

Durante a primeira parte do curso as aulas são expositivas mas os alunos são levados a construir seu próprio conhecimento, entregando, semanalmente, um resumo do assunto visto. Este resumo é baseado num esquema de documentação com cabeçalho, rodapé e três seções. O cabeçalho deve conter: título, nome do grupo ou do aluno, data e versão. O rodapé deve conter: número da página e total de páginas. As seções são as seguintes: "o que aprendi", "o que não entendi", "o que duvido", e a "bibliografia consultada". A Figura 1 ilustra o padrão adotado para os resumos.

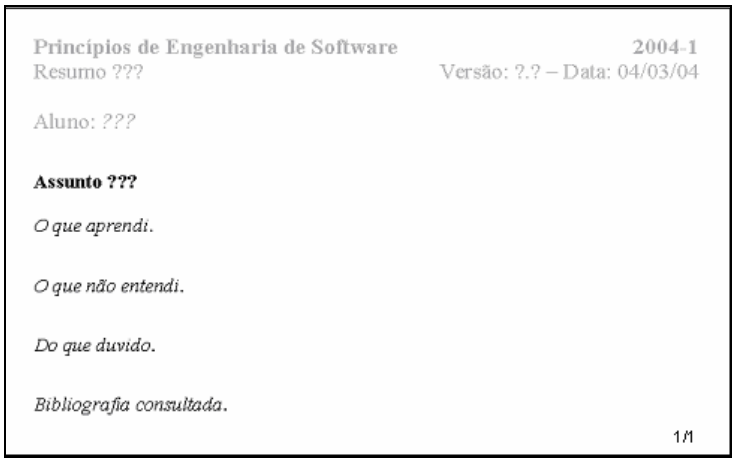

Figura 1. Padrão adotado para os resumos

Além disto, eles começam a se familiarizar com o ambiente WEB, pois seus resumos devem ser entregue através da página da disciplina onde cada aluno tem um login e senha. Nesta mesma página o professor disponibiliza resumos e atribui as notas das avaliações realizadas. Para cada resumo é atribuído quatro notas referentes à corretude, apresentação, amplitude e outros (bibliografia consultada, exercício, exemplo). Nas Figuras 2a) e 2b) ilustramos a interface do ambiente desenvolvido para entrega de resumos e atribuição de notas, respectivamente. No final das aulas teóricas os alunos são submetidos à primeira prova. Essa visa avaliar a aprendizagem dos conceitos apresentados.

Figura 2. a) Inteface do ambiente para atribuição de notas pelo docente; b) Interface do ambiente para submissão de resumos pelos alunos

Para aplicar esses conceitos num cenário real de desenvolvimento de software selecionamos um processo de desenvolvimento e uma aplicação de maneira a oferecer uma maior quantidade de conhecimento teórico e prático sobre Engenharia de Software. A escolha do XP foi também por sua aplicabilidade a projetos pequenos, do porte que geralmente se desenvolve em disciplinas acadêmicas e melhor se adequa ao nível de proficiência dos alunos em métodos e técnicas de engenharia de software. Além disto, no XP as iterações do ciclo Planejamento-Teste-Codificação-Desenho são breves e há uma maior dinamicidade nos papéis a serem assumidos, fazendo com que os alunos percebam os resultados mais rapidamente e experimentem diferentes papéis no desenvolvimento. Na Seção 3.1 descrevemos as principais características do XP e como o adaptamos aos nossos propósitos.

Como o processo XP não prioriza a documentação do projeto de software escolhemos uma aplicação que trate deste assunto como maneira de tornar este tópico um assunto experimentado no estudo de caso. A aplicação escolhida é uma ferramenta para modelagem de requisitos através da descrição dos termos do Léxico e de Cenários. Para desenvolvê-la os alunos precisaram entender estas duas técnicas e usá-las. Na Seção 3.2 apresentamos a aplicação desenvolvida.

Nesta segunda parte do curso, o processo XP é apresentado e é explicada a maneira em que os alunos irão trabalhar e a infraestrutura que irão utilizar. A turma é convidada a formar equipes de aproximadamente 10 alunos, dependendo do tamanho da turma. Cada equipe deve se estruturar em um gerente, um historiador e pares de programadores, divulgando o nome da equipe e sua estrutura. As aulas passam a ser realizadas no laboratório de computação, deixam de ser expositivas e começam a ser como reuniões: cada equipe tem cerca de uma hora de reunião em cada aula. Nas reuniões os alunos são levados a conversar com o cliente e o consultor (papéis desempenhados pelos docentes) de maneira a obter os requisitos do sistema, tirar dúvidas e a relatar o andamento do projeto.

A infraestrutura disponibilizada para os alunos no laboratório é composta de: um servidor WEB com PHP e o banco de dados MySQL instalados. Cada equipe possui uma conta neste servidor e consegue acessá-lo somente através de FTP. Sendo assim, cada equipe precisa definir uma política de gerência de configuração, já que várias pessoas podem estar trabalhando sobre o mesmo arquivo de código. Como o horário da aula não é suficiente para realizar o trabalho, aconselhamos os alunos a criarem uma lista de discursão e a usarem programas de chat, temos utilizado o YahooGroups e o ICQ, respectivamente. Esta dificuldade serve para fazer com que os alunos vivenciem o desenvolvimento cooperativo mesmo estando longe uns dos outros.

O uso de código encontrado na internet é incentivado, e os melhores trabalhos realizados pelos grupos de PES são mantidos na página da disciplina para que sejam consultados e reutilizados. Desta forma, fazemos com que os alunos percebam as vantagens do reuso, e as dificuldades de evoluir um sistema no qual eles não fizeram parte do desenvolvimento, se este não estiver apropriadamente documentado.

3.1. O processo de desenvolvimento utilizado - Extreme Programming

O XP [Beck99] é um processo de desenvolvimento de software definido basicamente por quatro atividades: Planejamento, Teste, Implementação e Desenho, veja na Figura 3. Na fase de Planejamento os requisitos são coletados e negociados com o cliente. Em seguida, os testes são elaborados a partir das estórias do cliente, e a codificação é realizada visando atender estes testes. Por fim, o sistema é novamente desenhado (ou reconstruído) a medida que novas funcionalidades são incorporadas. Estas fases ocorrem iterativamente até que o produto esteja pronto.

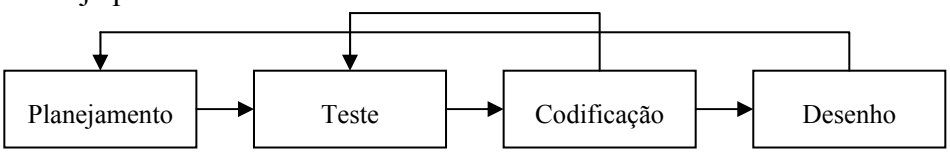

Figura 2. Processo do Extreme Programming [Beck99]

Em PES, adotamos o tempo de 3 semanas para cada ciclo. O número de integrantes dos times de desenvolvimento é entre 10 e 8 pessoas por equipe. Os papéis são divididos

em um gerente, um historiador e três ou quatro pares de programadores. O papel de cliente foi exercido pelo professor da disciplina e o de consultor pelo monitor da disciplina. O processo XP é regido por doze princípios [Beck99]. A seguir resumimos estes princípios e como eles foram adotados em PES:

1. O jogo do planejamento – Visa rapidamente determinar o escopo da próxima versão, combinando prioridades de negócio e estimativas técnicas. Em PES o cliente e o consultor se reúnem com as equipes duas vezes por semana. Nestas reuniões há uma permanente negociação de requisitos e prazos. Ao mesmo tempo, o professor esclarece as dúvidas surgidas e contrasta o XP com outros métodos quando surgem dificuldades, assim os alunos percebem as limitações do método adotado.

2. Pequenas versões – Visa produzir um sistema simples rapidamente, planejando novas versões em um ciclo muito pequeno. Os ciclos duram três semanas. Em PES há um total de 3 ciclos para desenvolver a aplicação inteira. No fim de cada ciclo as equipes apresentam uma versão completa do sistema. Como os alunos precisam relatar sobre o andamento do projeto em todas as reuniões então eles tentam ter sempre algo novo a mostrar para o cliente, construindo pequenas versões.

3. *Refactoring* – Os programadores reestruturam o sistema sem mudar seu comportamento para remover duplicação, melhorar a comunicação, simplificar ou adicionar flexibilidade. Em PES, esta prática é motivada pelo professor. Apesar de não haver cobrança desta prática, os alunos são levados a realizá-la já que alguns cenários se repetem, podendo ser implementados de maneira genérica para evitar retrabalho.

4. Programação em pares – Todo o código produzido é escrito com dois programadores em cada máquina. Em cada ciclo trocam-se os pares de programadores. Em PES instruímos o gerente de cada equipe a atribuir o trabalho de implementação a grupos de dois programadores. Devido à limitação de tempo e espaço disponíveis os alunos trabalham em pares, mas normalmente não na mesma máquina.

5. Propriedade coletiva – Qualquer pessoa pode mudar qualquer código em qualquer tempo. Em PES, esta prática é motivada pelo professor.

6. Integração contínua – Visa integrar e construir o sistema muitas vezes ao dia, todas as vezes que uma tarefa tiver sido finalizada. Em PES, os integrantes de cada equipe trabalham sobre uma base centralizada, assim, sempre que terminam alguma parte de seus trabalhos eles precisam disponibilizá-la naquela base, e assim integrá-la às partes restantes.

7. 40 horas por semana – Impõe que o time não trabalhe mais que 40 horas por semana. Mas se necessário que não faça hora-extra duas semanas seguintes. A prática de 40 horas por semana não foi adotada tendo em vista as outras atividades dos alunos. Assim, incentivamos arduamente a comunicação via e-mail e chat entre eles além de reforçar a importância do horário de aula mesmo quando eles não estão em reunião com o cliente e o consultor.

8. Cliente *on-site* – O cliente deve fazer parte do time e estar disponível em tempo integral para responder perguntas. Esta prática também não pôde ser adotada mas o cliente está presente durante uma hora duas vezes por semana e o consultor pode responder mensagens eletrônicas a qualquer hora.

9. Metáfora – Todo o desenvolvimento é guiado por uma simples estória compartilhada de como o sistema como um todo funciona. A metáfora ajuda o time a ter um mesmo entendimento sobre o vocabulário utilizado pelo domínio do projeto e assim ajuda-o a nomear funções e variáveis apropriadamente. Em PES a metáfora usada é "Editor de Cenários e Léxico em hiper-texto". Cenários é uma técnica de modelagem de requisitos. Eles descrevem situações do mundo real que ocorrem num certo contexto. Em PES substituimos as user-stories (estórias do usuário ou estorietas) por cenários para representar as funcionalidades que o sistema deve oferecer e guiar o desenvolvedor, mas ao contrário do que ocorre em XP, os cenários não são descartados depois. Eles servem para fazer estimativas e para elaborar os testes de aceitação. Eles cumprem as funções das estórias do usuário e além disto eles podem ser detalhados e utilizados como desenho, auxiliando a programação. Utilizar esta representação de cenários como 'user-stories' é uma proposta definida em [Breitman02]. Na segunda reunião de cada ciclo os alunos devem entregar ao professor os cenários que serão implementados naquele ciclo.
10. Padrões de codificação – Os programadores devem escrever o código de acordo com os padrões definidos pelo time, dando ênfase à comunicação através do código. Assim, o código é a única documentação mantida no processo. Em PES, aconselhamos as equipes a definirem seus padrões de codificação baseando-se nos padrões que eles aprenderam na disciplina Programação em Ponto Grande. E determinamos que eles devem utilizar os cenários para comentar o código. Os cenários apoiam o desenvolvedor durante a implementação pois esclarecem as pré e pós-condições de cada módulo, e facilitam a enumeração das funcionalidades encapsuladas em cada porção de código, veja [Silva03] para maiores detalhes.
11. Desenho simples – O sistema deve ser projetado tão simples quanto possível. A complexidade extra é removida assim que detectada. Em PES, com a definição dos cenários as equipes conseguem fazer um melhor planejamento da arquitetura do código.
12. Teste contínuo – Os programadores continuamente escrevem unidades de teste e os executam para que o processo de desenvolvimento continue. Em cada ciclo os programadores apresentam uma versão testável do software ao cliente. Os clientes escrevem testes para validar o sistema. Em PES, com os cenários disponibilizados pelos alunos em cada ciclo, o professor elabora casos de testes para esses cenários e disponibiliza-os antes que o ciclo termine para que os alunos possam testar a aplicação continuamente.

A notação para cenários escolhida é a proposta em [Leite97]. Esta representação ao mesmo tempo que facilita a compreensão através da utilização de linguagem natural, força a organização da informação através de uma estrutura bem definida. Um cenário possui as seguintes propriedades: título, objetivo, atores, recursos, contexto, restrições, episódios e exceções.

Os cenários podem ser tão genéricos ou detalhados quanto se queira. Como no XP não há uma preocupação em realizar uma modelagem mais precisa utilizamos os cenários de maneira que as estórias dos usuários possam ser mais estruturadas e permitindo que o desenvolvedor possa identificar com mais precisão um plano de atividades para realizar cada requisito do cliente e não partir dos requisitos direto para a programação.

O processo de construção de cenários está relacionado a existência do Léxico Ampliado da Linguagem (LAL). O LAL é implementação de conceitos de semiótica num hipergrafo [Leite93], onde a noção e a denotação são expressas para cada símbolo do léxico. Estas descrições seguem dois princípios: o da circularidade e o princípio de vocabulário mínimo. O princípio da circularidade impõe que cada descrição de denotação ou conotação faça referência a outros símbolos da linguagem. As partes da descrição que não são símbolos devem ser de um sub-conjunto reduzido de palavras com significado bem definido (vocabulário mínimo).

3.2. A Aplicação desenvolvida – Editor de Léxico e Cenários

A aplicação desenvolvida oferece as funcionalidades de edição de léxico e cenários e gerenciamento de projetos e usuários. É uma aplicação desenvolvida em PHP e está disponível em sua terceira versão em [C&L]. Para desenvolvê-la os alunos se depararam com a necessidade de tratar as diversas propriedades intrínsecas não só à edição, mas também à elaboração do LAL e dos cenários de um projeto, visto que os alunos têm que verificar e validar a aplicação que estão construindo. A verificação ajuda-os a planejar e editar corretamente o LAL e os cenários. A validação ajuda-os a levantar os requisitos de edição de maneira a oferecer um maior apoio a estas atividades. Além disto, eles precisam elaborar uma boa base de testes para validar a aplicação.

Verificação e Validação são duas atividades realizadas no processo XP através dos testes. Os testes são realizados constantemente entre os pares de programação e a cada versão são realizados também pelo cliente. Como no caso em questão o cliente é o professor este gera casos de teste que realmente avaliam os limites do software, a corretude, e se o software está dentro do nível de aceitação, seriam uma espécie de casos de teste ideais. Assim, os alunos identificam e aprendem quais os pontos críticos que merecem atenção quando se utilizam Cenários e Léxico na modelagem de Requisitos.

4. Resultados Obtidos

Adotar este método de ensino na disciplina Princípios de Engenharia de Software nos fez perceber que os alunos se tornaram mais participativos e críticos no que diz respeito a distinguir os diversos processos de desenvolvimento existentes e a questionar em quais situações um método é mais adequado que outro.

O desenvolvimento com o processo XP ocorre através de pequenas iterações. A elicitação, modelagem e análise dos requisitos ocorre permanentemente visto que o cliente está sempre disponível. O desenho é realizado através do detalhamento dos cenários e alguns rascunhos da arquitetura de partes críticas do software, e do banco de dados. E por fim, a implementação que é a atividade central do processo XP.

Muitas dificuldades encontradas pelos alunos para desenvolver o software sob as condições impostas levam-os a identificar falhas e a discutir o que seria mais apropriado em tais situações e a perceber o porquê dos diversos aspectos estudados pela disciplina de engenharia de software. A seguir, discutimos algumas destas dificuldades:

- Equipes com 8 a 10 integrantes – No início os alunos acham a idéia excelente pois pensam que assim todos vão ter pouco trabalho, mas logo percebem que nem todos os componentes têm os mesmos interesses, habilidades e preferências. E não demora muito para que justifiquem o atraso do trabalho com a ausência de um integrante da

equipe ou do trabalho dele. A comunicação é mais difícil e manter todos motivados e a par de todas as decisões de projeto e implementação é uma tarefa difícil. Assim, eles começam a perceber a importância da documentação e planejamento prévio da arquitetura.

- Infra-estrutura de trabalho – A infra-estrutura centralizada acessada somente através de FTP faz com que os alunos montem seu próprio servidor para desenvolver e estabeleçam uma estratégia para controle de versões tão logo seus primeiros arquivos de código sejam sobrescritos por versões mais antigas. Quanto às linguagens utilizadas PHP, XML, e HTML, os alunos reclamam no início de PHP porque a maioria deles não conhece, mas logo se habituam à linguagem.
- Iterações curtas – Já que o ciclo elicitar-modelar-implementar-testar é muito curto (às vezes ocorre no tempo de uma aula), inicialmente os alunos não percebem a necessidade de fazer um planejamento prévio, simplesmente alocam partes diferentes do sistema a pares de programadores diferentes. Assim, logo surgem componentes de código com funcionalidades que sobrepõem umas as outras, exigindo que os alunos reestruturem o código. No segundo e terceiro ciclos de desenvolvimento XP, a complexidade da aplicação aumenta mas tendo a experiência anterior e conhecendo mais sobre a arquitetura do sistema eles conseguem distinguir e realizar todas estas rotinas apesar delas ocorrerem quase simultaneamente.
- Foco na codificação e falta de documentação – os alunos percebem que a comunicação entre eles muitas vezes pode ser facilitada por pequenos documentos como diagrama de entidade-relacionamento do banco ou mesmo pequenos comentários no código. Apesar de ser muito atrativa a idéia de reuso logo eles percebem a dificuldade quando é necessário modificar qualquer parte do código reusada que não tem qualquer documentação. Percebem também que nem todo código é reusável pois às vezes possuem estruturas muito complicadas ou alto acoplamento com outros componentes.

5. Trabalhos Relacionados

A disciplina Engenharia de Software vem sendo discutida em diversos workshops tais como WEI, ITCSE, SIGCSE. O interesse pelo processo XP ou por algumas de suas práticas como por exemplo a programação em pares tem crescido [McDowell03]. Isto retrata um certo amadurecimento dos aspectos técnicos do desenvolvimento de software, e o início da preocupação nos aspectos sociais que impactam grandemente na engenharia de software. Sabe-se que o mercado está mudando constantemente e cada vez mais rápido, consequentemente, as técnicas utilizadas mudam com a mesma velocidade e é necessário estar apto a se adaptar a tais mudanças. É necessário oferecer aos estudantes um conjunto de ferramentas para elaborarem uma solução, incluindo a habilidade para adaptarem os processos de maneira a atender suas necessidades.

Há diversas propostas para o ensino de engenharia de software tais como:

Em [Tomayko87], distingue-se quatro modos em que esta disciplina é ministrada: aulas expositivas; aulas expositivas e seminários; aulas expositivas e estudo de caso com pequenas equipes; e estudos de caso com grandes equipes, de 15 a 30 componentes, o instrutor sendo o gerente da equipe. Cada forma tem suas vantagens e desvantagens. Ele adota a última e relata sua experiência num curso de 1 semestre. Em [Tomayko91] ele

relata sua experiência, dificuldades e vantagens em utilizar esta abordagem em um curso de 16 meses.

Em [Bergin04], resumem-se as discussões tidas em um painel sobre ensino de métodos de desenvolvimento em geral e XP em particular. Em [Hazzan02] demonstra-se dez princípios de ensino neste tópico e exemplifica-se como fazer isto com o processo XP. Em [Hazzan01], demonstra-se a preocupação em tratar os aspectos humanos do desenvolvimento de software. Em [Schneider03], há uma análise de como o uso de XP em estudos de caso em engenharia de software impede e/ou ajuda a atingir os objetivos educacionais, propostos em [SEEK], e as dificuldades encontradas em implantar o método no ambiente acadêmico. Stevens, em [Stevens01], relata sua experiência e a reação dos alunos em um curso de um semestre, o estudo de caso foi realizado por grupos pequenos, de 3 integrantes. Em [Hedins03] há um relato da experiência obtida com o XP aplicado ao ensino de cursos introdutórios de Engenharia de Software.

Todos estes trabalhos estão relacionados ao nosso por compartilharmos as mesmas dificuldades no ensino desta disciplina ou por compartilharmos alguma parte da idéia de como contorná-las. O estudo de caso descrito em [Hedins03] é o que mais se asemelha ao nosso. Hedins utiliza o XP em grupos de 8 a 10 alunos. Estes times desenvolvem a mesma aplicação. E o intuito não é ensinar XP em si, mas sim usar XP como veículo para ensinar os conceitos básicos de engenharia de software, tais como, desenvolvimento iterativo, gerência de configuração e comunicação. Vale ressaltar também o enfoque evolutivo, face à existência de implementações anteriores às quais procuram-se agregar novos requisitos tanto funcionais quanto não-funcionais. Agregamos a estas características o uso de técnicas como Cenários e LAL, e cenários como padrão de comentários no código. É essencial também, a primeira parte do curso composto de aulas expositivas, que com a cobrança dos resumos, os alunos são levados a criar uma visão crítica dos conceitos ensinados.

6. Conclusões e Trabalhos Futuros

O assunto “Engenharia de Software” é muito extenso e não absorvido pelos alunos se este não estiver entrelaçado com um trabalho prático. Entretanto, realizar um trabalho prático que queira abranger todas ou grande parte das técnicas difundidas pela Engenharia de Software é além de impraticável, tedioso e inadequado para o tipo de aplicação que se faz em disciplinas introdutórias de Engenharia de Software. Impraticável porque são muitas, inadequado porque normalmente os alunos desenvolvem aplicações razoavelmente simples, e tedioso porque para tais aplicações os alunos não entendem para que tantos métodos, modelos e técnicas se eles se acham capazes de construir o software sem lançar mãos de muitos artifícios.

Abordar aspectos sociais do desenvolvimento ao invés de abordar somente aspectos técnicos mostrou-se ser uma boa abordagem. Por exemplo, realizar o trabalho com equipes grandes, se comparado com o que se adota normalmente quando utilizamos outros métodos, é muito interessante porque transmite o cenário real de uma equipe de desenvolvimento. Os vários problemas de comunicação e gerência surgem, fazendo com que os alunos percebam o sentido de se estudar os diversos ramos da Engenharia de Software. A falta de documentação característica do processo XP mostrou aos alunos os custos de tempo e esforço despendidos quando eles têm que reutilizar o código escrito pelas equipes dos semestres anteriores ou pelos outros integrantes de sua própria equipe.

Estamos investigando algumas mudanças para o método de ensino de Engenharia de Software apresentado neste artigo que possam melhorar os resultados obtidos. Uma dessas linhas é um aprofundamento do entendimento da nossa experiência. Para isso pretendemos fazer um levantamento quantitativo, através de questionários, de que aspectos são mais bem absorvidos pelos alunos. Uma outra linha é investigar como os conceitos de arquitetura de software podem ajudar na reestruturação (*refactoring*) e qual melhor momento para que isso seja feito. Também tem sido uma preocupação constante [Leite01] como efetivamente garantir mais qualidade no processo XP. Em função disso, pensamos em adaptar o processo XP inserindo a atividade de controle de qualidade, através de inspeções.

Referências Bibliográficas

[Beck99] Beck, K. Extreme Programming explained. 1st edition. Addison-Wesley Publishing Company, 1999.

[Bergin04] Bergin, J., Caristi, J., Dubinsky, Y., Hazzan, O., Williams, L. Teaching Software Development Methods: The Case of Extreme Programming. Proceedings of the Thirty Fifth SIGCSE Technical Symposium on Computer Science Education, Norfolk-Virginia, USA, March 3-7, 2004, pp 448-449.

[Breitman02] Breitman, K., Leite, J. Managing User Stories. International Workshop on Time Constrained Requirements Engineering, 2002.

[Brooks87] Brooks, F. P. No Silver Bullet: Essence and Accidents of Software Engineering. Computer, April 1987.

[C&L] http://sl.les.inf.puc-rio.br/cel - acessado em abril de 2004.

[Hazzan02] Hazzan, O., Dubinsky Y. Teaching a Software Development Methodology: The Case of Extreme Programming. Proceedings of the 16th Conference on Software Engineering Education and Training (CSEE&T 2003), Madrid, Spain, March 20-22, 2003. pp 176-184.

[Hazzan01] Hazzan, O.Teaching the Human Aspect of Software Engineering – A Case Study. Proceedings of the Thirty Second SIGCSE Technical Symposium on Computer Science Education, Volume 33, Charlotte, NC, USA, February 2001, pp.124-128.

[Hedin03] Hedin, G., Bendix, L., Magnusson, B. Introducing Software Engineering by means of Extreme Programming. Proceedings of the 25th International Conference on Software Engineering, Portland, Oregon, May 03-10, 2003, pp 586-593.

[Ementa] http://www-nt.inf.puc-rio.br/cgilua/cgilua.exe/disc.htm?cxorig=1&id=119&cxid=disciplina% 5Fcurso - acessado em abril de 2004.

[Leite93] Leite, J.C.S.P. and Franco, A.P.M. A Strategy for Conceptual Model Acquisition. First International Symposium on Requirements Engineering. Proceedings. IEEE Computer Society Press, 1993. pp. 243-246.

[Leite97] Leite, J.C.S.P, Rossi, G., Balaguer, F., Maiorana, V., Enhancing a requirements baseline with scenarios. In: Third IEEE International Symposium on Requirements Engineering - RE97, Proceedings. IEEE Computer Society Press, January, 1997, pp 44-53.

[Leite01] Leite, J.C.S.P. Extreme Requirements. Proceedings of Jornadas de Ingeniería de Requisitos Aplicada, JIRA, Univesidade de Sevilha, Junho de 2001, pp. 1-13.

[McDowell03] McDowell, C., Hanks, B., Werner, L. Experimenting with Pair Programming in the Classroom. Proceedings of the 8th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2003), June 30 - July 2, 2003, Thessaloniki, Greece.

[PES] www.er.les.inf.puc-rio.br/pes - acessado em abril de 2004.

[Schneider03] Schneider, J., Johnston L. Extreme Programming at Universities – An Educational 1Perspective. Proceedings of the 25th International Conference on Software Engineering, Portland, Oregon, May 03-10, 2003, pp 594-599.

[SEEK] Software Engineering Education Knowledge (SEEK). Second draft, available at http://sites.computer.org/ccse/, December 2002.

[Silva03] Silva, L, Sayão M., Leite J.C.S.P., Breitman, K. Enriquecendo o Código com Cenários. Anais do Simpósio Brasileiro de Engenharia de Software (SBES), Manaus, 2003.

[Sommerville01] Sommerville, Ian. Software Engineering, 6th edition, Addison-Wesley, 2001.

[Stevens01] Stevens, K. T. Experiences Teaching Software Engineering for the First Time. Proceedings of the 6th annual conference on Innovation and technology in computer science education, Canterbury, United Kingdom, 2001, pp 77-80.

[Tomayko87] Tomayko, J. E. Teaching a Project-Intensive Introduction to Software Engineering. Carnegie Mellon University, Software Engineering Institute, CMU/SEI-87-TR-20, August 1987.