PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 43/08

Fórum de Educação em Engenharia de Software

Julio Cesar Sampaio do Prado Leite
Cláudia Maria Lima Werner

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900
RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 43/08 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena Outubro, 2008

Fórum de Educação em Engenharia de Software

Julio Cesar Sampaio do Prado Leite[1] Cláudia Maria Lima Werner[2]

[1]Departamento de Informática – PUC-RIo
[2]Coope--UFRJ

www.inf.puc-rio.br/~julio, www.cos.ufrj.br/~werner

Abstract. Proceedings of the first Brazilian Meeting to discuss issues related to the education of software engineers. This meeting is in the context of the Brazilian Symposium on Software Engineering.

Keywords: Software Engineering, Education, Learning Techniques, Survey, State of the Practice.

Resumo. Anais do primeiro Fórum de Educação em Engenharia de Software realizando no âmbito do Simpósio Brasileiro de Engenharia de Software. Esses anais apresentam 14 contribuições sobre o tema.

Palavras-chave: Engenharia de Software, Educação, Técnicas de Aprendizado, Levantamento do estado da prática.

---

* **Editores** dos Anais do Fórum de Educação em Engenharia de Software

In charge of publications (**Responsável por publicações,** se o texto for em português):

Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22451-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3527-1516 Fax: +55 21 3527-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

Comitê de Programa

Alfredo Goldman USP
Ana Regina Rocha COPPE/UFRJ
Antonio Francisco Prado UFScar
Carlos Lucena PUC-Rio
Christina Chavez UFBA
Daltro Nunes UFRGS
Guilherme Travassos COPPE/UFRJ
Hugo Fuks PUC-Rio
Itana Maria de Souza Gimenes UEM
Jaelson Freire Brelaz de Castro UFPE
José Carlos Maldonado USP
José Luís Braga UFV
Karin Breitman PUC-Rio
Paulo Masiero USP
Renata Araujo UNIRIO
Roberto Bigonha UFMG
Silvio Meira CESAR
Simone Barbosa PUC-Rio
Thais Vasconcelos Batista UFRN
Vera Werneck UERJ

Sumário

Prefácio

Consideramos a primeira edição do FEES um grande sucesso. Foram submetidos ao FEES 2008 22 artigos (20 artigos completos e dois resumidos), envolvendo diversas regiões do pais, sendo 5 da região nordeste, 2 do centro oeste, 2 do sul e 13 do sudeste. Desses, 14 foram aceitos (10 completos e 4 resumidos, tendo estes últimos sido indicados para aceitação como artigos resumidos, embora submetidos como completos) para apresentação e publicação nos anais. Cada artigo foi avaliado por, pelo menos, três membros do Comitê de Programa, segundo critérios pré-estabelecidos. Os coordenadores tomaram a decisão final a partir dos comentários dos avaliadores e da relevância do tema do artigo para o fórum.

Esses números demonstram que a comunidade acadêmica de Engenharia de Software está, além de sua forte atuação de pesquisa, comprometida também em como passar para os futuros profissionais o conceito de produzir software com qualidade. Essa preocupação é fundamental para que possamos aprimorar, ainda mais, nossa posição global no que diz respeito a essa nova Engenharia.

Conforme observamos na chamada de artigos: "A educação e o treinamento adequados podem melhorar significativamente a prática de Engenharia de Software e representam, também, um pré-requisito para o avanço do estado da arte nesta área. O objetivo do Fórum de Educação em Engenharia de Software é expor, analisar e investigar as dificuldades e particularidades do ensino e da aprendizagem de Engenharia de Software e como preparar e melhorar as atividades educacionais. O SBES-FEES tem, também, como objetivo prover um fórum para a discussão de tópicos relacionados à educação e ao treinamento de Engenharia de Software, criando um canal de integração e de troca de experiências entre diferentes tipos de atores interessados no assunto." Os artigos submetidos e aceitos concordaram com essa observação.

Além da apresentação e discussão desses trabalhos, teremos também um painel no dia 16 de Outubro destinado a debater o tema, levando em consideração tanto depoimentos da área acadêmica como do governo e da indústria. Esperamos que esse debate possa, também, contribuir para aperfeiçoarmos o ensino de Engenharia de Software.

Em particular gostaríamos de lembrar aos leitores que os Anais do Fórum estão disponíveis na rede e também permitem a troca de impressões entre leitores. Com isso pretendemos manter o dialogo aberto com todos os interessados. Vejam mais no sítio de FEES (fees.inf.puc-rio.br).

Mais uma vez agradecemos aos membros do comitê de programa e aos autores. Em particular, é importante ressaltar a qualidade das revisões que recebemos.
Esperamos que o Fórum seja uma experiência positiva para todos.

Cláudia Maria Lima Werner e Julio Cesar Sampaio do Prado Leite

Ensinando Construção de Software Aplicada a Sistemas de Informação do Mundo Real *

Márcio de Oliveira Barros[1] Renata Mendes de Araujo[1,2]

[1] Programa de Pós-Graduação em Informática, UNIRIO
[2] Núcleo de Pesquisa e Prática em Tecnologia, UNIRIO

marcio.barros@uniriotec.br, renata.araujo@uniriotec.br

Abstract. In this paper we describe a case study conducted along the software construction course at the Information Systems Post-Graduate program at the Federal University of the Rio de Janeiro State (UNIRIO). In this course, we challenged students a set of tasks associated to an actual problem being faced by real organizations. Students should develop a set of functional requirements but also to adapt the system to answer to events happening in the real world, notified by open communications and the media. Therefore, we created a learning environment closer to the industrial situation that students would find in their professional career.

Keywords: case study, information systems education, programming education.

Resumo. Neste artigo, apresentamos um estudo de caso realizado durante um curso de construção de software no Mestrado de Sistemas de Informação do PPGI/UNIRIO. Neste curso, apresentamos aos alunos um conjunto de trabalhos associados a um problema contemporâneo e que também estava sendo tratado por empresas reais. Os alunos deveriam desenvolver um conjunto de requisitos funcionais, mas também adaptar o sistema em desenvolvimento para responder a eventos acontecendo no mundo real e sendo notificados através de comunicados abertos ou na mídia. Com isto, criamos um ambiente de aprendizado mais próximo a uma situação industrial, que os alunos encontrarão em suas carreiras.

Palavras-chave: estudo de caso, ensino de sistemas de informação, ensino de programação.

1 Introdução

A construção de software [Mc Connel, 1993] se refere às atividades intermediárias do ciclo de vida de desenvolvimento de software, que compreendem o projeto detalhado, a codificação e os testes de unidade. Ela é precedida pelo levantamento de requisitos e pelo projeto arquitetural, sendo sucedida pela implantação e manutenção do sistema. Se analisada isoladamente, a construção de software compreende o conjunto de atividades de engenharia em que as habilidades técnicas (relacionadas ao computador) são aplicadas de forma mais intensa que as habilidades sociais e de comunicação (relacionadas aos seres humanos).

Lehman [Lehman, 1996] propõe um esquema para classificação de sistemas de software no qual estes são divididos em três classes: S-systems, P-systems e E-systems. A classe dos S-systems compreende o conjunto de sistemas que podem ser formalmente definidos por uma especificação e derivados a partir desta – isto é, o problema a ser resolvido por estes sistemas é completamente definido para todo conjunto de circunstâncias nas quais ele pode ser usado e a solução para este problema é conhecida [Pfleeger, 1998]. A classe dos P-systems inclui os sistemas nos quais o problema a ser resolvido não pode ser totalmente descrito ou a implementação completa de uma solução teórica para este problema não é viável com os recursos computacionais disponíveis no momento. Assim, a implementação de um P-system é limitada a uma aproximação da solução teórica, eventualmente aplicada a um subconjunto das situações em que o problema pode se apresentar no mundo real. Finalmente, os E-systems são sistemas nos quais o problema a ser resolvido sofre mudanças contínuas: enquanto nos S- e P-systems o mundo real é considerado estável e constante, um E-system incorpora a natureza mutável do mundo real e precisa ser continuamente atualizado para permanecer útil aos seus usuários.

Observamos que grande parte dos sistemas de informação reais – ou seja, sistemas utilizados para apoiar processos de negócio de uma indústria ou setor econômico real – podem ser caracterizados como E-systems, cujos requisitos podem mudar em função de eventos externos ao projeto de desenvolvimento do sistema e cuja utilidade depende da capacidade da equipe de desenvolvimento de atualizar continuamente os sistemas, mantendo-os funcionais ao longo do tempo, à medida que seus requisitos mudam.

Assim, o desenvolvimento de sistemas de informação reais é uma tarefa complexa, na qual as atividades de construção de software e levantamento de requisitos estão entremeadas: como os requisitos mudam ao longo do ciclo de desenvolvimento, não existe uma divisão clara entre as atividades de construção e levantamento de requisitos. Desta forma, o conhecimento técnico (como princípios de projeto ou proficiência em linguagem de programação) e as habilidades relacionadas a seres humanos (como capacidade de comunicação, pró-atividade, planejamento, entre outras) devem ser vivenciadas em conjunto, permitindo que problemas observados em um aspecto ampliem o efeito do outro, exigindo a tomada de decisão com informação incompleta e a cuidadosa avaliação das premissas a partir das quais o desenvolvimento será conduzido.

Por outro lado, a educação de profissionais que trabalharão com desenvolvimento de sistemas geralmente separa o aprendizado de aspectos técnicos do aprendizado de atividades relacionadas com seres humanos. Geralmente os responsáveis pelo ensino de aspectos técnicos apresentam exercícios em que os alunos recebem especificações precisas (ou seja, vivenciam o desenvolvimento de S-systems), de forma que eles

possam aplicar o conhecimento técnico adquirido independente de qualquer mal entendimento sobre os requisitos. Os responsáveis pelo ensino de aspectos humanos, por outro lado, comumente apresentam estas atividades sem que um produto real seja construído a partir do resultado de sua aplicação, ou seja, sem que haja a necessidade de negociar a introdução de novos requisitos ou mudanças no plano de projeto entre os clientes e os desenvolvedores. Muitos educadores envolvidos com cursos técnicos reduzem o valor das atividades relacionadas ao ser humano ("papel aceita tudo! papel não compila!"), enquanto o segundo grupo coloca os aspectos técnicos em segundo plano ("não faz sentido escrever um sistema que ninguém quer utilizar"). Desta forma, desenvolvedores inexperientes são geralmente incapazes de lidar com aspectos técnicos e aspectos humanos ao mesmo tempo.

Acreditamos que os envolvidos no ensino de Engenharia de Software, em particular de Sistemas de Informação, deveriam apresentar aos alunos trabalhos e estudos de caso mais próximos a sistemas do mundo real (ou seja, E-systems ao invés de S-systems). Neste artigo apresentamos um estudo de caso executado em uma disciplina de Construção de Software do curso de Mestrado do Programa de Pós-Graduação em Informática da Universidade Federal do Estado do Rio de Janeiro (UNIRIO). Ao longo do curso, os alunos foram avaliados através de trabalhos relacionados com um problema que foi alvo da atenção de diversas software-houses brasileiras no mesmo período: a implementação da plataforma regulamentar Basiléia II em instituições financeiras nacionais. Os trabalhos exigiram que os alunos desenvolvessem um sistema para dar suporte a um conjunto de funções relacionadas à plataforma Basiléia II, visando auxiliar o trabalho da agência regulamentadora de instituições financeiras – no caso, o Banco Central do Brasil. A necessidade de adaptação do sistema para mantê-lo útil à medida que o Banco Central apresentava mudanças na plataforma regulamentar foi característica importante e considerada parte do trabalho, de modo que os alunos foram diretamente influenciados por eventos do mundo real. Assim, os alunos vivenciaram a construção de um E-system, visto que a especificação do trabalho mudava em função de eventos fora do seu controle e a estrutura interna do sistema deveria ser alterada para apoiar estas mudanças.

Este artigo está organizado em cinco seções. A primeira seção contém esta introdução. A seção 2 descreve a plataforma regulamentar Basiléia II, que é a base do estudo de caso. A seção 3 descreve os objetivos e organização do curso PPGI017, cujos alunos participaram no estudo de caso relatado. A seção 4 apresenta a cronologia de eventos relacionados à plataforma Basiléia II e ao curso PPGI017, ressaltando como os eventos disparados pelo Banco Central do Brasil influenciaram os trabalhos propostos ao longo do curso. A seção 5 resume as lições aprendidas com o estudo de caso e apresenta algumas conclusões que pudemos obter a partir de sua análise.

2 A Plataforma Regular Basiléia II

Em junho de 2006, o Comitê da Basiléia para Supervisão Bancária publicou a versão final da plataforma regulamentar Basiléia II. Essa plataforma foi desenvolvida como resultado de sete anos de trabalho para garantir convergência internacional no tocante a regulamentações governamentais sobre exigências de capital de instituições financeiras ativas e com negócios realizados no exterior. A plataforma Basiléia tem como objetivo criar um padrão internacional a ser seguido por agências regulamentadoras de diversos países quando elas definirem regras relacionadas a quanto capital as instituições financeiras devem depositar para assegurar as posições em instrumentos financeiros de

risco de mercado, de crédito e operacional. Com isto, as agências regulamentadoras poderiam evitar que as instituições concretizem estratégias de investimento com risco acima de sua capacidade de restabelecimento.

A plataforma Basiléia II descreve um conjunto de medidas e um padrão mínimo de exigência de capital que as autoridades nacionais deveriam implementar sob a forma de leis (domésticas) e procedimentos de comunicação de estratégias realizadas e posições de investimento [Comitê da Basiléia para Supervisão Bancária, 2008]. A plataforma procura aprimorar as regras anteriores, alinhando a exigência de capital com os riscos incorridos pelas instituições financeiras (tais como bancos, gerenciadores de fundos de pensão, empresas de seguro e resseguro, entre outras). Adicionalmente, a plataforma pretende promover procedimentos de supervisão que não se baseiem apenas em informações passadas, mas nas possibilidades de perda potencial dos investimentos de uma instituição em diferentes cenários sobre o futuro, encorajando as instituições financeiras a identificarem claramente os riscos que estão correndo e aprimorarem seus mecanismos de controle sobre estes riscos. Como resultado deste esforço, a plataforma Basiléia II é flexível e capaz de se adaptar às mudanças freqüentes nos procedimentos utilizados pelo mercado para medir e controlar riscos.

Por outro lado, a plataforma Basiléia II não pretende ser um acordo internacional no sentido jurídico: não há assinaturas ou contratos entre os países participantes. As agências regulamentadoras que tiverem interesse em adotar a plataforma podem endossar o padrão internacional e implementá-lo, criando leis que sigam seus princípios.

O Banco Central do Brasil, agência regulamentadora do sistema financeiro nacional, endossou a plataforma Basiléia desde sua primeira versão (a plataforma Basiléia I foi adotada pelo Banco Central do Brasil em 1994). No entanto, em 2004 o Banco Central adotou uma política mais conservadora em relação à segunda versão da plataforma Basiléia, criando regras que exigiam que "a maior parte das instituições financeiras adotem uma versão simplificada do padrão internacional, aprimorando sua implementação da primeira versão da plataforma e incluindo elementos que, utilizando instrumentos financeiros próprios para mitigar riscos de crédito, permitissem um melhor alinhamento entre o capital exigido e a exposição a risco" [Banco Central do Brasil, 2004]. Posteriormente, as instituições financeiras deveriam preparar seus mecanismos de controle interno – e os sistemas que oferecem suporte a estes mecanismos – de modo a serem capazes de implementar novas regras relacionadas com a plataforma Basiléia II. No período entre 2007 e 2008, a transição de uma implementação simplificada para uma implementação mais ampla da plataforma foi acelerada e o Banco Central lançou um conjunto de regras exigindo que as instituições financeiras informassem suas exposições a risco de mercado em um relatório mensal (o DRM, ou Demonstrativo de Risco de Mercado).

O DRM e os eventos que ocorreram entre janeiro e junho de 2008 compõem o contexto para o estudo de caso relatado nas próximas seções. No estudo de caso, um grupo de alunos projetou e implementou um sistema de informação para processar os valores adquiridos de um conjunto de relatórios DRM (valores aleatórios formatados como relatórios DRM), sendo influenciados por mudanças sofridas por este relatório no mundo real.

3 O Curso sobre Construção de Software

A disciplina PPGI017 – Técnicas Avançadas de Construção de Software – tem como objetivo ensinar o desenvolvimento de sistemas de alta qualidade, aplicando padrões reconhecidos nas atividades de projeto detalhado e codificação de software. Os alunos aprendem a importância de manter uma boa organização e identação no código do sistema, princípios de projeto de software, características observáveis em bons e maus projetos, tópicos relacionados à introdução de mudanças em um sistema, princípios de projeto orientado a objetos, caracterização de classes, projeto de pacotes de classes e padrões de projeto. Inclui também temas práticos relacionados à construção de software, como gerência de configuração, registro de defeitos e testes de unidade.

PPGI017 é uma disciplina opcional no currículo do curso de Mestrado em Sistemas de Informação do Programa de Pós-Graduação em Informática da Universidade Federal do Estado do Rio de Janeiro (UNIRIO). A edição de 2008 da disciplina ocorreu no primeiro semestre deste ano e foi atendida por apenas seis alunos (um número pequeno, porém esperado, de alunos de pós-graduação cujos interesses incluem projeto e codificação de software). Os instrumentos de avaliação do curso incluem uma prova escrita (realizada no meio do curso e com peso unitário) e quatro trabalhos de implementação (com peso dois), cujas notas são somadas para compor a média da disciplina com a nota da prova. Essa estratégia permite determinar se os alunos desenvolveram conhecimento teórico nos assuntos tratados pela disciplina, sem perder de vista a aplicação prática destes conceitos.

Uma das diretrizes do curso de Sistemas de Informação é que seja dada ênfase ao lado prático, ou seja, à aplicação do conhecimento adquirido pelos alunos. Neste sentido, a aplicação prática dos conceitos ensinados na disciplina PPGI017 sempre foi um forte direcionador na seleção dos projetos de curso a serem desenvolvidos pelos alunos. No entanto, é difícil identificar oportunidades de projetos que sejam suficientemente grandes para ser representativos do mundo real, mas pequenos o suficiente para que seu desenvolvimento seja viável nos quatro meses que compõem o período do curso. Além disso, seria desejável apresentar trabalhos com características próximas às encontradas em projetos do mundo real – como requisitos imprecisos, mudanças tardias nos requisitos, competição entre provedores de solução e datas inflexíveis para a entrega dos subprodutos –, ao invés de projetos com requisitos completos e estáveis. Seria desejável submeter os alunos a ambientes de desenvolvimento "não tão controlados", de modo que eles possam vivenciar problemas no ambiente de rede, falhas de comunicação, mudanças não agendadas na plataforma de trabalho, atualizações inesperadas de compiladores e ferramentas de desenvolvimento, entre outras. Afinal, estes são cenários que podem acontecer na indústria e os alunos devem estar preparados para enfrentá-los de forma adequada.

Na preparação da oferta da disciplina em 2008, buscamos um problema real que ainda estivesse em aberto e que nos desse a oportunidade de explorar a dinâmica entre a construção do sistema e o surgimento de novos requisitos, sem que os alunos pudessem justificar as imperfeições presentes nos requisitos iniciais dos trabalhos. A implementação da plataforma Basiléia II se apresentou como um exemplo perfeito para pano de fundo dos trabalhos do curso. Propusemos que os alunos desenvolvessem uma ferramenta para ajudar o Banco Central do Brasil a analisar os relatórios DRM enviados pelas diversas instituições financeiras sob sua supervisão. Cada relatório contém milhares de valores financeiros e uma ferramenta computacional que suportasse as análises destes números poderia auxiliar o Banco Central em sua tarefa

de supervisor. Para efeito de avaliação dos alunos na disciplina, o desenvolvimento da ferramenta poderia ser dividido em três ou quatro trabalhos de curso distintos.

A especificação parcial da ferramenta proposta, apresentada como descrição do primeiro trabalho, introduziu os alunos ao domínio das finanças – apenas um aluno tinha experiência anterior neste domínio – e enumerou as funções que a ferramenta deveria oferecer aos seus usuários. Nesta especificação, foram indicados links para um conjunto de documentos publicados pelo próprio Banco Central do Brasil, incluindo layouts e especificações de fórmulas para calcular os valores apresentados no relatório DRM. Estes documentos foram rigorosamente os mesmos utilizados por software-houses reais desenvolvendo sistemas para atender à implementação da plataforma Basiléia II. Os alunos puderam enviar questões e dúvidas diretamente aos responsáveis pelo projeto no Banco Central, da mesma forma desenvolvedores e líderes de projeto poderiam fazer na indústria. Atrasos decorrentes da falta de respostas puderam ser vivenciados, tal como no mundo real. Os alunos puderam ler sobre o assunto do seu trabalho nos jornais e sentir como se estivessem trabalhando em software-houses reais, desenvolvendo um software para instituições financeiras ou para o próprio Banco Central – e levaram isto muito a sério no desenvolvimento dos trabalhos. A motivação foi fascinante, tal como nunca havíamos observado em outras disciplinas de programação!

Além disso, um esquema de avaliação competitivo foi colocado em prática. Os alunos foram divididos em três grupos (A, B e C), cada qual composto por dois alunos. Cada grupo deveria apresentar resultados para cada parte do trabalho. Estes resultados foram avaliados por sua funcionalidade e também por suas propriedades internas, como legibilidade do código, simplicidade, organização do código e do projeto detalhado. A melhor solução receberia uma nota que poderia atingir 10 (dez); a segunda melhor solução receberia uma nota que poderia atingir apenas 8 (oito), enquanto a terceira estaria limitada a um máximo de 6 (seis). Empates seriam permitidos e atrasos seriam punidos com a perda de um ponto por dia útil de atraso após a data programada para a entrega. Estas regras reforçaram o senso de competição que se observa no mercado e este senso foi bem recebido pelos grupos.

4 O Relatório DRM – Cronologia dos Eventos

A seguir apresentamos os eventos que compreenderam a definição e os requisitos regulatórios do relatório DRM e como influenciaram a disciplina PPGI017, permitindo aos alunos vivenciarem o desenvolvimento de um sistema de informação sujeito a mudanças do mundo real.

- Jan/2008 – a Associação Brasileira de Instituições Bancárias, associação que apóia o Banco Central na definição dos instrumentos regulatórios relacionados ao relatório DRM, lança uma nota pública sobre o relatório, estabelecendo a maioria das fórmulas de cálculo de exposição ao risco que devem ser seguidas e o formato no qual as organizações financeiras devem entregá-lo ao Banco Central;
- Mar/2008 – A disciplina é iniciada e os alunos são apresentados ao estudo de caso da plataforma Basiléia II. Foi realizada uma apresentação sobre como a informação de exposição de risco deve ser fornecida ao Banco Central e os alunos receberam a nota pública da Associação Brasileira de Instituições Bancárias e os formatos propostos para o relatório;

- Mar/2008 (uma semana depois) – a primeira tarefa é apresentada aos alunos. Nela, eles devem desenvolver (projetar e codificar) um conjunto de classes que modelem a informação sobre exposições de riscos a serem apresentadas ao relatório DRM;
- Abr/2008 (duas semanas depois) – os alunos entregam os resultados da primeira tarefa e são avaliados pelo professor da disciplina. O grupo A apresenta a melhor entrega, apesar da organização interna do código ainda apresentar possibilidades de melhoria. Recebe a nota 9 (nove), enquanto que os grupos B e C se mantêm em segunda posição, recebendo nota 7 (sete). Um resumo da avaliação de cada grupo foi disponibilizado para todos os alunos, bem como as implementações. Isto permitiu que aprendessem com os erros alheios e se preparassem melhor para a segunda tarefa. A versão entregue pelo grupo A foi aperfeiçoada para se tornar a implementação de referência, a ser utilizada por todos nas tarefas seguintes;
- Abr/2008 (uma semana depois) – a segunda tarefa é apresentada aos alunos, na qual deveriam desenvolver classes para leitura das informações de exposição de risco de um relatório DRM para a estrutura de classes construída na tarefa anterior. Exemplos de relatórios DRM, gerados utilizando planilhas, foram entregues como base de testes para a implementação (acrescidos de um erro de formatação conhecido em uma das entradas);
- Abr/2008 (uma semana depois) – dois grupos detectaram o erro de formatação e comunicaram ao professor (A e B). Foi solicitado que corrigissem o erro manualmente no relatório e informassem ao terceiro grupo. O grupo B também identificou um erro não esperado (um erro real, não intencionalmente introduzido pelo professor da disciplina), tratado da mesma maneira. Além disso, algumas regras de negócio que não haviam sido claramente definidas na especificação (tais como o tamanho e o conteúdo de alguns campos do relatório), são estabelecidas pelo professor;
- Mai/2008 (uma semana depois) – dois grupos entregam as tarefas no prazo (B e C), enquanto que o terceiro (A) entrega com dois dias de atraso. Os três grupos se posicionam na primeria posição, embora todos pudessem ter aprimorado o projeto das classes utilizadas para implementar a funcionalidade requerida. Cada grupo recebe a nota 8 (oito) e o grupo A sofre uma penalidade de 2 pontos pela entrega tardia (terminando com nota 6). Assim como na tarefa anterior, o resumo de cada avaliação e as versões de implementação entregues são publicados e uma implementação de referência é desenvolvida para as próximas tarefas. Desta vez, no entanto, a implementação de referência é codificada "do zero" pelo professor;
- Mai/2008 (uma semana depois) – a terceira tarefa é apresentada aos alunos, pedindo que desenvolvam uma ferramenta que ofereça uma visualização gráfica dos valores de exposição a risco de um grupo de relatórios recebidos em um mesmo período (de um mesmo mês e ano). A representação gráfica deve ser customizável, permitindo aos usuários selecionar partes do relatório DRM que devem ser apresentados na tela. A ferramenta foi descrita de forma pouco detalhada e a discussão durante a apresentação da tarefa contribuiu para "aparar as arestas" de sua especificação;
- Mai/2008 (mesma semana) – O Banco Central publica uma norma regulatória descrevendo a versão final do relatório DRM. Essa versão tem diversos aspectos distintos do proposto em Janeiro. Entre estes aspectos, (a) o relatório deve incluir uma nova seção, agregando a exposição a risco de algumas de suas seções; (b) o relatório, que deveria ser publicado em formato texto, deve agora ser publicado em formato XML; (c) um vértice extra foi incluído em cada curva de mercado na qual a

instituição financeira pode apresentar exposição a risco; e (d) fatores de risco atômicos (como preços de ações, commodities e taxas de conversão de moedas) devem ser tratados como curvas com um único vértice;

- Jun/2008 (três semanas depois) – a terceira tarefa é entregue aos alunos. Eles esperam uma quarta tarefa que seja uma simples extensão da terceira, muito provavelmente uma nova estratégia para visualizar a informação que eles já sabem ler e manipular através das estruturas de classes criadas nas tarefas 1 e 2. Entretanto, eles recebem o instrumento regulatório oficial liberado em Maio, em seu formato original, e que deverão atualizar a ferramenta de forma a que ela continue útil. Eles têm duas semanas para entregar a nova versão da ferramenta;
- Jun/2008 (duas semanas depois): em final de período, os três grupos se esforçam para entregar a última tarefa do trabalho. O grupo B recebe a melhor nota, enquanto os grupos A e C ficam praticamente empatados na avaliação. Como resultado do curso, temos uma ferramenta completa e que pode ser realmente interessante para a agência regulamentadora. No entanto, o principal resultado não é tangível e está representado na experiência vivida pelos alunos (e pelo professor) na condução da disciplina.

5 Lições Aprendidas e Conclusões

A expectativa acerca da disciplina era que os alunos veriam o valor das discussões teóricas sobre organização do código e princípios de projetos no desenvolvimento do projeto proposto. Na quarta tarefa, quando precisariam rever o código escrito meses atrás, deveriam apreciar os comentários escritos na época e o esforço extra para tornar o código legível. Esperávamos que compreendessem o valor de prever mudanças futuras e dividir o sistema em componentes que limitem os efeitos colaterais das mudanças sofridas pelo software. Nestes aspectos, como pudemos observar em conversas com os alunos ao final do curso, não nos frustramos. No entanto, conduzir uma experiência de projeto como a descrita neste artigo não é uma tarefa fácil. A seguir, são discutidas as dificuldades encontradas:

- É difícil encontrar problemas reais, como a implementação da plataforma Basiléia II. Problemas reais são mais complexos do que o razoável para uma disciplina de curta duração. Por outro lado, uma situação similar pode ser simulada ou o problema complexo pode ser particionado no tamanho certo para as necessidades e limitações da disciplina;
- Uma vez que o mundo real está sendo tratado, o cenário que influencia as tarefas propostas aos alunos não pode ser totalmente controlado pelo professor: não há garantia de que questões relevantes possam aparecer. Além disso, eventos podem ocorrer em uma direção diferente do que a esperada. Esta última situação é mais fácil de ser tratada, com o professor atuando como um filtro para as novidades vindas do ambiente externo. A primeira situação não pode ser prevista ou controlada e, portanto, o professor pode precisar simular a ocorrência de eventos relevantes. Isto pode afetar negativamente a motivação, uma vez que a proximidade com o mundo real, e com eventos que realmente aconteceram, gera motivação nos alunos;
- É bastante útil trabalhar com entregas parciais, a serem analisadas pelo professor e detalhadas em implementações de referência para as tarefas na sequência. A avaliação de cada fase permite aos alunos aprender a partir de seus erros e obter

melhores resultados nas entregas subseqüentes. A avaliação compartilhada permite que os alunos aprendam com os erros cometidos pelos demais. Finalmente, a implementação de referência contribui para manter a competição, uma vez que os alunos partem do mesmo ponto a cada nova entrega;

- O esforço de organizar o projeto de curso é muito grande. Deve ser contabilizado tempo para: selecionar um projeto apropriado, avaliar as entregas enviadas pelos alunos (uma vez que elas serão a base para tarefas futuras, elas precisam ser corrigidas pelo professor o mais rápido possível), aprimorar a melhor solução em uma implementação de referência, acompanhar e filtrar os eventos vindos do ambiente externo. Estas atividades se somam à preparação de aulas, preparação e correção de provas (no caso desta disciplina, uma única prova), preparação de exercícios, etc. O professor deve estar preparado para dispender mais tempo durante a execução da disciplina e para "colocar as mãos na massa" para preparar as implementações de referência, mesmo em um cenário em que nenhum dos grupos pôde entregar uma solução para uma dada tarefa;
- Apesar do custo de preparar e executar a disciplina, a reutilização do material elaborado e das tarefas sugeridas para uma nova oferta no futuro é restrita. Uma vez que o projeto real é temporário (deveria ser, pelo menos), os eventos e notícias relacionados a ele supostamente terminam dentro de um período de tempo. Então, após este período, o projeto (junto com seu material relacionado) será parte do passado. Desta forma, perderá algumas de suas relações com o mundo real, apesar de permanecer um problema concreto a ser tratado pela indústria: será uma tarefa melhor do que um projeto abstrato e genérico.

Agradecimentos: agradecemos aos alunos que participaram deste estudo de caso – Daniel Serrano, Fernando Netto, João Gonçalves, Leonardo Jobim, Martin Pereira e Wallace Ugulino.

Referências

BANCO CENTRAL DO BRASIL. **Comunicado nº 12.746 .** Disponível através de busca por número de documento em: https://www3.bcb.gov.br/normativo/pesquisar.paint?method=pesquisar Acesso em maio de 2008.

COMITÊ DA BASILÉIA PARA SUPERVISÃO BANCÁRIA. Disponível em: http://www.bis.org Acesso em maio de 2008.

LEHMAN, M. Laws of Software Evolution Revisited. Lecture Notes in ComputerScience, Springer, Berlin/Heidelberg, v. 1149, p. 108 – 124, 1996.

Mc CONNEL, STEVEN. **Code Complete: A Practical Handbook of Software Construction**. Microsoft Press, Redmond, WA, 1993. 960 p.

PFLEEGER, Shari L. **Software Engineering: Theory and Practice**. Prentice-Hall Inc. Upper Saddle River, NJ, 1998. 659 p.

Engenharia de Software no Curso de Ciência da Computação da UERJ

Vera Maria B. Werneck; Rosa Maria E. M. da Costa; Maria Clicia Stelling de Castro; Alexandre Sztajnberg; Paulo Eustáquio D. Pinto; Roseli S.Wedemann
Departamento de Informática e Ciência da Computação
Instituto de Matemática e Estatística - IME
Universidade do Estado do Rio de Janeiro– RJ – Brazil
{vera, rcosta, clicia, alexszt, pauloedp, roseli}@ime.uerj.br

Abstract. This paper describes the new curriculum of the Computer Science Program at the Universidade do Estado do Rio de Janeiro (UERJ) and presents the Software Engineering area and its associated disciplines. This new curriculum considered different perspectives: the academic experience acquired in the previous curriculum, the proposal done by the Sociedade Brasileira de Computação (SBC-Brazilian Computer Science), the requirements of the MEC (Brazilian Education Ministry) and the market demands. The curriculum is composed by a obliged basic nucleus of disciplines complemented by Basic, Practical and Technical Electives disciplines. To demonstrate the methodology adopted in the construction of this new curriculum, we consider in this work the Software Engineering area. This area is present in the basic nucleus of the course (Software Engineering, Analysis and Project of Systems and Human-Computer Interfaces) and in a group of optional Technical Electives disciplines (Requirements Engineering, Software Quality, Systems Modeling, Management of Information Technology and Special Topics). This structure allows the student to have a general vision of the area and a more specific education, aiming at the diverse activities of project, maintenance, quality evaluation and software development.

Keywords: Software Engineering, Computer Science Curriculum.

Resumo. Este artigo descreve o novo currículo de Ciência da Computação da Universidade do Estado do Rio de Janeiro (UERJ) apresentando o núcleo de Engenharia de Software e suas disciplinas associadas. O novo currículo considerou diferentes perspectivas: a experiência acadêmica obtida no currículo anterior, a proposta da Sociedade Brasileira de Computação (SBC), os requisitos do MEC e as demandas do mercado. Este currículo é composto por um núcleo básico de disciplinas Obrigatórias além de disciplinas Eletivas Básicas, Técnicas e Práticas. Para demonstrar a metodologia adotada na construção deste novo currículo, neste trabalho consideramos a área de Engenharia de Software (ES). Esta área está contemplada no núcleo básico do curso (Engenharia de Software, Análise e Projeto de Sistemas e Interfaces Humano-Computador) e num grupo de Eletivas Técnicas optativas (Engenharia de Requisitos de Software, Qualidade de Software, Modelagem de Sistemas e Gestão de Tecnologia de Informação e Tópicos Especiais). Esta estruturação permite ao aluno ter uma visão geral da área e uma formação mais específica, visando às diversas atividades de projeto, manutenção, avaliação da qualidade e suporte ao desenvolvimento de software.

Palavras-chave: *Engenharia de Software, Currículo de Engenharia de Software.*

1 Introdução

Nos anos 80, na Universidade do Estado do Rio de Janeiro -UERJ foi criado o curso de Matemática, Modalidade Informática. A estratégia para formação do corpo docente deste curso foi buscar profissionais que atuavam no mercado desta área, uma vez que, uma abordagem mais prática era requisito essencial em relação aos modelos explorados no cenário nacional. No decorrer dos anos, o curso foi desvinculado do curso de matemática, sendo então denominado Bacharelado em Informática. Após terem sido implementadas algumas modificações pontuais, as rápidas e constantes mudanças tecnológicas mais atuais, criaram novas demandas, exigindo uma reformulação mais profunda no currículo. Um estudo minucioso foi iniciado para realizar uma reforma curricular, que deveria atender às recomendações oficiais vigentes, sendo enquadrada no contexto de um Bacharelado em Ciência da Computação [Diretrizes, 2008].

No ano de 2007, após várias reuniões, formação de comissões para coordenar as discussões e terem sido realizados todos os trâmites exigidos pela universidade, o novo currículo foi aprovado. No primeiro semestre de 2008, este currículo foi implantado, e no segundo semestre foi iniciada a migração dos discentes antigos (aproximadamente 650 alunos).

A proposta do novo curso inclui uma formação sólida com base matemática e um estudo abrangente e consistente das diversas áreas de Ciência da Computação [Reforma, 2008]. A proposta segue as recomendações da SBC [SBC, 2008], além de contemplar as disciplinas exigidas pelo MEC [Diretrizes, 2008], [MEC,2008], [SESu, 2008]. Outros pontos importantes foram observados, como os conceitos que têm sido abordados no ENADE [Enade, 2008] e no POSCOMP [SBC, 2008]. Estas duas avaliações são mecanismos orientadores importantes e considerados por diversas instituições brasileiras, com reconhecida excelência. A partir da integração de todos esses requisitos, o curso foi composto por um núcleo básico de disciplinas obrigatórias abrangendo Matemática, Física e Ciência da Computação, além de disciplinas Eletivas Básicas, Técnicas e Práticas.

As disciplinas do núcleo básico permitem uma formação gradual e consistente, preparando o discente para atuar na área comercial, acadêmica ou científica. As disciplinas pertencentes ao grupo de Eletivas Básicas fazem parte da formação de consolidação do conhecimento. Elas permitem o aprofundamento dos conceitos abordados no núcleo básico. As disciplinas pertencentes ao grupo de Eletivas Técnicas permitem ao discente direcionar seus estudos e conhecimentos para áreas específicas, como, por exemplo, Engenharia de Software, Computação Científica, Sistemas Distribuídos, Redes, Algoritmos entre outras. Essas disciplinas têm uma forte relação com as especialidades do nosso Corpo Docente[1]. É aconselhado que o discente escolha uma determinada linha de estudos, de acordo com o seu perfil.

O objetivo deste trabalho é descrever a área de ES em particular, como nosso exemplo de construção do conhecimento. Além disso, mostrar como estão distribuídas as disciplinas desta área e as estratégias utilizadas. Ressaltamos que a mesma estrutura foi aplicada nas demais áreas específicas oferecidas no novo currículo.

Este trabalho está organizado da seguinte forma: na Seção 2 apresentamos uma visão do currículo novo. A Seção 3 aborda as diferenças entre o currículo anterior e a

1 Atualmente nosso Corpo Docente tem um perfil mais acadêmico com professores doutores e mestres de 40 horas, além de continuar tendo alguns professores de 20 horas que são profissionais do mercado.

nova proposta. Na Seção 4 concluímos o artigo discutindo alguns aspectos que consideramos fundamentais para a formação na área de Engenharia de Software.

2 Núcleo Básico do Currículo de Ciência da Computação

A Figura 1 apresenta a grade do novo currículo de Ciência da Computação da UERJ [Reforma, 2008]. A contribuição deste currículo é oferecer um embasamento teórico e prático na área de Ciência da Computação e introduzir algumas disciplinas sem um pré-requisito explícito. Por exemplo, para a disciplina de Engenharia de Software o aluno poderá cursá-la se tiver concluído pelo menos 30% dos créditos ou 58 créditos. Esta estratégia permite diminuir a necessidade de pré-requisitos específicos.

Outro aspecto importante é a introdução de duas disciplinas da área de Letras: Português Instrumental e Inglês Instrumental. Estas disciplinas são justificadas devido a observação das dificuldades que os discentes apresentam na redação e leitura de documentos.

As disciplinas Eletivas deste currículo são divididas em Eletivas Básicas e Eletivas que podem ser Técnicas, Práticas e Complementares. A escolha das disciplinas Eletivas Básicas (no mínimo uma, entre 7 opções) e Eletivas Técnicas, Práticas e Complementares (no mínimo quatro disciplinas) deve ser baseada no perfil do discente e de acordo com a área que deseja complementar. As disciplinas Eletivas Técnicas direcionam o currículo para conhecimentos em áreas específicas, como Engenharia de Software. Além disso, existem as Eletivas Práticas que permitem a simulação e experimentação em laboratórios. Tópicos Especiais (Eletivas Complementares) é uma disciplina de ementa aberta, relacionada ao estado da arte em algum domínio da Ciência da Computação. No currículo anterior, essa disciplina já era ministrada. Essa disciplina permitiu o contato mais próximo dos alunos de graduação com experiências em áreas de pesquisa.

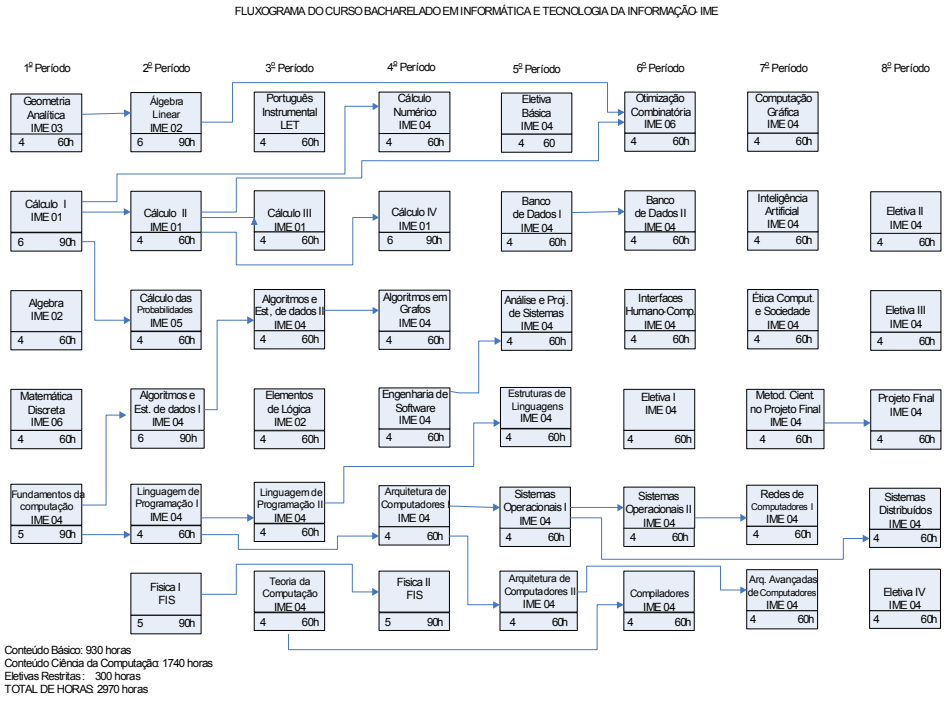


Figura 1. Grade do novo currículo de Ciência da Computação da UERJ [2].

O discente deve ainda cursar a disciplina de Projeto Final, onde ele deve preparar uma monografia sobre algum tópico relevante na área de Ciência da Computação. A

inovação em relação ao currículo anterior foi criar a disciplina de Metodologia de Pesquisa, que introduz conhecimentos sobre o método científico e auxilia a elaboração do anteprojeto da monografia final.

3 A Área de Engenharia de Software

No currículo anterior, a ênfase das disciplinas da área de ES estava nas áreas de Análise e Projeto de Sistemas. Quatro dessas disciplinas abordavam a modelagem, uma a Gerência de Informação, outra os Sistemas de Informação, além de Organização e Métodos.

Neste novo currículo, a área de ES está formalmente representada em três disciplinas obrigatórias: Engenharia de Software, Análise e Projeto de Sistemas e Interfaces Humano-Computador. Entretanto, os conceitos de avaliação de software, testes, documentação de código, reutilização, orientação a objetos são também abordados nas duas disciplinas de Linguagem de Programação I (linguagem C) e II (linguagem JAVA). A idéia é, desde o início do curso, abordar esses conceitos, fornecendo a base do conhecimento necessário não só para solucionar um problema, mas também para construir programas de acordo com técnicas atuais, que permitem obter produtos de qualidade, documentados e reutilizáveis. As disciplinas fundamentais da área estão descritas a seguir.

A disciplina de ES apresenta os conceitos básicos sobre processo de desenvolvimento de software, métodos, técnicas e ferramentas de construção, qualidade e gerência de desenvolvimento e manutenção de software. A disciplina de Análise e Projeto de Sistemas tem como objetivo capacitar os alunos para analisar e projetar sistemas com qualidade, utilizando técnicas orientadas a objetos, como a linguagem padrão UML (Unified Modeling Language) e o modelo de processo unificado RUP (Rational Unified Process). O curso de Interfaces Humano-Computador (IHC) oferece uma noção geral da área, bases teóricas, modelos e métodos para projeto e avaliação de interfaces, principalmente em relação ao objetivo, custo e benefício da aplicação. Acreditamos que, dessa forma, capacitamos os discentes para selecionar metodologias, técnicas e ferramentas mais adequadas aos seus objetivos.

As disciplinas Eletivas Técnicas da área de ES são: Engenharia de Requisitos de Software, Qualidade de Software, Modelagem de Sistemas e Gestão de Tecnologia de Informação. A disciplina de Engenharia de Requisitos apresenta os conceitos básicos e o processo de definição dos requisitos de software, assim como métodos, técnicas e ferramentas de gerenciamento de requisitos. Em Qualidade de Software são abordados conceitos sobre a qualidade em relação ao produto e ao processo de desenvolvimento, certificações e normas vigentes na área e ferramentas de qualidade de software. Na disciplina Modelagem de Sistemas é oferecido uma visão de Modelagem Funcional de Sistemas, como as seguintes modelagens: de Negócios; Orientada a Objetivos; e Orientada a Agentes, com estudos de casos. Em Gestão de Tecnologia de Informação são discutidas as metodologias, técnicas e ferramentas do âmbito de gestão. Esses tópicos abordados permitem ao aluno evoluir no conhecimento dos principais aspectos da gestão nas organizações, entendendo o alinhamento da Tecnologia de Informação com os objetivos e metas das empresas.

A área de ES, também, pode oferecer disciplinas práticas cuja idéia é permitir a simulação e experimentação em laboratórios de técnicas e ferramentas de ES. Neste caso, temos possibilidade de utilizar ferramentas do mercado ou realizar processos de experimentação e avaliação de exemplos próximos aos reais. A criação de Tópicos Especiais em ES está prevista para atender e oferecer uma visão de uma área específica da área de ES, como por Metodologias Orientadas a Objetivos e de ES para Sistema

Multi-Agentes. Estas disciplinas já foram oferecidas no currículo antigo e permitiram um contato mais próximo dos alunos de graduação com esses temas e experiências em áreas de pesquisa. Dessa forma, obtivemos resultados significativos, onde os alunos que fizeram projeto final na área geraram artigos em Workshops e Conferências, além de terem despertado o interesse por temas avançados, que por sua vez, poderão ser complementados em programas de mestrado e doutorado.

Além disso, temos o programa de Bolsas de Iniciação Científica, o programa de Estágio Complementar e a possibilidade de realização de intercâmbio com a York University, em Toronto. O projeto de cooperação estabelecido entre a UERJ e a York University tem possibilitado a troca de professores para lecionar disciplinas associadas a ES nos cursos de graduação dessas instituições. Atualmente, temos alunos que estão iniciando seu intercâmbio neste semestre (2008/2).

4 Conclusões

Este artigo descreveu o novo currículo de Ciência da Computação que está em fase de implantação na UERJ, destacando a área de Engenharia de Software. O objetivo desta proposta é dar uma formação sólida em conceitos básicos da Ciência da Computação. Em particular, destacamos através da disciplina de Engenharia de Software, a integração das áreas e temas específicos, de acordo com o perfil desejado pelo aluno.

No segundo semestre de 2008, a primeira turma com a abordagem descrita, está concomitante com o grupo de alunos antigos que optarem pelo novo currículo. Assim, oferecemos pela primeira vez a disciplina de Engenharia de Software e Linguagem de Programação I com a integração de conceitos. Anualmente, pretendemos realizar avaliações periódicas, tanto com os alunos, quanto com os docentes para que sejam coletadas informações, através de questionários, que proporcionem possíveis ajustes nas estratégias até então desenvolvidas. Dessa forma, no final do próximo ano, em 2009, poderemos ter uma visão mais ampla do impacto dessa nova proposta, já que teremos oferecido as três disciplinas básicas da área de ES.

Apesar dos egressos do curso, em geral, se colocarem muito bem no mercado de trabalho, sendo aceitos em diversas instituições que oferecem mestrado e doutorado, reconhecemos que o processo educacional na área exige um longo processo de amadurecimento. Esperamos que a coleta periódica de informações sobre o curso ofereça indicadores para apoiar contínuas modificações curriculares, visando atender as metodologias de ensino-aprendizagem e as demandas da evolução tecnológica.

Referências

Diretrizes Curriculares de Cursos da Área de Computação e Informática. Disponível em ftp://ftp.inf.ufrgs.br/pub/mec/diretrizes.doc. Acesso em: ago 2008.

ENADE, "Exame Nacional de Desempenho de Estudantes". Disponível em: http://www.inep.gov.br/superior/enade/. Acesso em ago 2008.

MEC/SESu, "Comissão de Especialistas de Ensino de Computação e Informática". Disponível em http://www.inf.ufrgs.br/mec. Acesso em: ago 2008

Reforma Curricular do Bacharelado em Informática e Tecnologia da Informação. Departamento de Informática e Ciência da Computação,Universidade do Estado do Rio de Janeiro. Disponível em http://www.ime.uerj.br/graduacao/tecdainformacao.html. Acesso em: ago 2008.

SBC, "Currículo de Referência para Cursos de CC, EC e SI". Disponível em: http://www.sbc.org.br/index.php?language=1&subject=28&content=downloads&id=82. Acesso em: ago 2008.

SESu, "Diretrizes Curriculares para os Cursos de Graduação". Disponível em : http://portal.mec.gov.br/sesu/index.php?option=content&task=view&id=430&Itemid=420. Acesso em: ago 2008.

Bacharelado em Engenharia de Software na Universidade Federal de Goiás

Fábio Nogueira de Lucena[1] Juliano Lopes de Oliveira[1]
Auri Marcelo Rizzo Vincenzi[1*]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)

{fabio,juliano,auri}@inf.ufg.br

Abstract. This paper describes the pedagogical project of the Software Engineering Bachelor´s course, a recently created undergraduate course at Universidade Federal de Goiás (UFG). The aim of this new course is to prepare professionals to be able to conduct effective software enterprises, competing in the global market. The elaboration of this project was based on the 25 years of experience on coordinating undergraduate courses in Computer Science at UFG. The course structure reflects successful experiences on the integration of educational practices of different disciplines to perform large scale projects, which provide the main context for the teaching-learning process.

Keywords: Graduating course, Software Engineering, pedagogical project.

Resumo. Este texto descreve o projeto pedagógico do curso de Bacharelado em Engenharia de Software, recém criado na Universidade Federal de Goiás (UFG). A missão deste novo curso é formar profissionais aptos a conduzirem efetivamente empreendimentos de software que sejam competitivos dentro do mercado global. O projeto pedagógico do curso foi elaborado com base na experiência de 25 anos do Instituto de Informática da UFG na coordenação de cursos de graduação na área de Computação. A estrutura do curso reflete experiências bem-sucedidas de integração das práticas educacionais de diferentes disciplinas para construir projetos de maior envergadura que formam o contexto do processo ensino-aprendizado.

Palavras-chave: Curso de graduação, Engenharia de Software, projeto pedagógico.

*Trabalho patrocinado com apoio do CNPq (309503/2006-0).

1 Introdução

Este texto apresenta os principais conceitos do projeto pedagógico do curso de Bacharelado em Engenharia de Software (ES) da UFG, criado para atender esta demanda por Engenheiros de Software capacitados. Espera-se que os profissionais formados neste curso sejam capazes de atender as necessidades da nossa indústria de software, contribuindo para alavancar a participação brasileira no mercado internacional de software.

Devido à restrição de espaço, diversos pontos referentes ao projeto pedagógico do curso tiveram que ser omitidos. Para mais informações, pode ser consultado o sítio do curso http://engenhariadesoftware.inf.br/ [Lucena, 2008]. Na Seção 2 são apresentadas as bases do projeto pedagógico. Na Seção 3 são descritas as características do curso. A Seção 4 apresenta o particionamento e a perspectiva das disciplinas. A Seção 5 discute a redefinição das disciplinas utilizando o conceito de projeto e, finalmente, na Seção 6 são apresentadas as conclusões e perspectivas do curso.

2 Bases do Projeto Pedagógico

Tomando como ponto de partida as diretrizes definidas em [ACM e IEEE, 2004], a elaboração do projeto pedagógico do curso de Bacharelado em ES da UFG buscou alinhar os aspectos técnicos da formação do perfil de Engenheiro de Software com as normas estabelecidas pelo Ministério de Educação e Cultura (MEC) e também pela UFG, no que concerne a cursos de bacharelado. Adicionalmente, as recomendações da Sociedade Brasileira de Computação [SBC, 1999] também foram consideradas na especificação detalhada do currículo do curso.

Apesar de se apoiar em diretrizes e normas reconhecidas nacional e internacionalmente, o projeto pedagógico do curso está fortemente embasado na experiência acumulada pelo Instituto de Informática da UFG ao longo de 25 anos na coordenação de cursos na área de Computação. Durante este período, a abordagem orientada a projetos tem sido experimentada com sucesso em disciplinas da área de ES dos cursos de graduação em Ciência da Computação (CC) e Engenharia da Computação (EC), nos cursos de especialização em ES, e nas disciplinas da área de SI do mestrado em Computação.

Em particular, a experiência do bacharelado em CC merece destaque. O curso possui várias disciplinas na área de ES sendo que a realização de projetos que envolvem conhecimentos das várias disciplinas é feita dentro da disciplina Construção de Software, cuja carga-horária é exclusivamente prática, ou seja, é um espaço para a aplicação de todos os conceitos vistos nas disciplinas da área. Os resultados dessa prática são bastante satisfatórios. Uma demonstração disto é o conceito máximo obtido pelos estudantes do curso de CC da UFG no último Exame Nacional de Desempenho de Estudantes na área de CC [INEP, 2006].

3 Características do Curso

O curso tem duração de quatro anos e foi projetado para viabilizar a inserção do estudante no mercado de trabalho durante o estudo de graduação. A carga horária semanal exige a dedicação do estudante no período noturno e vespertino durante a semana, e nos períodos matutino e vespertino aos sábados. Com isso, a expectativa do

curso é formar um profissional preparado para uma sólida carreira na indústria de software. Especificamente, o bacharel em ES (engenheiro de software) deverá ser capaz de (i) especificar requisitos de software; (ii) projetar (arquitetura e design detalhado de) software; (iii) construir (programar) software com qualidade; e (iv) coordenar projetos de manutenção (corretiva, adaptativa e evolutiva) ou desenvolvimento de software.

O software considerado nos itens anteriores deve ter características reais, ou seja, deve ser capaz de solucionar problemas complexos, exigindo o trabalho de uma equipe de engenheiros de software ao longo de vários meses, e as disciplinas do curso devem propiciar tal ambiente, considerando os seguintes princípios: 1) o conhecimento é indissociável da prática correspondente; 2) estabelecer um ambiente receptivo à criatividade e inovações; 3) a aplicação do conhecimento é melhor que o conhecimento em si; e 4) trabalhar com o contexto de uma fábrica de software.

4 Particionamento e Perspectivas das Disciplinas

O núcleo epistemológico do curso classifica as disciplinas em três grandes fases: ambientação, fundamentação e maturidade (Figura 1). Essas fases determinam o enfoque da disciplina e orientam a sua regência, estabelecendo diferentes objetivos para cada disciplina ofertada no curso. Como pode ser observado, há superposições para qualquer assunto abordado no curso. Contudo, em cada instante de tempo e para um determinado assunto, a seqüência recomendada pela estrutura curricular do curso segue a ordem: ambientação, fundamentação e finalmente a maturidade.

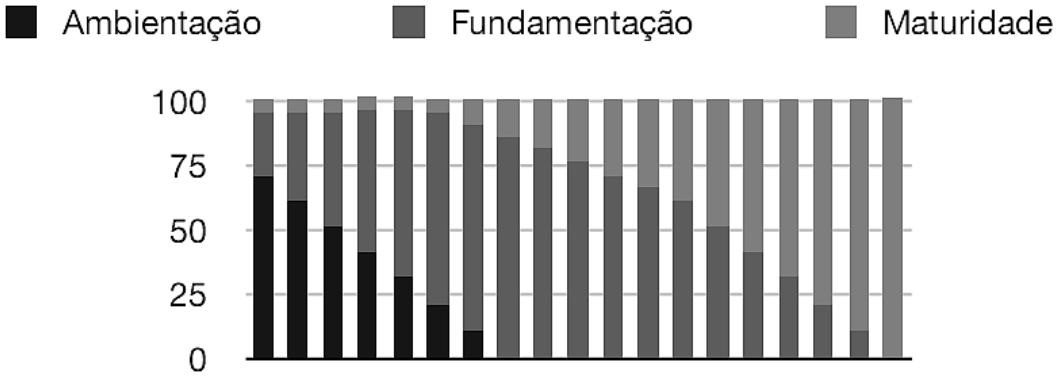


Figura 1: Distribuição do Esforço ao longo do tempo nas Fases do Curso.

As atenções do curso são inicialmente dedicadas à ambientação e, paulatinamente, substituídas por aquelas da fundamentação e, finalmente, pela maturidade. Isto se faz claro pelo papel desempenhado por professores e estudantes, e pelo conteúdo e complexidade dos problemas abordados nas disciplinas em cada fase. Uma de interpretar tais fases é associar à ambientação a formação do estudante, à fundamentação um programa de *trainee* para uma fábrica de software e, finalmente, à maturidade a capacidade de aplicar o conhecimento de forma independente.

- ***Ambientação*** – A fase inicial de ambientação apresenta ao estudante recém-chegado ao curso o contexto da ES. Nesta fase inicial do curso o modelo de ensino "tradicional" é empregado.
- ***Fundamentação*** – Na fase de fundamentação a visão horizontal e panorâmica é substituída por uma perspectiva mais focada e vertical dos conceitos. O modelo de ensino expositivo perde espaço, ocorrendo uma transição para um modelo no qual o estudante é o centro do processo de aprendizado, embora sua participação na resolução de problemas seja acompanhada de perto pelos docentes.

- ***Maturidade*** – O conhecimento adquirido na ambientação, e a experiência vivenciada na fundamentação, na fase de maturidade passam a ser exercitados em cenários mais complexos e elaborados. Assim, a fase de maturidade busca exercitar o conhecimento para que a formação do egresso se fortaleça.

5 Redefinição de Disciplinas: o conceito de Projeto

Cada disciplina do curso define um subconjunto do conhecimento da ES. Uma disciplina possui carga horária, método de avaliação, e um professor responsável. A proposta do curso não altera o papel administrativo de uma disciplina, porém esta deixa de ser o elemento principal, atômico e isolado, da formação do estudante. O principal fator de formação no curso de ES será participação do estudante em projetos reais. Ao longo de um projeto, vários conhecimentos (ou disciplinas) são exercitados, criando oportunidades de aprendizado e, conseqüentemente, de avaliação. Por exemplo, o estudante passa a ser avaliado, ao longo do curso, pelo desempenho no exercício do conhecimento atribuído a cada disciplina em atividades de um ou mais projetos.

Os docentes, a coordenação do curso e os estudantes desempenharão atividades dirigidas por projetos. Em vez de simplesmente cursar disciplinas, os estudantes são membros de um ou mais projetos ao longo do curso. As atividades nos projetos devidamente monitoradas e controladas servem como base para a avaliação do estudante.

A execução desta disciplina pode ocorrer em uma ou mais edições, conforme a matriz curricular. Em todas as edições a ementa é a mesma, mas o conteúdo não necessariamente. As edições são o mecanismo empregado para a transição do modelo tradicional de ensino para aquele definido como de maturidade. Neste sentido, é provável que a primeira edição de uma disciplina seja próxima do modelo tradicional. A segunda, contudo, já trará mudanças, assim como as posteriores. O objetivo não é simplesmente "fragmentar" a ementa em diversas edições, nem repetir várias vezes o mesmo conteúdo, mas revê-la várias vezes, embora com perspectivas distintas, em geral, considerando projetos diferentes.

Convém ressaltar que o conjunto das edições de uma disciplina é mais relevante que cada edição em si. Não necessariamente um mesmo professor irá ministrar todas as edições de uma disciplina e, desta forma, um mesmo assunto poderá ser abordado de formas diferentes. Em qualquer caso, contudo, uma edição deve estar, em relação à edição anterior, em nível mais próximo daquele estabelecido para o curso.

6 Conclusões

Este texto descreveu os principais aspectos do projeto pedagógico de um curso de bacharelado em ES, recém criado na UFG, sob a coordenação do Instituto de Informática. Da perspectiva do escopo de ES abordado, o curso segue as orientações do núcleo epistemológico proposto pela ACM e pela IEEE [ACM e IEEE, 2005], contemplando todo o corpo de conhecimento em ES [SWEBOK, 2004].

Da perspectiva didática, o curso oferece extensa oportunidade de prática, por meio de uma fábrica de software, na qual projetos realistas são executados com a participação dos estudantes e com o acompanhamento do corpo docente do curso.

Assim, a formação do aluno é consolidada pela experiência prática, e há oportunidade para o estudante se especializar, conforme as suas atribuições em projetos.

A proposta pedagógica do curso segue a mesma abordagem utilizada em cursos similares existentes em diversos países, tais como Estados Unidos, Austrália, Inglaterra, e Canadá. No entanto, no Brasil, até onde os autores conhecem, não existe um curso de graduação estabelecido voltado especificamente para a formação de Engenheiros de Software. Assim, o curso aqui discutido abre uma nova possibilidade para o atendimento das necessidades estratégicas do Brasil em termos de ES.

Embora o corpo de conhecimento abordado seja o mesmo adotado em cursos semelhantes ofertados no exterior, o processo utilizado para o estudo deste corpo de conhecimento representa uma inovação importante. A abordagem clássica, que estabelece a noção das disciplinas como o principal instrumento do processo de ensino-aprendizado, é substituída na proposta do curso por uma abordagem baseada na resolução de problemas, e focada no conceito de projeto.

A abordagem baseada em problemas e, particularmente, o uso de projetos como instrumento principal das atividades de ensino-aprendizado tem sido empregada com sucesso nas disciplinas da área de ES em diversos cursos coordenados pelo Instituto de Informática da UFG, tanto na graduação em CC quanto em cursos de pós-graduação (*lato* e *stricto sensu*). A expectativa é que esses resultados positivos sejam potencializados no curso de bacharelado em ES, uma vez que as principais atividades do Engenheiro de Software são realizadas no contexto de projetos de software.

Referências

ACM; IEEE. The Joint Task Force on Computing Curricula – IEEE and ACM (2004) "Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering". In **The Computing Curricula Series**. Disponível em: http://sites.computer.org/ccse/SE2004Volume.pdf. Acesso em: 23 ago. 2004.

ACM; IEEE. The Joint Task Force on Computing Curricula – IEEE and ACM (2005) "Computing Curricula 2005 – The Overview Report covering undergraduate degree programs in Computer Engineering, Computer Science, Information Systems, Information Technology, and Software Engineering". In **The Computing Curricula Series.** Disponível em: http://www.computer.org/portal/cms_docs_ieeecs/ieeecs/education/cc2001/CC2005-March06Final.pdf. Acesso em: 30 set. 2005.

INEP. **Resultados do Exame Nacional de Desempenho de Estudantes.** Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira – INEP. Disponível em: http://enade2006.inep.gov.br/novo/Site/?c=CUniversidade&m=mostrar_lista_area. Acesso em: 21 set. 2006.

LUCENA, Fabio N. **Projeto Pedagógico do Curso Bacharelado em Engenharia de Software – Universidade Federal de Goiás – UFG**. Disponível em: http://engenhariadesoftware.inf.br/. Acesso em: 21 set. 2008.

SBC. **Currículo de Referência da SBC para Cursos de Graduação em Computação e Informática**. Sociedade Brasileira de Computação – SBC. Disponível em: http://www.sbc.org.br. Acesso em: 21 set. 1999.

SWEBOK. **The Guide to the Software Engineering Body of Knowledge**, IEEE Computer Society. Disponível em: http://www.swebok.org/. Acesso em: 21 set. 2004.

Novas tendências em educação de Engenharia de Software: um estudo de caso no domínio de Teste de Software

Marco Aurélio Graciotto Silva[1] Vanessa Araujo Borges[1]
Ellen Francine Barbosa [1] José Carlos Maldonado[1]

[1] Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo (USP)
magsilva@icmc.usp.br, va.borges@icmc.usp.br,
francine@icmc.usp.br, jcmaldon@icmc.usp.br

Abstract. Information dissemination, allied with new technologies, has changed the educational scenario, strengthening concepts that led to the building of new cognitive environments and improved the teaching and learning process. In this context, new educational perspective should be explored in the following years: distance learning, digital television, collaboration, and open content development. What are we doing to build this future? Researches related to educational content modeling and engineering process have shown the benefits of the systematic production of educational modules. Evidences, gathered at Software Testing undergraduation courses, have shown improvement in learners' performance. The upcoming digital television foster the specialization and creation of new methods, enacting the investigation of education under a new perspective, in the building of interactive, flexible, and reusable content, and leveraging the quality of educational modules.

Keywords: education, software engineering, educational modules, quality, digital television.

Resumo. A disseminação de informação, aliada ao uso de novas tecnologias, tem alterado de forma significativa o cenário educacional. Assim, vivencia-se o fortalecimento de conceitos que favorecem a construção de novos ambientes cognitivos, capazes de contribuir para a melhoria da qualidade no processo de ensino e aprendizado. Dentro desse contexto, novas tendências e perspectivas de educação deverão ser exploradas nos próximos anos: educação à distância, televisão digital, colaboração e desenvolvimento de conteúdos abertos. O que estamos fazendo para construir este futuro? Pesquisas relacionadas à modelagem e ao processo de engenharia de conteúdos educacionais demonstram os benefícios da sistematização da produção de módulos educacionais. Aplicados em cursos de graduação, no domínio de Teste de Software, a qualidade dos módulos é evidenciada pelo melhor desempenho dos alunos. O advento da televisão digital instiga a especialização e criação de novos métodos, possibilitando explorar o ensino sob uma nova perspectiva, na construção de conteúdos interativos, flexíveis e reutilizáveis, e propiciando um novo incremento na qualidade dos módulos educacionais.

Palavras-chave: educação, engenharia de software, módulos educacionais, qualidade, televisão digital.

1 Introdução

A educação em Engenharia de Software, ao longo dos anos, tem incorporado novas abordagens e técnicas visando à melhoria na qualidade: integração entre disciplinas, jogos, projetos reais de software, entre outras. Na maioria dos casos, entretanto, essas experiências restringem-se ao ambiente universitário e na modalidade presencial. O universo de interessados que não possuem acesso aos centros de excelência em Engenharia de Software e, conseqüentemente, aos conhecimentos essenciais e de contínuo aperfeiçoamento na disciplina, encontra-se sem muitas alternativas.

Importantes iniciativas têm sido desenvolvidas para sanar essa deficiência. A disponibilização de material didático na Internet é uma prática comum e se observa a crescente adoção de sistemas de gerenciamento de aprendizado. Cursos de graduação e de pós-graduação *lato sensu* à distância estão sendo criados, tanto pela iniciativa pública quanto privada. Com a recente introdução do SBTVD (Sistema Brasileiro de Televisão Digital), vê-se uma nova oportunidade para alavancar a educação à distância no país, usufruindo-se da maior disseminação da televisão e das características de interatividade e mobilidade acrescidas pela digitalização.

Uma questão a ser resolvida é o desenvolvimento de conteúdos de qualidade para a televisão digital. O processo de produção deve contemplar questões de interatividade, comunicação, computação e pedagogia, que demandam equipes multidisciplinares. Os produtos gerados devem possuir atributos mínimos de qualidade e facilidade para adaptação aos diferentes tipos de aprendizes que o utilizarão.

No âmbito do ICMC (Instituto de Ciências Matemáticas e de Computação - Universidade de São Paulo), desenvolveu-se o Processo Padrão para o Desenvolvimento de Módulos Educacionais [Barbosa 2004, Barbosa e Maldonado 2006c], e uma abordagem para a modelagem dos conteúdos educacionais associados – a AIM-CID (Abordagem Integrada de Modelagem – Conceitual, Instrucional e Didática) [Barbosa 2004, Barbosa e Maldonado 2006b]. Módulos educacionais para Teste de Software [Barbosa e Maldonado 2006a] e ensino integrado de Fundamentos de Programação e Teste de Software [Barbosa et al. 2008a] foram gerados a partir dessas pesquisas e têm sido utilizados em disciplinas na área [Barbosa et al. 2008b]. A especialização do Processo Padrão e da AIM-CID para a geração de módulos educacionais para o SBTVD e o uso de mecanismos ricos de interação, explorando as idéias de computação social e de objetos abertos de aprendizado (OER – *Open Educational Resources*) [Albright 2005], são tópicos atualmente em pesquisa no grupo.

A partir da experiência em ensino e desenvolvimento de módulos educacionais no domínio de Teste de Software, este artigo discute a ligação entre os ambientes e sistemas educacionais, as novas tendências em educação à distância, televisão digital e conteúdos educacionais abertos, e a adequação dos mecanismos de apoio ao desenvolvimento de módulos educacionais propostos frente a esse novo cenário.

O restante deste artigo estrutura-se da seguinte forma. A Seção 2 discursa sobre as ferramentas utilizadas para o gerenciamento de aprendizado no ICMC, refletindo as dificuldades encontradas e a evolução de seu uso na instituição. A Seção 3 descreve a AIM-CID e o Processo Padrão para o Desenvolvimento de Módulos Educacionais. A aplicação dos mecanismos propostos é discutida considerando-se o domínio de Teste de Software. A Seção 4 destaca alguns pontos da AIM-CID e do Processo Padrão que precisam ser revistos de modo a adequá-los à idéia de módulos educacionais abertos, à

educação à distância e à televisão digital. A Seção 5 resume os resultados esperados no contexto das novas tendências e perspectivas de ensino.

2 Ambientes e sistemas educacionais

Ambientes e sistemas educacionais têm por objetivo apoiar o processo de ensino e aprendizado, fornecendo um conjunto de ferramentas de comunicação síncrona e assíncrona para auxiliar a condução de trabalhos colaborativos, a avaliação e acompanhamento dos aprendizes. No âmbito do ICMC, alguns ambientes e sistemas educacionais têm sido utilizados no ensino de Engenharia de Software, acompanhando a evolução desses sistemas e as necessidades de ensino: WebCT [Goldberg et al. 1996], CoTeia [Arruda Jr. et al. 2002], Moodle [Moodle 2006] e Tidia/Ae [Projeto TIDIA (FAPESP), 2007]:

- **WebCT.** Fornece um conjunto de ferramentas que facilitam a criação de cursos baseados na Web, criando um ambiente de ensino e treinamento. Integra as seguintes funcionalidades: conferência, bate-papo, correio eletrônico, acompanhamento do aprendiz, suporte a projetos colaborativos, auto-avaliação, questionários, distribuição e controle de notas, glossário, controle de acesso, calendário do curso, geração automática de índices e pesquisa, entre outras. Além de ferramentas educacionais, também fornece um conjunto de ferramentas administrativas para auxiliar no gerenciamento do curso.

- **CoTeia.** Ferramenta de edição colaborativa baseada na Web. Sob a perspectiva de ensino e aprendizado, ela fornece um espaço colaborativo no qual os aprendizes podem trabalhar em grupo, ajudando-se mutuamente na articulação do conhecimento para a realização das tarefas e projetos propostos.

- **Moodle.** Sistema destinado à gestão de ensino e de trabalho colaborativo, permitindo a criação de cursos on-line, páginas de disciplinas, grupos de trabalho e comunidades de aprendizagem, seguindo uma abordagem sócio-construtivista da educação. O sistema é adequado para atividades inteiramente à distância e ainda pode apoiar e complementar atividades do ensino presencial. As principais funcionalidades do ambiente são: gerenciamento do site, usuários e cursos; módulos de interação entre os usuários (tarefas, bate-papo, enquete, fórum, *quiz*, repositório de arquivos, grupos de trabalho e glossário).

- **Tidia/Ae.** Possui, como base, o Sakai [Sakai Project 2005]. O processo de gerência de aprendizagem do Tidia/Ae é realizado por um conjunto de ferramentas: gerenciadores de contexto (responsável pelo controle do ciclo de vida de um ambiente de aprendizado), usuários e ferramentas; *whiteboard* (captura de informações do professor durante a aula de forma síncrona, por meio de uma lousa eletrônica); comunicador instantâneo; laboratório remoto (ferramenta que engloba todo o ciclo de vida de um experimento, desde a criação até a efetiva realização pelo usuário); hipertexto; correio eletrônico; portfólio; e fórum de discussão.

O WebCT foi o primeiro ambiente utilizado pelo ICMC, no início de 2000. Seu uso foi descontinuado em favor da CoTeia, uma ferramenta simples para usuários não familiarizados, que oferece mecanismos de edição de fácil uso e possui um bom suporte no instituto, com a rápida implementação de novas funcionalidades.

A CoTeia foi a ferramenta mais utilizada no instituto. O mecanismo ágil de edição de textos (*wiki*) e a possibilidade de edição e envio de material por parte dos alunos facilitava a manutenção do conteúdo das disciplinas. No entanto, seus mecanismos de controle de acesso eram limitados, permitindo a alteração indevida dos conteúdos.

Recentemente, alguns professores adotaram o Moodle para suas disciplinas. O Moodle é um ambiente completo, tal como WebCT, mas com a vantagem de ser um software livre. Ele não oferece a mesma facilidade de edição da CoTeia, mas possui um sistema de controle de acesso sofisticado, o que sana os problemas de segurança da CoTeia. Além disso, ele provê funcionalidades adicionais, como controle de submissão de trabalhos, cadastro de alunos e notas.

Por fim, este ano entrou em uso experimental o Tidia/Ae. Ele possui, como diferenciais, uma *wiki* e o suporte a ferramentas avançadas de colaboração e interação, como o quadro branco eletrônico (*whiteboard*), *tablet PCs*, comunicador instantâneo e laboratório remoto.

A Tabela 1 resume as características relevantes de cada ferramenta. Observa-se, por exemplo, que o Tidia/Ae, com seus ricos mecanismos de interação com os usuários, constitui uma plataforma adequada para a educação à distância. No entanto, todos os ambientes listados possuem uma deficiência: a geração dos conteúdos é uma tarefa a qual não são oferecidas ferramentas adequadas. De fato, apesar do suporte a objetos de aprendizado, a geração desses objetos é uma tarefa a ser realizada por outra ferramenta. Esse conceito de objeto fechado, sem possibilidade de evolução ou uso personalizado dentro do ambiente, não é adequado para os novos propósitos de educação. A AIM-CID, que define mecanismos para modelagem de conteúdos de um módulo educacional aberto, aborda tais aspectos. A seção a seguir apresenta uma visão geral a respeito da abordagem e do processo desenvolvidos, bem como os trabalhos em andamento para adequá-los às novas tendências e demandas de ensino.

Tabela 1. Comparação dos sistemas de gerenciamento de aprendizado.

	WebCT	CoTeia	Moodle	Tidia/AE
Software livre		X	X	X
Wiki		X		X
Facilidade de uso		X	X	X
Facilidade de manutenção			X	X
Controle de acesso	X		X	X
Interação rica				X
Suporte a padrões abertos	X		X	X
Suporte a objetos de aprendizado	X		X	X

3 Mecanismos de apoio ao desenvolvimento de módulos educacionais

Com o avanço da tecnologia de informação e comunicação, vários aspectos relacionados à criação de conteúdo educacional digital estão sendo investigados a fim de tornar mais efetivo o processo educacional nas modalidades presencial, híbrida e a distância. Sob essa perspectiva, a modelagem de conteúdos educacionais de qualidade representa um aspecto essencial para a estruturação do conhecimento, apoiando a identificação e definição de conceitos e possibilitando que eles sejam disponibilizados de forma coerente e ordenada.

No ICMC, uma das linhas de pesquisa explorada refere-se à modelagem de conteúdos educacionais e ao processo de engenharia dos mesmos. Um dos resultados obtidos foi a AIM-CID (Abordagem Integrada de Modelagem – Conceitual, Instrucional e Didática) e o Processo Padrão para o Desenvolvimento de Módulos Educacionais [Barbosa 2004].

A AIM-CID estabelece que os conteúdos educacionais sejam especificados em três modelos: conceitual, instrucional e didático. O Modelo Conceitual consiste em uma descrição de alto-nível do domínio de conhecimento que se deseja ensinar, representados com um mapa conceitual [Novak 1981]. Sua construção envolve a definição dos conceitos relevantes para a compreensão do domínio e a especificação da forma pela qual os mesmos se relacionam. O Modelo Instrucional define itens de informação (conceitos, fatos, procedimentos e princípios) e elementos instrucionais (exemplos, informações complementares, exercícios e avaliações), associando-os aos conceitos já identificados. O Modelo Didático associa os objetos anteriormente modelados, estabelecendo uma seqüência de apresentação entre eles. Ambos os modelos, instrucional e didático, empregam o modelo HMBS [Turine 1998], uma extensão da técnica de *statecharts* [Harel 1987], para especificar a estrutura semântica de hiperdocumentos.

O Processo Padrão para o Desenvolvimento de Módulos Educacionais foi estabelecido com base na norma ISO/IEC 12207 [ISSO/IEC 1995], a qual foi adaptada ao contexto de produção de módulos educacionais por meio da inclusão de práticas específicas de projeto instrucional, modelagem de conteúdos educacionais (com a incorporação da AIM-CID) e de desenvolvimento cooperativo e distribuído. Adicionalmente, foi proposto um modelo de maturidade de processos para a elaboração de módulos educacionais (CMMI-Educacional), visando a apoiar a especialização do processo padrão em diferentes níveis de maturidade.

Os mecanismos propostos no trabalho de Barbosa [Barbosa 2004] foram aplicados no domínio de Teste de Software, culminando nos módulos "Teste de Software: Teoria e Prática" (Figura 1), ITonCode (sobre técnicas de teste e inspeção) [Barbosa et al. 2008b], "Fundamentos de Programação e Teste de Software" [Barbosa et al. 2008a] e "Teste de Mutação" [Barbosa e Maldonado 2006a]. Estudos sobre os módulos gerados evidenciaram a efetividade de sua aplicação, observando-se um melhor desempenho dos alunos quanto à satisfação de requisitos de teste de aplicações-exemplos se comparado com métodos de ensino utilizados anteriormente na mesma disciplina [Barbosa et al. 2008b].

Ainda no contexto de Teste de Software, desenvolveu-se a ProgTest [Corte 2006, Barbosa et al. 2008a], uma ferramenta para submissão e avaliação de trabalhos práticos com base em atividades de teste. A ferramenta encontra-se integrada ao módulo "Fundamentos de Programação e Teste de Software".

De modo geral, os aspectos observados no desenvolvimento e utilização dos módulos educacionais evidenciaram a viabilidade da aplicação dos mecanismos, tanto no que se refere à modelagem de conteúdos como no contexto de padronização de processos para módulos educacionais. Os mecanismos estabelecidos propiciaram a reestruturação de materiais didáticos na área de Teste de Software, a fim de proporcionar o domínio e a disseminação de conhecimentos técnico-científicos na área.

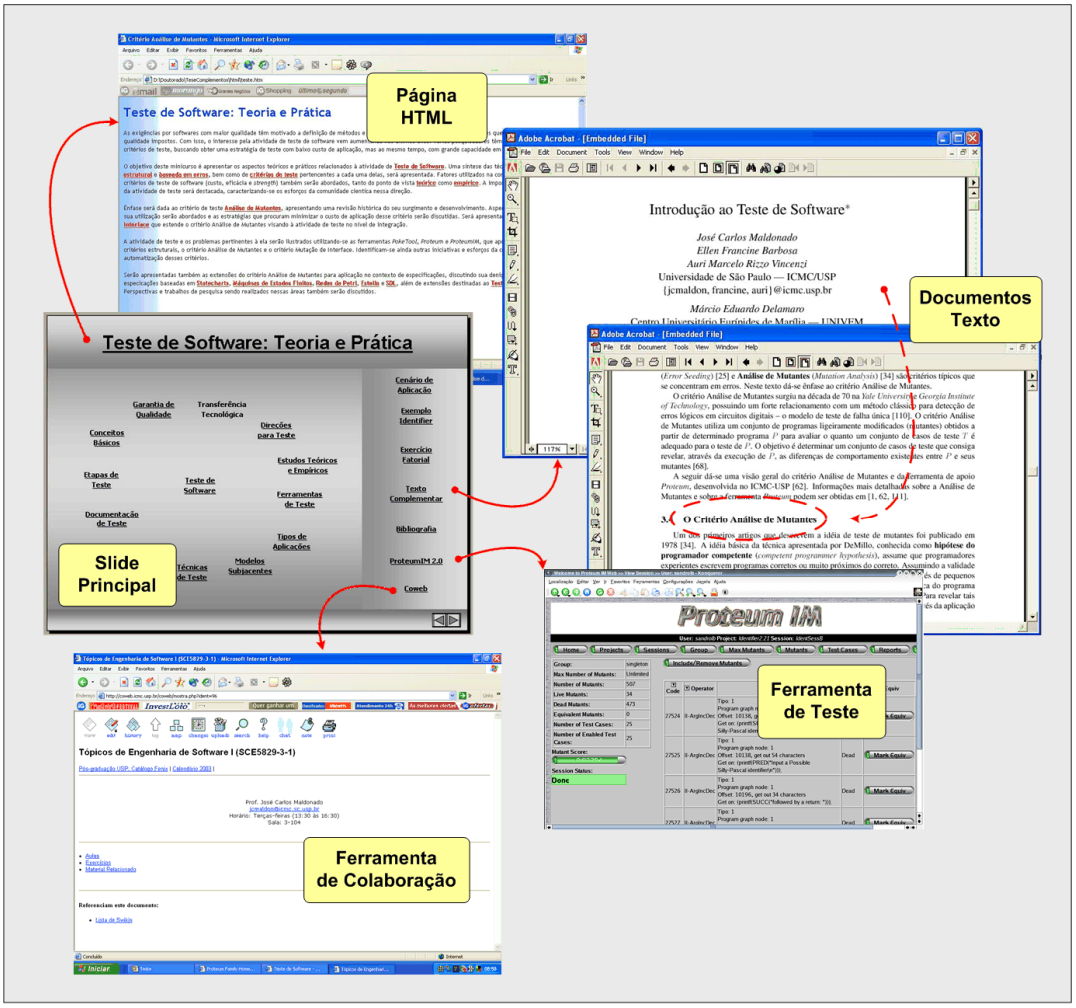


Figura 2. Módulo educacional "Teste de Software: Teoria e Prática".

Ressalta-se, entretanto, a necessidade de sua aplicação e avaliação em projetos mais abrangentes e com características diferenciadas. Assim, pesquisas estão sendo conduzidos nessa perspectiva. Um dos trabalhos em andamento propõe o desenvolvimento de uma ferramenta Web para apoiar a modelagem e a geração automática de conteúdos educacionais segundo a AIM-CID [Borges 2008]. A ferramenta tratará aspectos de simulação, validação e apresentação dos conteúdos modelados de modo a permitir que suas especificações sejam utilizadas na geração automática de conteúdos personalizados. Além disso, ela gerará conteúdo segundo padrões de metadados, garantindo a interoperabilidade, reúso e compartilhamento do conteúdo educacional gerado.

Em outra frente, investiga-se um processo padrão para módulos educacionais para a televisão digital [Silva 2008], incorporando o estado da arte da tecnologia do processo de software livre e objetos de aprendizado, bem como elementos da engenharia de conteúdo televisivo.

Espera-se, com a adoção de um modelo colaborativo de desenvolvimento e o uso de ferramentas de autoria, o controle do processo e de suas atividades, propiciando

conteúdos educacionais interativos e de qualidade para televisão digital na educação, seja ela presencial, híbrida ou à distância.

4 Perspectivas para módulos educacionais abertos para a televisão digital

Conforme ressaltado anteriormente, o Processo Padrão para o Desenvolvimento de Módulos Educacionais foi instanciado para cursos presenciais no ensino de Teste de Software. Uma das propostas de trabalhos em andamento é a sua utilização para outras modalidades e tipos de módulos educacionais (como, por exemplo, cursos à distância) e sua aplicação no contexto de desenvolvimento colaborativo e distribuído.

A televisão digital mostra-se como um meio próspero para educação à distância e oferece um contexto de colaboração entre seus usuários, tornando-a um veículo adequado para módulos educacionais e o uso do Processo Padrão.

A geração de um módulo educacional adequado à veiculação para a televisão digital é possível com a transformação do Modelo Didático (proposto como parte da AIM-CID) em um programa escrito uma linguagem declarativa, a NCL (*Nested Context Language*), compatível para execução na plataforma Ginga-NCL da SBTVD. Em linhas gerais, os estados do *statechart* subjacente ao Modelo Didático correspondem a contextos do Ginga-NCL e as ligações a ligações e âncoras. Vídeos podem ser modelados como itens de informação no Modelo Instrucional (também proposto como parte da AIM-CID) e inseridos como nós de mídia no documento NCL. No entanto, recursos de interação, possíveis com a televisão digital, não podem ser gerados de modo tão direto, devendo ser investigados mais detalhadamente.

Outro interesse de pesquisa é a construção de módulos educacionais abertos e a análise da AIM-CID e do Processo Padrão sob os auspícios do processo de software livre [Reis 2003]. Esse anseio casa-se com a expectativa de maior interatividade da televisão digital e a possibilidade de evolução do módulo educacional. Do ponto de vista da AIM-CID, novos conceitos podem ser sugeridos e, a partir deles, novos itens de informação podem ser definidos; usuários poderiam criar elementos instrucionais e disponibilizá-los publicamente. Assim, o programa deixaria de ser um conteúdo fechado (quanto à modificação de seu conteúdo, e não apenas quanto a questões de navegação, conforme atualmente abordado na AIM-CID) e poderia evoluir, de acordo com as necessidades de cada comunidade.

É importante ressaltar que tal evolução não pode transcorrer de modo desordenado. O estabelecimento de arquiteturas-padrão para as aplicações e a identificação das variabilidades permitiriam o estabelecimento de linhas de produtos. A cada comunidade estabelecer-se-ia um produto diferente, criado a partir de uma única linha. A capacidade de multiprogramação da televisão digital permitiria o envio de vários produtos (programas de televisão diferentes) ou de suas variações em cada segmento de um canal de televisão, permitindo a uma mesma região geográfica o acesso a diferentes produtos associados a um mesmo módulo educacional.

Os ambientes educacionais podem ser integrados a esse cenário, permitindo a organização dos programas de televisão, por meio de um guia de programação eletrônico gerado automaticamente pela agenda de um curso, a alteração dos conteúdos apresentados de acordo o material inserido no sistema e a geração

automática de programas conforme o rendimento dos alunos, citando apenas alguns dos exemplos possíveis.

5 Conclusão

Entre as linhas que têm sido objeto de pesquisa no contexto de ensino, destaca-se o desenvolvimento de módulos educacionais. No ICMC, mecanismos de apoio ao processo de desenvolvimento e modelagem de módulos educacionais têm sido investigados e definidos, propiciando a reestruturação de materiais didáticos na área de Engenharia de Software, mais especificamente dentro da temática de Teste de Software.

Em continuidade aos trabalhos realizados, pretende-se agora investigar e definir mecanismos de apoio ao processo de desenvolvimento aberto, cooperativo e distribuído de módulos educacionais. A idéia é que os conteúdos construídos sejam flexíveis, customizáveis e reutilizáveis, como forma de estabelecer mecanismos apropriados para contribuir na melhoria do processo de ensino, tornando-os compatíveis com as novas demandas impostas pelas transformações educacionais em curso. Ainda, o advento da SBTVD permite a exploração de novos recursos no ensino. A especialização dos mecanismos propostos para televisão digital e sua instanciação para a produção de módulos educacionais, com as considerações sobre conteúdos abertos, integração com ambientes educacionais e evolução controlada apresentadas neste artigo, deverão propiciar o ensino em uma nova perspectiva, especialmente quanto à modalidade de educação à distância.

Por fim, ressalta-se que tais aspectos deverão ser explorados não somente no contexto de Teste de Software, mas também em outras subáreas de Engenharia de Software e outras áreas de Computação e, ainda, considerando-se disciplinas de ensino básico e fundamental como, por exemplo, Matemática, Física e Biologia, entre outras.

6 Referências

ALBRIGHT, P. Open educational resources final forum report. Technical report, UNESCO. Internet Discussion Forum: Open Educational Resources - Open Content for Higher Education. 2005.

ARRUDA JR., C. R. E., IZEKI, C. A., e PIMENTEL, M. G. C. CoTeia: uma ferramenta colaborativa de edição baseada na web. In Workshop de Ferramentas e Aplicações do VIII SBMIDIA, p. 371--374, Fortaleza, Brasil. 2002.

BARBOSA, Ellen Francine. **Uma contribuição ao processo de desenvolvimento e modelagem de módulos educacionais**. Tese de Doutorado - Universidade de São Paulo, Instituto de Ciências Matemáticas e de Computação, São Carlos - SP, Brasil. Orientador: José Carlos Maldonado. 2004.

BARBOSA, E. F. e MALDONADO, J. C. Establishing a mutation testing educational module based on IMA-CID. In Second Workshop on Mutation Analysis, Raleigh, NC, USA. 2006a.

BARBOSA, E. F. e MALDONADO, J. C. An integrated content modeling approach for educational modules. In IFIP International Conference on Education for the 21st Cen-

tury - 19th IFIP World Computer Congress (WCC 2006), p. 17-26, Santiago, Chile. 2006b.

BARBOSA, E. F. e MALDONADO, J. C. Towards the establishment of a standard process for developing educational modules.In 36th Annual Frontiers in Education Conference (FIE 2006), San Diego, CA. CD-ROM. 2006c.

BARBOSA, E. F., SILVA, M. A. G., CORTE, C. K. D., e MALDONADO, J. C. Integrated teaching of programming foundations and software testing.In 38th Annual Frontiers in Education Conference (FIE 2008), Saratoga Springs, NY. 2008a.

BARBOSA, E. F., SOUZA, S., e MALDONADO, J. C. An experience on applying learning mechanisms for teaching inspection and software testing. In 21st IEEE-CS Conference on Software Engineering Education and Training, p. 189-196, Charleston, South Carolina. 2008b.

BORGES, Vanessa Araujo. **Uma contribuição ao desenvolvimento de ferramentas de apoio à modelagem e geração de conteúdos educacionais.** Monografia de Qualificação de Mestrado. Orientadora: Ellen Francine Barbosa. 2008.

CORTE, Camila Kozlowski Della. **Ensino integrado de fundamentos de programação e de teste de software**. Dissertação de Mestrado, Universidade de São Paulo, São Carlos, SP. Orientador: José Carlos Maldonado. 2006.

GOLDBERG, M. W., SALARI, S., e SWOBODA, P. World Wide Web - Course Tool: An environment for building WWW - based courses. Computer Networks and ISDN Systems, 28(7-11):1219 - 1231. 1996.

HAREL, D. Statecharts: A visual formalism for complex systems. Science of Computer Programming, 8:231--274. 1987.

ISO/IEC. **ISO/IEC 12207 - standard for information technology - software life cycle processes**. Standard. 1995.

MOODLE, D. G. **Moodle – A free, open source course management system for online learning**. Disponível em: http://moodle.org/ [08/04/2007]. 2006.

NOVAK, Joseph D. **Uma Teoria de Educação**. Editora Pioneira, São Paulo, 1981.

PROJETO TIDIA (FAPESP). **Programa de Tecnologia da Informação para o desenvolvimento da Internet Avançada**. Disponível em: http://tidia-ae.incubadora.fapesp.br/portal [08/11/2007]. 2007.

REIS, Christian Robottom. **Caracterização de um modelo de processo para projetos de software livre**. Dissertação de Mestrado, Universidade de São Paulo, São Carlos, São Paulo. Orientadora: Renata Pontin de Mattos Fortes. 2003.

SAKAI PROJECT. **Sakai**. Programa de computador. 2005.

SILVA, Marco Aurélio Graciotto. **Uma contribuição ao processo de desenvolvimento de módulos educacionais abertos para televisão digital**. Projeto de Doutorado. Orientador: José Carlos Maldonado. 2008.

TURINE, Marcelo Augusto Santos. **HMBS: Um modelo baseado em statecharts para a especificação formal de hiperdocumentos**. Tese de Doutorado, IFSC-USP, São Carlos, SP. 1998.

Discussões e Questionamentos sobre a Evolução de Cursos de Engenharia de Software na Educação Superior Brasileira

Antonio M. P. de Resende[1] Ana Rubélia M. L. Resende[1] Heitor A. X. Costa[1]
Fábio F. Silveira[2] Valter V. de Camargo[1]

[1] PqES – Grupo de Pesquisa em Engenharia de Software -
Departamento de Ciência da Computação – Universidade Federal de Lavras (UFLA)
P.O. Box 3037, 37200-000, Lavras – MG, Brazil
[2] Departamento de Ciência e Tecnologia (DCT)
Universidade Federal de São Paulo (UNIFESP) - São José dos Campos,SP -Brasil

{tonio,heitor }@dcc.ufla.br, rubelia.resende@gmail.com, fsilveira@unifesp.br,
valter@dcc.ufla.br

Abstract. In 2004, the Joint Task Force on Computing Curricula formed by IEEE Computer Society and Association for Computing Machinery (ACM) published the Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering (SE) in order to direct lawless SE undergraduate courses [ACM/IEEE-SE, 2004]. This article shows up data about SE and Computer Science courses between 1991 and 2006. Data present the almost inexistence of Software Engineering courses in Brazil and an opportunity for Universities to eliminate this gap.

Keywords: Software Engineering Courses, Software Engineering in Brazil.

Resumo. Em 2004, uma força tarefa criada pela IEEE e ACM publicou um currículo de referência para a área de Engenharia de Software (ES), cujo objetivo era nortear a criação explosiva de cursos de graduação em ES no mundo [ACM/IEEE-SE, 2004]. O presente artigo mostra que esta explosão de cursos em ES não ocorreu no Brasil e apresenta dados dos cursos de ES e Ciência da Computação no período de 1991 até 2006. Os dados mostram que cursos de graduação em ES no país são praticamente inexistentes, havendo grande oportunidade para Universidades e grande lacuna para atender profissionais em computação.

Palavras-chave: Cursos de Engenharia de Software, Engenharia de Software no Brasil.

1 Introdução

O objetivo deste trabalho consiste em apresentar uma análise baseada em dados do período de 1991 até 2006, disponibilizados pelo Ministério da Educação (MEC) da República Federativa do Brasil, cujo objetivo é mostrar, quantitativamente, a pouca importância dada à educação de ES no Brasil, em nível superior, não considerando os cursos de pós-graduação. Ao final deste artigo o leitor será capaz de perceber que existe uma grande diferença entre a importância dada, por outros países, na Educação, refletida na explosão de cursos em ES relatado em [IEEE Computer Society, 2004] [ACM/IEEE-SE, 2004], enquanto no Brasil praticamente desconsidera-se seu valor no preparo de profissionais para esta área, em nível superior, sem contemplar a pós-graduação. Na seção 2, apresentam-se dados da área de ES no Brasil, realizando comparações com o curso e a área de Ciência da Computação. Na seção 3, os dados são discutidos e trabalhos em andamento e futuros são propostos. Na seção 4, apresentam-se as conclusões, ressaltando-se a necessidade de se incentivar a criação de cursos de graduação em ES no Brasil e a necessidade de um planejamento para que tais cursos já comecem dentro de um formato padrão, a fim de evitar disparidades e inadequações dos conteúdos programáticos, cuja a correção é custosa para professores e alunos.

2 O Cenário da Engenharia de Software no Brasil

A partir de dados consultados diretamente na base de dados de cursos do Ensino Superior, sem considerar pós-graduação, do Ministério da Educação (MEC) do Brasil, abrangendo o período de 1991 até 2006, vários gráficos foram construídos, objetivando avaliar o cenário da ES no Brasil [MEC, 2008]. A base de dados do MEC possui todas as áreas de conhecimento, porém esta investigação concentrou-se na área de Ciência da Computação e dentro desta área os seguintes cursos do Ensino Superior (graduação e seqüenciais) são relacionados a seguir: a) Administração de Redes; b) Banco de Dados; c) Ciência da Computação; d) Computação Gráfica; e) Engenharia de Computação; f) Engenharia de Software; g) Informática; h) Linguagens de Programação; i) Robótica; j) Sistemas Operacionais; k) Tecnologia da Informação; l) Tecnologia em Desenvolvimento de Softwares; e m) Tecnologia em Informática.

A fim de reduzir a complexidade da análise realizada, primeiramente, verificaram-se as informações pertinentes aos cursos de ES e, em seguida, realizaram-se comparações da ES com o curso de Ciência da Computação, com a área de Ciência da Computação e com o cenário de todos os cursos no Brasil. A área de Ciência da Computação refere-se ao somatório dos dados referentes a todos os cursos supracitados. O leitor deve ficar muito atento ao termo "Ciência da Computação" (CC) que aparece referindo-se, em alguns trechos, ao **curso** de Ciência da Computação e em outros trechos vinculado a **área** de Ciência da Computação. Toda vez que este termo aparecer, ele estará devidamente associado ao **curso** ou a **área**, sem o destaque de negrito e sublinhado dado neste parágrafo.

2.1 Vagas Ofertadas

A Figura 3 apresenta a evolução da oferta de vagas de ES ofertados. Nota-se, em 2000, que as ofertas começaram com 30 vagas, subindo para 150 em 2001, havendo queda para 30 vagas em 2004, permanecendo neste número até 2006. Portanto, o saldo no período de 1991 a 2006 foi de 30 vagas em 1 curso de ES no país. A Figura 4 apresenta a evolução da oferta de vagas do curso de ES por região, havendo cursos apenas nas regiões sul e sudeste. O primeiro curso foi criado em 2000 no sul, estado do Paraná, com 30 vagas. O segundo curso foi criado no Sudeste, estado de São Paulo, com 120 vagas, e encerrando as atividades a partir de 2004.

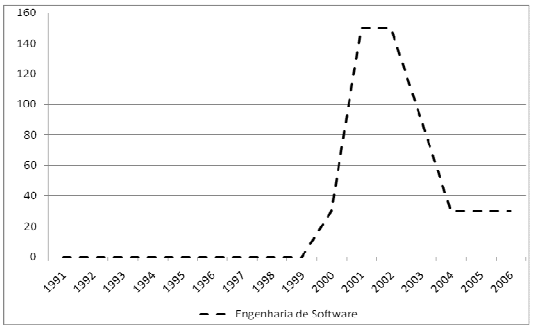


Figura 3: Evolução do número de vagas para cursos de ES ofertados

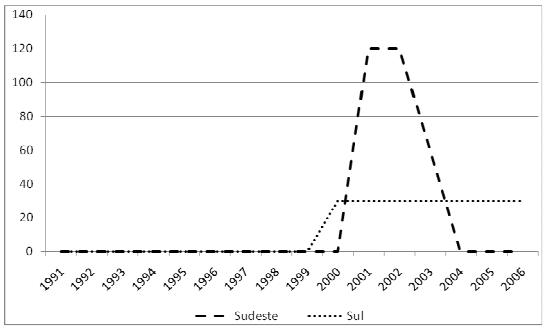


Figura 4: Evolução do número de vagas para cursos de ES ofertados por região

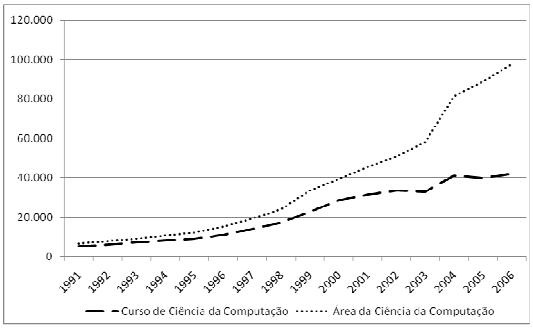


Figura 5: Evolução da oferta de vagas do curso e da área de CC

A Figura 5 apresenta a evolução da oferta de vagas do curso de CC, comparando com a área de CC, mostrando que a área de CC deixou de ser representada apenas por cursos de CC.

Nota-se, nos dados do MEC, que a oferta de vagas na área de CC aumentou 15 vezes, enquanto o curso de CC, em mais de 8 vezes, no período.

2.2 Número de Candidatos Inscritos

A Figura 6 apresenta a evolução dos candidatos inscritos no processo seletivo para cursos de ES com 366 candidatos inscritos em 2000, ápice em 2002 com 534 e redução para 317 inscritos em 2006, devido o fechamento de um dos cursos.

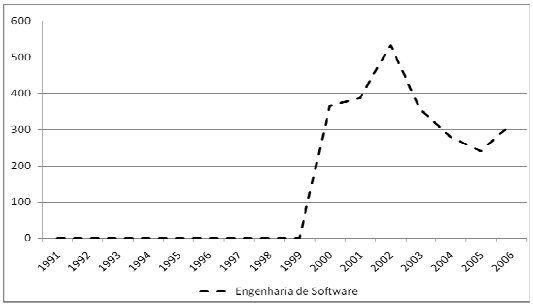


Figura 6: Evolução do número de candidatos para cursos de ES

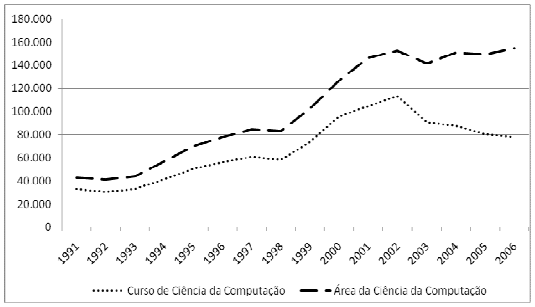


Figura 7: Evolução do número de candidatos aos cursos e a área de CC

A Figura 7 apresenta a evolução dos candidatos inscritos em processos seletivos de cursos de CC, comparando com os dados da área de CC. Nota-se que o número de inscritos nos cursos de CC aumentou em 2,36 vezes e na área de CC 3,6 vezes, no período.

2.3 Evolução da Relação Candidatos/Vagas

A Figura 8 apresenta a evolução da relação candidatos/vagas no processo seletivo para cursos de ES. Nota-se, em 2000, quando os cursos começaram a ser ofertados, uma relação de 12,2 candidatos/vaga, caindo abruptamente em 2001 para 2,6 candidatos/vaga. Em 2006, a relação foi de 10,6 candidatos/vaga. A Figura 9 apresenta a evolução da relação candidatos/vaga dos processos seletivos de cursos de CC,

comparando com os dados da área de CC.

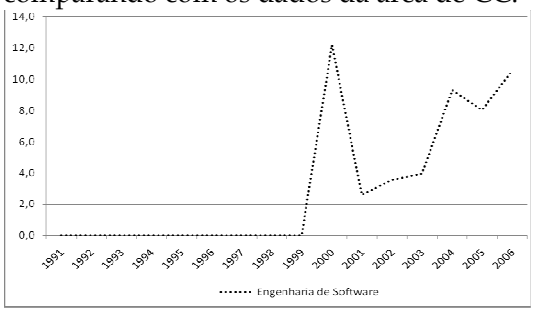


Figura 8: Evolução do número de candidatos para cursos de ES

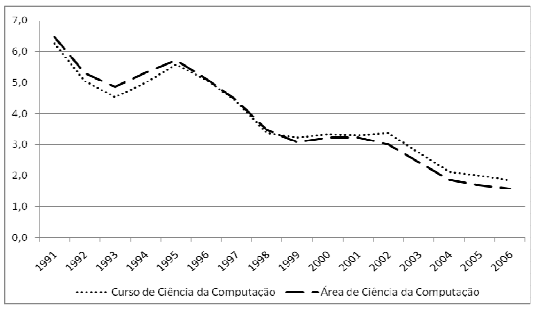


Figura 9: Evolução do número de candidatos aos cursos e a área de CC

3 Discussão dos Dados e Questões em Aberto

Nesta seção, realiza-se uma breve discussão dos dados apresentados, objetivando a compreensão do cenário da ES e sua relação com cursos CC e a área de CC no Brasil.

3.1 Vagas Ofertadas

O surgimento do curso de ES em 2000 não pode ser considerado tardio se tomarmos como referência que os primeiros documentos norteadores da área como o Currículo da IEEE/ACM para ES e o SWEBoK tiveram suas versões mais completas liberadas em 2004. Por outro lado, justamente em 2004 quando os documentos foram liberados, observa-se um declínio no número de vagas ofertadas, mostrado na Figura 3. Era de se esperar que mais cursos de ES fossem criados, aproveitando a nova nomenclatura, os documentos recém lançados por instituições renomadas, a discussão da área e de seus cursos por centenas de profissionais espalhados pelo mundo, dentre outros. Porém, o Brasil tomou caminhos opostos, sendo indispensável o levantamento das seguintes questões: a) Cursos de ES não possui valor estratégico para o país?; b) O que ocasionou o fechamento de um curso de ES no estado de São Paulo, quando o cenário geral apontava para um aumento na criação destes cursos, como tem ocorrido em outras partes do mundo?; c) Os alunos não estão interessados na área? Se sim, por que tal desinteresse?; d) As instituições de ensino superior não estão interessadas em oferecer à sociedade o curso de ES? Se sim, por que tal desinteresse?; e e) Por que a instituição do Paraná que ofereceu o curso não aumentou seu número de vagas?

Dividindo o total das 97.786 vagas ofertadas, mostrado na Figura 5, pelos 13 cursos apresentados na seção 2 , obtém-se a média de 7.522 vagas oferecidas para cada curso da área de CC em 2006, implicando que a ES deveria ter oferecido, em 2006, este número de vagas à sociedade, caso houvesse uma procura/oferta equilibrada entre os cursos. Porém, atualmente ofertam-se 30 vagas. Desta forma, pergunta-se: a) Por que tamanha diferença?; e b) Por que houve diminuição das vagas ofertadas com uma tendência de aumento no mundo inteiro?

3.2 Número de Candidatos Inscritos na ES e no Ensino Superior

Na Figura 6 nota-se uma oscilação no número de candidatos inscritos, voltando a crescer em 2006. Considerando a ES como novo curso e a demanda no mercado devido à crescente comercialização de software no mundo, pergunta-se: a) Existe demanda real no Brasil, atualmente?; b) A baixa procura estaria associada à falta de divulgação de tais cursos?; c) A baixa procura estaria associada à falta de esclarecimento junto a Sociedade da formação provida pela ES e o no que ela difere de Sistemas de Informação, Ciência da Computação e Engenharia da Computação?; d) Como houve desacoplamento do número de candidatos inscritos no curso de CC e na área de CC, como mostrado na Figura 7, por que o número de inscritos em ES não aumentou?; e e)

Por que o curso no estado mais populoso fechou, São Paulo, e no Paraná ele se manteve?

3.3 Evolução da Relação Candidatos/Vagas na ES e no Ensino Superior

Na Figura 9, observa-se uma queda considerável na relação candidatos/vaga dos cursos de CC e da área de CC, podendo-se concluir que: a) A concorrência em tais cursos diminuiu. Conseqüentemente, isto implica numa diminuição de ingressantes melhor preparados; e b) A relação candidatos/vaga dos cursos de ES, mostrado na Figura 5, apresenta-se desacoplado a relação candidatos/vaga dos cursos e da área de CC. Acredita-se que este desacoplamento deva-se ao fato de se ter uma série histórica muito curta, pois o fato dos cursos de ES ser uma novidade no mercado, isto não contribuiu para que houvesse um aumento crescente inicial na procura por tais cursos.

3.4 Trabalhos em Andamento e Futuros

É importante a realização de um macro estudo do Ensino da ES no Brasil e da qualidade da formação dos profissionais, evitando, o quanto antes, a explosão desordenada de cursos nesta área. O grupo PqES da UFLA tem aplicado esforços nos seguintes trabalhos: a) Levantar um estudo dos cursos de graduação em ES existentes nos EUA e União Européia, comparando os com o Brasil; b) Realizar um estudo do número de cursos Lato Sensu em ES, considerando o aumento da quantidade de propagandas de tais cursos no Brasil; c) Levantar os cursos de ES existentes no país, atualmente, e verificar quais estão adequados ao currículo de referência da ACM; d) Fazer um estudo das importações e exportações de software brasileiro e levantar as estimativas de comercialização de software no Brasil e no exterior, estabelecendo a importância de se formar profissionais devidamente treinados a partir do cenário econômico brasileiro e mundial nesta área; e e) Propor diretrizes norteadoras para criação de cursos na área de ES, evitando cursos com discrepâncias nas grades curriculares e o trabalho de ajuste.

4 Conclusões

Este artigo apresentou uma análise de dados do MEC, de 1991 até 2006, relacionados a cursos de ES, CC e da área de CC, não considerando cursos de pós-graduação. A partir desta análise levantaram-se implicações, ponderações e questões em aberto, a fim de nortear um crescimento ordenado dos cursos de ES no Brasil. É indispensável que as questões em aberto sejam respondidas, a fim de se entender o porquê dos cursos de ES não terem um crescimento explosivo como relatado no currículo da IEEE/ACM [ACM/IEEE-SE 2004]. Considerando que inúmeros documentos de políticos e especialistas informam a importância estratégica da ES para o país, é necessário compreender o que está impedindo o desenvolvimento destes cursos no Brasil. Mais informações são necessárias e o aprofundamento desta análise está sendo realizado pelo Grupo de Pesquisa em ES (PqES) da Universidade Federal de Lavras.

5 Referências

ACM/IEEE-SE Task Force on Computing Curricula (2004) "Software Engineering 2004 - Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering ", http://sites.computer.org/ccse/SE2004Volume.pdf, um volume da série Computing Curricula Series, August 23, 2004.

COHEN, N. “How does ada affect the way we teach software engineering? How does software engineering affect the way we teach ada?” In CSC '84 - Proceedings of the ACM 12th Annual Computer Science Conference on SIGCSE Symposium, publicado pela ACM, 1984.

IEEE Computer Society, “SWEBOK - Guide to the Software Engineering Body of Knowledge”,

2004 Version

MEC “Base de dados de graduações do MEC”. Base de dados pertencente ao Ministério da Educação (MEC) do Brasil, consulta feita em 01/07/2008.

O Uso de Jogos Educacionais para o Ensino de Gerência de Projetos de Software *

Rafael Prikladnicki[1] Christiane Gresse von Wangenheim[2]

[1]Faculdade de Informática (FACIN) – PUCRS – POA – RS
[2]Centro de Educação São José – LQPS – UNIVALI – São José – SC

rafaelp@pucrs.br, gresse@gmail.com

Abstract. Nowadays, there is an increasing demand for the learning of project management and for non-trivial ways to teach those that have no experience in this area. For this reason, this paper presents the idea to use games to support software project management education, based on the experiences of PUCRS and UNIVALI. We motivate the application of educational games for project management and present three games developed at the two institutions for the education in this area.

Keywords: Software project management, computer-based games, educational games.

Resumo. Hoje em dia, existe uma demanda cada vez maior pelo aprendizado de gerenciamento de projetos e de formas não-triviais para ensinar aqueles que ainda não possuem experiência na área. Por este motivo, este artigo apresenta a idéia de usar jogos para apoiar o ensino de gerência de projetos de software, a partir da experiência da PUCRS e na UNIVALI. Apresentamos a motivação para o uso de jogos no ensino de gerência de projetos e três jogos desenvolvidos nas duas instituições para o ensino nesta área.

Palavras-chave: Gerência de projetos de software, jogos baseados em computador, jogos educacionais.

---

* O jogo X-MED foi financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq – Brasil e pela UNIVALI (Universidade do Vale do Itajaí). Informações sobre Planager podem ser encontradas em http://www.inf.pucrs.br/~rafael/Planager. Informações sobre o jogo Scrumming podem ser encontradas em http://www.inf.pucrs.br/~rafael/Scrumming.

1 Introdução

Existe uma demanda cada vez maior pelo aprendizado em gerenciamento de projetos e de formas não-triviais para ensinar aqueles que ainda não possuem experiência na área. Foi observado, do ponto de vista acadêmico, que as disciplinas de gerência de projeto de software em cursos de graduação e pós-graduação não possuem o efeito desejado se o profissional não tiver alguma vivência prática prévia, por mais simples que seja (Reif & Mitri, 2005). Assim, acredita-se que a abordagem didático-pedagógica para alunos iniciantes na área deve ser diferente das abordagens atuais. Deve-se buscar uma abordagem mais prática e interativa, para que a falta de maturidade e a inexperiência não comprometa o ensino e o aprendizado (Kieling & Rosa, 2006).

Neste contexto, principalmente na área de gerência de projetos de software, surgiram alguns jogos para apoiar o ensino, como, por exemplo, TIM: The Incredible Manager (Dantas et al, 2004), Virtual Team (Gurgel et al, 2006), SimSE (Oh Navarro & van der Hoek, 2004) e IT Manager Game (http://itmg2.intel.com). Pensando neste cenário, este artigo tem como principal objetivo apresentar o uso de jogos como um método alternativo para o ensino de gerência de projetos de software, visando facilitar o aprendizado dos alunos, a partir de experiências da PUCRS e da UNIVALI.

O artigo está organizado em 5 seções: as seções 2 e 3 apresentam os conceitos relacionados com ensino de gerência de projetos e jogos educacionais, a seção 4 apresenta os jogos já desenvolvidos ou em desenvolvimento, e a seção 5 apresenta as conclusões.

2 Ensinando gerência de projetos de software

O gerenciamento de projetos de software (GPS) é a aplicação de conhecimento, habilidades, ferramentas e técnicas às atividades do projeto a fim de atender aos seus requisitos. O gerenciamento de projetos é realizado através da aplicação e da integração de processos em cinco etapas do ciclo de vida de gerenciamento de projetos: iniciação, planejamento, execução, monitoramento e controle, e encerramento (PMBOK, 2004).

O gerente de projetos é a pessoa responsável pela realização dos objetivos do projeto. No entanto, o entendimento e a aplicação do conhecimento, das habilidades, das ferramentas e das técnicas amplamente reconhecidas como boas práticas não são suficientes isoladamente para um gerenciamento de projetos eficaz. Um gerenciamento de projetos eficaz exige que se entenda e use o conhecimento e as habilidades de pelo menos cinco áreas de especialização (PMBOK, 2004): Conjunto de conhecimentos em gerenciamento de projetos; conhecimento, normas e regulamentos da área de aplicação; entendimento do ambiente do projeto; conhecimento e habilidades de gerenciamento geral e habilidades interpessoais.

Neste contexto, o uso de métodos tradicionais para ensino de conceitos de gerência de projetos acaba não sendo suficiente. Um gerente de projeto precisa aprender não apenas com a teoria, mas com a prática e vivência de projetos. Sendo assim, a experiência prática precisa ser incorporada nas estratégias de ensino. Uma opção que tem se mostrado eficiente tem sido o uso de casos reais. Mais recentemente, outros métodos começaram a aparecer com mais freqüência, tais como jogos (jogos de tabuleiro, cartas ou computador), simuladores, *role-plays*, entre outros. Neste artigo, concentramos nossa contribuição no uso de jogos educacionais por computador.

3 Jogos Educacionais

Jogos têm sido utilizados para ajudar no ensino de diversas áreas do conhecimento e muitas vezes despertam maior interesse por parte do aluno (Gramigma, 1994). Segundo Betz (1995), existe uma grande associação entre jogos e aprendizado. Os jogos de computador permitem a visualização e a experimentação de conceitos e possuem ambientes que despertam a criatividade dos jogadores. Além disso, a competição e definição de objetivos são componentes motivadores nestes jogos (Neal, 1990). Pesquisas cognitivas também sugerem que a percepção humana e a ação estão profundamente interconectadas (Gee, 2004). Desta forma, o uso de jogos para treinar, aprender e executar atividades reais em ambientes realísticos pode melhorar o desempenho dos alunos, pois possibilita a vivência de experiências de aprendizagem produzidas individualmente de acordo com o estilo de cada aluno. O uso de jogos de computador como ferramenta para potencializar o ensino tem sido demonstrado em diversos estudos (Ribeiro et al, 2006), e isto não é diferente na gerência de projetos.

Considerando o uso de jogos na educação universitária, muitos são voltados para a área de gerência de projetos em geral. Mas já existem alguns jogos especificamente voltados para GPS. A maioria destes jogos simula o planejamento e monitoramento e controle de um projeto, focando, tipicamente, em algumas áreas de conhecimento. Desta forma, observa-se uma oportunidade interessante na exploração do tema complementando os jogos em tópicos variados da GPS, considerando também "*soft skills*" (tais como comunicação, liderança, etc.).

4 Jogos em GPS: A Experiência da PUCRS e da UNIVALI

A PUCRS e a UNIVALI têm pesquisado, nos últimos anos, o planejamento, especificação e desenvolvimento de jogos para ensino de GPS. Até o presente momento, dois jogos foram desenvolvidos na Faculdade de Informática da PUCRS, e um no LQPS da UNIVALI. Os jogos são descritos a seguir.

4.1 Planager

O jogo *Planager* foi desenvolvido para apoiar o ensino de conceitos de gerência de projetos (Kieling et al, 2006). O jogo proposto não possui como objetivo simular todos os processos utilizados na gerência de projetos, mas foca o grupo de processos de planejamento. Devido à limitação de tempo para o desenvolvimento da primeira versão, foram escolhidos cinco processos de planejamento de duas das áreas de conhecimento do PMBOK 2004 (corpo de conhecimento em gerência de projeto do PMI – *Project Management Institute*): gerenciamento do escopo e gerenciamento do tempo, pois elas possuem processos que servem de base para uma grande quantidade de outros processos, que poderão ser utilizados em futuros módulos do jogo. Na gerência de escopo foram escolhidos os processos: definição do escopo e criação da EAP (Estrutura Analítica do Projeto). Na gerência de tempo foram escolhidos os processos: definição da atividade, seqüenciamento de atividades e desenvolvimento do cronograma (com foco no cálculo do caminho crítico).

O desenvolvimento da ferramenta priorizou uma metodologia diferenciada dos treinamentos tradicionais, com um foco maior na didática e não tanto na necessidade de decorar conceitos. Desta forma, o objetivo foi proporcionar um aprendizado mais prático e divertido para o aluno. Dois módulos foram criados: o módulo de tutorial,

onde o aluno pode revisar os conceitos de gerência de projetos aprendidos em aula e ter uma visão de como é o jogo; e o módulo jogo, onde o aluno pode praticar seus conhecimentos de uma forma interativa. O objetivo do jogo é fazer com que o jogador passe por várias fases, sendo avaliado no final de cada uma delas.

Para o desenvolvimento, utilizou-se a linguagem de programação Java, usando JAVA JDK 5 como sistema *desktop*. O jogo foi desenvolvido utilizando-se a arquitetura cliente/servidor para manter a aplicação independente de recursos *web*. Esta abordagem possibilita que o jogo seja utilizado em diversos locais e que os alunos possam levar cópias dele para casa para aperfeiçoar os seus conhecimentos.

O jogo possui dois tipos de usuários: o administrador e o jogador. O jogador pode utilizar o modulo tutorial para aprender sobre gerência de projetos e pode jogar em diversos cenários cadastrados na base de dados do jogo. Somente o administrador é capaz de adicionar, modificar e remover cenários. Cenários são representações de projetos e são compostos por uma descrição e por cinco fases: escopo, EAP (Figura 1), definição de atividades, seqüenciamento de atividades (Figura 2) e caminho crítico.

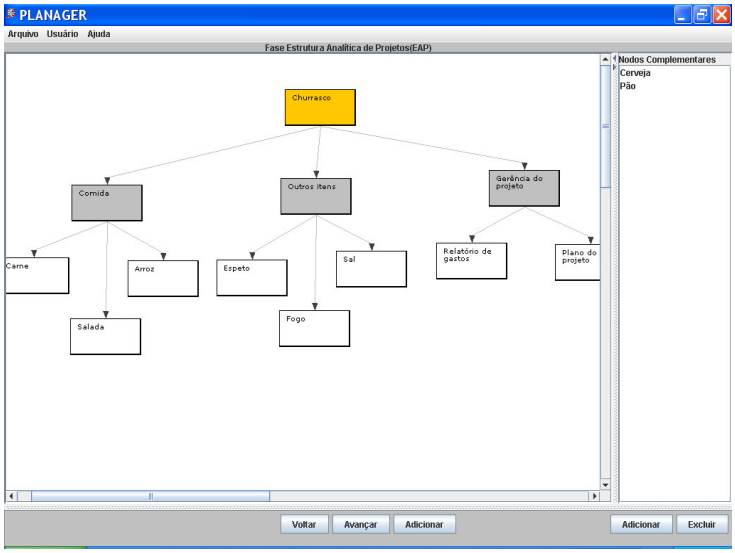


Figura 10. Criação da EAP

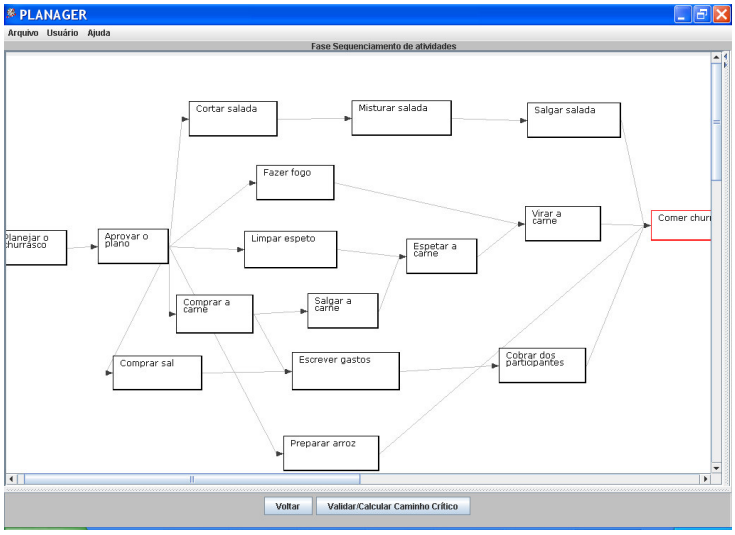


Figura 11. Seqüenciamento de atividades

Esta seqüência de fases é geralmente a mesma seqüência que os gerentes utilizam em um projeto real. O gerente de projetos utiliza as informações geradas pelos primeiros processos como entrada para vários outros (por exemplo, a definição de atividades gera informações que são utilizadas no seu seqüenciamento). O jogador, assim como o gerente de projetos, deverá utilizar as informações contidas nas fases anteriores para conseguir resolver corretamente as próximas fases do jogo. Mais informações sobre as fases podem ser encontradas em Prikladnicki et al (2007).

Como a ferramenta possui um módulo de criação de cenários, é possível encorajar os mais experientes a participar do jogo no papel de administradores criando novos cenários, aprimorando os conceitos utilizados durante jogo de forma mais aprofundada. Do ponto de vista de aplicabilidade dos resultados, o jogo está pronto para ser utilizado em disciplinas de graduação que ensinam conceitos básicos de gerência de projetos de software, e está em fase de avaliação com alunos do curso de Bacharelado em Sistemas de Informação da PUCRS[2].

4.2 Scrumming

O Jogo *Scrumming* (Isotton, 2008) é um jogo educacional que simula o uso de algumas práticas do Scrum (Schwaber, 2004), e busca suprir as necessidades encontradas no ensino de métodos ágeis para gerenciamento de projetos. O jogo não possui como objetivo simular todos os processos utilizados pelo Scrum. Ele está focado inicialmente na definição e simulação de *sprints* (conjunto de atividades de desenvolvimento realizadas durante um período pré-definido, usualmente entre duas e quatro semanas).

O jogo possui dois tipos de usuários, o administrador e o *Scrum Master* (membro da equipe do projeto, com responsabilidade de aplicar os valores e práticas do Scrum, remover obstáculos, garantir a plena funcionalidade e produtividade da equipe, e garantir a colaboração entre os diversos papéis e funções). O jogo possui ainda dois módulos: administrativo, utilizado pelo usuário administrador para realizar atividades que precedem a simulação (adicionar funcionários, adicionar ou remover atividades iniciais ao projeto, etc.); e módulo de simulação: utilizado pelo usuário para simular *sprints* de um projeto. Para a simulação, cinco tipos de funcionários podem ser criados, quais sejam: gerente de projetos, líder técnico, desenvolvedor, engenheiro de teste e testador. Além disso, para cada tipo de funcionário existem três níveis de senioridade: sênior, intermediário e iniciante, variando na competência e produtividade.

Para simular um *sprint*, é necessário cadastrar um conjunto de atividades no *backlog* do produto (lista priorizada dos requisitos e atividades do projeto). Após isto, um subconjunto destas atividades é selecionado para formar o *sprint backlog* (lista de tarefas que define o trabalho da equipe durante o *sprint*). Na simulação, o usuário age como se fosse um *Scrum Master*, realizando tarefas tais como definir um *sprint* (Figura 3), monitorar o andamento do *sprint* através da *taskboard* (um painel onde podem ser colocadas várias informações importantes para o acompanhamento do *sprint*), visualizar o gráfico de *burndown* (principal gráfico de controle do Scrum), e ainda adicionar ou remover atividades do *backlog* do produto.

A primeira versão do jogo foi desenvolvida utilizando JAVA JDK 6 como sistema *desktop* e banco de dados MySQL 5.0. Devido à limitação de tempo no desenvolvimento desta primeira versão, alguns processos, tais como a alocação de recursos para as atividades, normalmente realizado pelo *Scrum Master* são executados de forma

2 O Planager ficou entre os 12 melhores trabalhos do ciclo 2006 do PBQP Software/MCT.

automática pelo sistema. Além disso, o jogo não permite interação com todos os processos do Scrum, apenas a simulação de múltiplos *sprints*.

Em relação ao uso, num primeiro momento o jogo foi concebido pensando em ser utilizado por profissionais da indústria ou alunos de graduação. Entende-se que, por apresentar os conceitos de forma genérica, é possível utilizá-la com foco na gerência de projetos como um todo e não apenas em projetos de software.

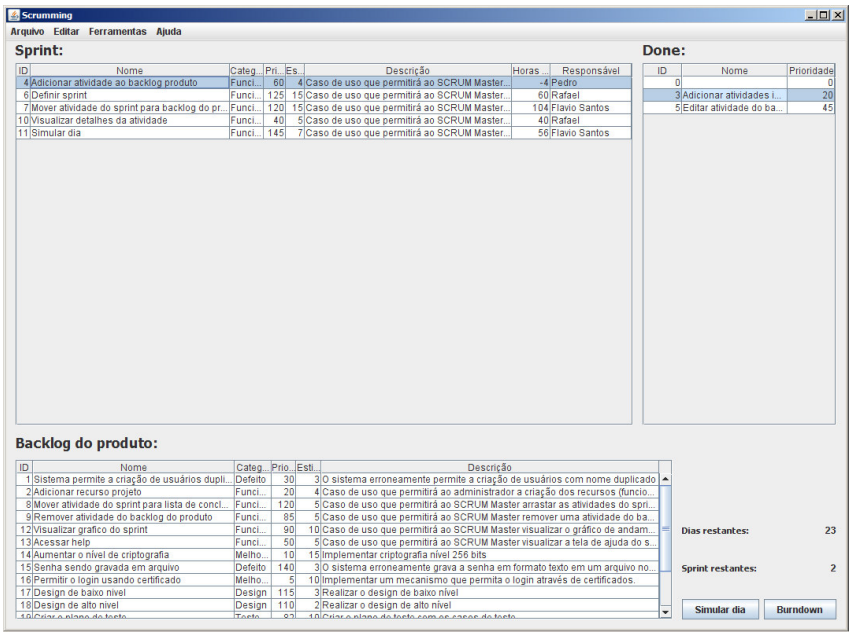


Figura 12. Definição de um *sprint*

Do ponto de vista de aplicabilidade dos resultados, o jogo está pronto para ser utilizado em disciplinas de graduação que ensinam conceitos básicos de gerência de projetos de software, e está em fase de avaliação com alunos do curso de Bacharelado em Sistemas de Informação da PUCRS.

4.3 *X-MED v1.0*

O X-Med (Lino, 2007) é um jogo educacional que simula a realização de um programa de medição de software voltada para a monitoração de projetos alinhado ao nível 2 de maturidade do CMMI-DEV v1.2 com base no GQM - *Goal/Question/Metric* (Basili et al, 1994) e integrando elementos do PSM – *Practical Software and Systems Measurement* (McGarry et. al., 2001). O objetivo de aprendizagem do jogo é reforçar conceitos de medição e ensinar a competência de aplicar conhecimento de medição.

Através da seleção de soluções adequadas a tarefas de medição, alunos são incentivados a aprender como desenvolver ou selecionar objetivos de medição, planos GQM, etc. A versão 1.0 do jogo segue um fluxo narrativo linear em que o aluno assume o papel de um analista de medição e segue seqüencialmente todas as tarefas de definir e aplicar um programa de medição. Em cada passo, uma tarefa é apresentada ao aluno. Por exemplo, no passo 2, o aluno é requisitado a identificar o objetivo de medição mais apropriado para a situação dada (Figura 4). Para isto, o jogo apresenta instruções, material (descrição do produto, registros de entrevistas, etc.). Para cada tarefa, apresenta seis alternativas como solução. De acordo com a alternativa selecionada, uma pontuação e um *feedback* são imediatamente fornecidos. No final, a

pontuação total é calculada somando as pontuações parciais e é gerado um relatório final.

O público alvo do jogo são estudantes de pós-graduação na área de Ciência da Computação ou profissionais de Engenharia de Software. O jogo é concebido como um complemento a cursos tradicionais ou *e-learning* fornecendo um ambiente para o aluno exercitar os conceitos apresentados. O jogo requer uma compreensão básica de Engenharia de Software, medição de software, gerência de projetos e do *framework* CMMI. O jogo é concebido para ser utilizado individualmente sem necessidade de interação com outros alunos ou instrutores. A duração de uma sessão do jogo é aproximadamente 2 horas. A versão atual do jogo foi desenvolvida em JAVA JDK 6 como sistema desktop. Esta versão consiste somente de um único cenário estático focando em medição para gerência de projetos.

Figura 13. Exemplos de telas do X-MED v1.0

O jogo X-MED versão 1.0 foi avaliado por meio de uma pesquisa exploratória analisando a sua efetividade como uma ferramenta educacional. Neste contexto foi realizada uma série de experimentos como parte de disciplinas de mestrado em 2007, envolvendo um total de 15 alunos. A primeira hipótese de pesquisa era que o X-MED versão 1.0 tem um efeito de aprendizagem positivo na capacidade do aluno em definir e executar programas de medição. Um segundo objetivo de pesquisa era a avaliação do jogo em termos de relevância, suficiência, grau de dificuldade e seqüência, método de ensino e duração, além de identificar principais pontos fortes e fracos para guiar a sua evolução. Neste estudo, um impacto no efeito de aprendizagem não pode ser demonstrado estatisticamente, mas avaliações subjetivas pelos alunos indicam o potencial do jogo para apoiar o ensino. Com base nos resultados desta avaliação, o jogo está atualmente sendo evoluído, considerando principalmente a sua dinâmica e fluxo, além de melhorar o *design* do jogo, com a integração de elementos multimídia.

4.4 Discussão

Os jogos apresentados neste artigo surgiram a partir de uma necessidade e de uma oportunidade. Necessidade de adotar métodos alternativos para o ensino de GPS, e oportunidade de utilizá-los em disciplinas de graduação e pós-graduação na PUCRS e na UNIVALI. A idéia em si não é nova, pois outros jogos vêm sendo desenvolvidos. Mas tem sido difícil comparar estes jogos com outros já desenvolvidos, devido a pouca informação disponibilizada dos mesmos. A tabela 1 apresenta uma comparação inicial dos três jogos apresentados, a partir de características gerais.

Tabela 2. Comparação dos jogos desenvolvidos

Critérios	Planager	Scrumming	X-MED
Usuários	Um jogador	Um jogador	Um jogador
Tipo de jogo	Simulação (fluxo linear)	Simulação	Simulação (fluxo linear)
Tópico	Escopo e tempo	SCRUM	Medição e análise
Projetos	Cenários (projetos)	Um projeto	Um projeto
Ambiente	Desktop	Desktop	Desktop
Foco	Graduação	Graduação e Indústria	Pós-graduação e Indústria
Interface	Textual	Principalmente textual	Textual e gráfica
Tecnologia	JAVA	JAVA	JAVA

Esta comparação também confirma o quadro demonstrado por outros jogos existentes. A grande maioria dos jogos na área de GP se enquadra na categoria de jogos de simulação com graus variados de dinâmica e flexibilidade. Dominar a flexibilização de cenários nos jogos se demonstrou altamente complexo, especialmente, quando considerado o grau da subjetividade na avaliação das soluções geradas pelo jogador. Muitos jogos existentes são sistemas *desktop*, sendo que mais recentemente têm sido criados mais jogos para *web* ou até no contexto de *microworlds*.

A principal motivação por trás do desenvolvimento destes jogos é o uso em temas específicos dentro da grande área de gerência de projetos, o uso constante no ensino de GPS, e a preocupação com a avaliação do uso dos mesmos, seja através de experimentos ou da prática. Desta forma, o que se pretende é avaliar com o tempo se jogos podem ser úteis para ensinar GPS, e quais as subáreas da GPS que mais podem se beneficiar do uso de jogos como método de ensino complementar.

Mesmo que a maioria dos jogos desenvolvidos até então se concentrem em uma das subáreas de GPS voltados, principalmente, ao nível cognitivo, entende-se que existe também uma gama ampla de jogos voltados para o ensino de "*soft skills*" na gerência em geral, mas raramente voltados para projetos de software, sendo esta também uma oportunidade a ser explorada no futuro.

5 Conclusão

Neste artigo apresentamos três jogos representando alternativas inovadoras para o ensino de temas específicos na GPS, complementando o estado da arte de jogos nesta área. Considerando que PUCRS e UNIVALI estão iniciando a pesquisa nesta área, o nosso principal foco foi na concepção e desenvolvimento dos jogos. Como passo futuro planeja-se a avaliação do impacto dos jogos na aprendizagem. Além disso, pretende-se realizar uma avaliação cruzada do uso dos jogos tanto na PUCRS como na UNIVALI e disponibilizar os jogos de forma mais ampla para a comunidade acadêmica e para a indústria. Finalmente, outros jogos estão sendo planejados, a partir da observação de necessidades de treinamentos em outros temas da gerência de projetos.

Referências

Basili, V., Caldiera, G., Rombach, D. Goal/Question/Metric Approach. In: J. Marciniak (ed.), **Encyclopedia of Software Engineering**, Vol 1. John Wiley & Sons, 1994.

Betz, J. A. Computer Games: Increases learning in an interactive multidisciplinary environment. Journal of Educational Technology Systems, n. 24, p. 195-205, 1995.

Dantas, A., Barros, M., Werner, C. A Simulation-Based Game for Project Management Experiential Learning. In: 16a INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING & KNOWLEDGE ENGINEERING (SEKE), Banff, Canadá, 2004.

Isotton, E. ***Scrumming* – Ferramenta Educacional para Apoio ao Ensino de Práticas de SCRUM**. Monografia de Trabalho de Conclusão de Curso de Graduação, Sistemas de Informação, FACIN, PUCRS, Porto Alegre, 2008.

Kieling, E., Rosa, R. ***Planager* - Um Jogo para Apoio ao Ensino de Conceitos de Gerência de Projetos de Software**. Monografia de Trabalho de Conclusão de Curso de Graduação, Ciência da Computação, FACIN, PUCRS, Porto Alegre, 2006.

Gee, J. P. Learning by design: Games as learning machines. In: Anais da GAME DEVELOPERS CONFERENCE. San Jose, CA, 2004.

Gramigma, M. R. M. **Jogos de empresa**. São Paulo: Makron Books, 1994.

Gurgel, I., Arcoverde, R. L., Almeida, E. W. M., Sultanum, N. B., Tedesco, P. A Importância de Avaliar a Usabilidade dos Jogos: A Experiência do Virtual Team. In: Anais do SBGames, Recife, 2006.

Lino, J. I. **Proposta de um Jogo Educacional para Medição e Análise de Software**. Trabalho de Conclusão de Curso, Sistemas de Informação, Universidade Federal de Santa Catarina, Florianópolis, 2007.

McGarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean, J., Hall, F. **Practical Software Measurement: Objective Information for Decision Makers**. Addison-Wesley Professional, 2001.

Neal, L. Implications of computer games for system design. In: Anais da INTERACT, North Holl and, 1990.

Oh Navarro, E., van der Hoek, A. SimSE: An Interactive Simulation Game for Software Engineering Education. In: Anais da 7th. INTERNATIONAL CONFERENCE ON COMPUTERS AND ADVANCED TECHNOLOGY IN EDUCATION, Havaí, 2004.

PMBOK. Project Management Institute - PMI: A guide to the project management body of knowledge, Syba: PMI Publishing Division, 2004.

Prikladnicki, R., Rosa, R., Kieling, E. Ensino de Gerência de Projetos de Software com o *Planager*, In: XVIII SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), São Paulo, 2007.

Reif, H. L., Mitri, M. How University Professors Teach Project Management for Information Systems. Communications of the ACM, Vol. 48, N. 8, Ago/2005.

Ribeiro, L. O. M., Timm, M. I., Zaro, M. A. Modificações em Jogos Digitais e seu uso Potencial como tecnologia Educacional para o Ensino de Engenharia. Cinted-UFRGS, Vol. 4, N. 1, Julho, 2006.

Schwaber, K. **Agile Project Management with Scrum**. Microsoft Press, 2004.

Relato de Experiência de Ensino de Modelagem e Implementação de Software em um Curso de Graduação em Ciência da Computação

Heitor Augustus Xavier Costa[1] Antônio Maria Pereira de Resende[2]
Fábio Fagundes Silveira[3]

[1, 2] Grupo de Pesquisa em Engenharia de Software (PqES) – Departamento de Ciência da Computação (DCC) – Universidade Federal de Lavras (UFLA)
Caixa Postal 3037 – CEP 37.200-000 – Lavras – MG – Brasil
[3] Departamento de Ciência e Tecnologia (DCT) – Universidade Federal de São Paulo (UNIFESP) – CEP 12231-280 – São José dos Campos – SP – Brasil

[1]heitor@ufla.br, [2]tonio@dcc.ufla.br, [3]fsilveira@unifesp.br

Abstract. This paper presents an experience report based on the teaching of six periods of the discipline Modeling and Implementing of Software offered in a course of Computer Science. This discipline was organized in three parts: conceptual approach, concepts fixation, and concepts application. The data presented in this article were obtained in the seminars, case studies, and way of evaluation. The reached results were important and relevant and they can be verified in the presented graphs. Especially, the last period of the discipline presented interesting result, credited in the verification of two factors: students' maturity and professor's maturity.

Keywords: Teaching of Software Modeling, Education in Informatics, Software Engineering

Resumo. Este artigo apresenta o relato de experiência baseada no ensino de seis semestres letivos consecutivos da disciplina Modelagem e Implementação de Software oferecida em um curso de graduação em Ciência da Computação. A ministração desta disciplina foi organizada em três partes: abordagem conceitual, fixação dos conceitos e aplicação dos conceitos. Os dados apresentados neste artigo foram obtidos dos seminários e dos estudos de caso realizados pelos alunos e da forma de avaliação dos alunos. Os resultados alcançados foram relevantes e podem ser constatados nos gráficos apresentados. Em especial, o último semestre letivo de oferta da disciplina apresentou resultado interessante, creditado na constatação de dois fatores: maturidade dos alunos e maturidade do professor.

Palavras-chave: Ensino de Modelagem de Software, Educação em Informática, Engenharia de Software

1 Introdução

A modelagem é uma das principais atividades no desenvolvimento de software, sendo ela uma ponte que visa encurtar a distância da abstração e entendimento da lógica de negócio entre os *stakeholders* e os profissionais da engenharia de software. Alguns cursos de graduação em Ciência da Computação têm abordado o tema Modelagem de Software na disciplina de Engenharia de Software, sendo atribuído curto tempo para realizar de maneira adequada a apresentação deste tópico, visto que há muitos temas para serem abordados.

Mesmo não estando explícito no currículo de referência proposto pela Sociedade Brasileira de Computação (SBC) [SBC, 2005], pode-se perceber, no item Detalhamento das Disciplinas, no tópico Tecnologia da Computação, disciplina Engenharia de Software, a presença do tema Modelagem de Software disseminada nos itens apresentados/sugeridos. Este currículo é usado como base para a avaliação dos cursos da área de Informática e Computação, podendo nortear as Diretrizes Curriculares desta área. Desta forma, mostra-se a relevância do tema.

O objetivo deste artigo é apresentar o relato de experiência baseada no ensino de seis semestres letivos consecutivos da disciplina Modelagem e Implementação de Software (MIS) oferecida no curso de graduação em Ciência da Computação pelo Departamento de Ciência da Computação da Universidade Federal de Lavras (DCC/UFLA), Lavras/MG. Os dados apresentados são relativos aos anos de 2005 a 2007, período em que esta disciplina foi ofertada como eletiva[3] aos alunos.

O artigo está organizado da seguinte forma: a seção 2 apresenta os objetivos almejados com a oferta da disciplina MIS; a seção 3 mostra a organização da ministração da disciplina em três partes; a seção 4 resume a metodologia utilizada na disciplina; a seção 5 organiza os dados coletados durante a oferta da disciplina; a seção 6 relaciona alguns trabalhos relacionados; e a seção 7 apresenta conclusões e sugestões de trabalhos futuros.

2 A Disciplina Modelagem e Implementação de Software no Curso de Ciência da Computação

A disciplina MIS ministrada no curso de graduação em Ciência da Computação da UFLA teve os seguintes objetivos:

- Oferecer ao aluno visão geral da Engenharia de Software, destacando a importância de realizar a modelagem de um software para posterior entendimento e evolução e contextualizando, no processo de desenvolvimento de software, o trabalho a ser desenvolvido por ele;
- Habilitar o aluno na concepção de um software em camadas, de tal forma que ele seja capaz de estruturar o software em, pelo menos, três camadas: i) Interface Homem-Máquina; ii) Lógica de Negócio; e iii) Acesso a Dados. Além destas camadas, é apresentada a camada Sistema;
- Capacitar o aluno na realização do mapeamento de dados entre o modelo orientado a objetos e o modelo relacional. O primeiro é utilizado como foco da disciplina e o segundo é o modelo de banco de dados mais encontrado no mercado [Costa, 2000];

3 Tem por finalidade complementar a formação do estudante, na área de conhecimento do curso, escolhida entre as definidas para esse e de forma a integralizar uma carga horária mínima estabelecida pelo Colegiado de Curso [PRG, 2007].

- Treinar o aluno para modelar software orientado a objetos usando UML (*Unified Modeling Language*), visto que é padrão para orientação a objetos [Jacobson *et al*., 1999; Jacobson *et al*., 2005];
- Conscientizar o aluno que a característica de qualidade de manutenção deve ter atenção desde o início do processo de desenvolvimento do software;
- Fornecer ao aluno dicas para implementar um software usando a linguagem de programação Java e visando a facilidade de manutenção.

Esta disciplina foi idealizada mediante o resultado da tese de doutorado de [Costa, 2005]. Ela possui carga horária de 68 horas distribuída em 17 semanas com 4 horas por semana. Desta 4 horas, 2 horas são destinadas para teoria (sala de aula) e 2 horas são destinadas para prática (laboratório de computadores). Os alunos desta disciplina estavam entre o 4º e o 8º período.

3 Etapas de Trabalho

A ministração da disciplina MIS foi organizada em três partes:

1. **Abordagem Conceitual**. A importância de realizar a modelagem de software, atentando à manutenibilidade do software, a caracterização de construção de software em camadas, o mapeamento de dados entre os modelos orientados a objetos e relacional e a apresentação dos diagramas UML foram abordados através de aulas expositivas, usando quadro negro e retro-projetor;
2. **Fixação dos Conceitos**. À medida que o conteúdo conceitual foi apresentado, uma modelagem de um software hipotético foi desenvolvida de modo a fixar os conceitos apresentados. O objetivo foi aflorar e solucionar dúvidas e questionamentos;
3. **Aplicação dos Conceitos**. Para os alunos colocarem em prática, de maneira efetiva, os conceitos abordados, eles foram a campo, visitando empresas, órgãos da universidade, comércio, entre outros locais (os alunos também podiam usar parte do seu trabalho de conclusão de curso, quando ele envolvia o desenvolvimento de um software), buscar a necessidade destes locais em construir um software da categoria de sistema de informação para atender alguma parte da sua lógica de negócio. Assim sendo, este sistema de informação tornou-se um estudo de caso real onde o objetivo foi o aluno realizar a sua modelagem e a sua implementação. Para isso, foram utilizadas aplicações livres: a ferramenta CASE (*Computer-Aided Software Engineering*) Jude[4] para fazer parte da modelagem e a IDE (*Integrated Development Environment*) Netbeans[5] para realizar a programação.

4 Metodologia de Trabalho

O início dos trabalhos da disciplina MIS foi através de aulas expositivas do conteúdo teórico previsto na sua ementa [Ementa, 2008]. O material utilizado foi quadro negro e transparências em retro-projetor.

No primeiro dia de aula, os alunos foram apresentados a ementa e informados da existência de um trabalho prático, consistindo no desenvolvimento de um software real. Para isso, eles foram a campo buscar um problema para resolver através da construção de um software da categoria de sistema de informação; eles tiveram quinze

4 http://jude.change-vision.com/jude-web/index.html.

5 http://www.netbeans.org/.

dias para apresentar o problema. Enquanto isso, a disciplina transcorreu normalmente. Após este prazo, os alunos entregaram um enunciado do problema para entendimento e acompanhamento dos trabalhos e delimitação do escopo, caso necessário.

Nas aulas seguintes, foi abordada a importância da Engenharia de Software, bem como a relevância em realizar a modelagem de software para compor a documentação necessária para apoiar a evolução do software.

Em seguida, foram apresentados diversos modelos de processo de desenvolvimento de software, entre eles: cascata, iterativo, incremental, modelo em V, prototipação e processo unificado. Assim sendo, o aluno tinha conhecimento das etapas envolvidas no desenvolvimento de um software, ou seja, o que consiste cada etapa, quais os artefatos de software gerados e a continuidade/ligação entre as etapas. O enfoque é apenas para contextualizar e dar subsídios aos alunos para caminharem de maneira mais consistente na disciplina. Foi colocado em sala de aula que estes assuntos são mais detalhadamente abordados na disciplina Engenharia de Software, prevista na estrutura curricular do curso [Estrutura Curricular, 2008].

Na apresentação do assunto desenvolvimento de software em camadas, inicialmente, foi contextualizado o que seria um software com duas camadas (Interface Homem-Máquina e Lógica de Negócio), apresentando o que eles faziam quando desenvolviam trabalhos das disciplinas iniciais de programação, ou seja, a entrada de dados, a manipulação destes dados e a apresentação dos resultados. Em seguida, foi acrescida uma camada (Acesso a Dados), argumentando a eles a necessidade de dar continuidade ao trabalho, ou seja, não perder as informações geradas durante a execução do software, armazenando em um meio persistente. Para finalizar, foi apresentada mais uma camada (Sistema) que consiste na comunicação do software com a funcionalidade oferecida pelo sistema operacional.

Por último, foram abordados os diagramas UML. Para cada um deles, foram apresentados a sua importância, os seus elementos de modelagem e uma breve descrição de como realizar a sua construção. À medida que os diagramas foram apresentados, a modelagem de um software-exemplo foi conduzida e os alunos foram construindo a modelagem do seu trabalho prático (organizados em um ou dois alunos). O trabalho prático teve o objetivo de causar dúvidas e questionamentos nos alunos quando na construção dos diagramas (quando eles põem a "mão na massa"), pois a simples apresentação do exemplo direcionado e explicativo não alcança este resultado.

Alcançando este resultado, foram previstas aulas que os ajudavam na resolução destas dúvidas e questionamentos. Isso foi feito através de seminários de apresentação do trabalho prático, onde o aluno expunha suas dúvidas para classe; primeiramente, ele apresentava seu problema contextualizando a turma e, em seguida, apresentava sua dificuldade, para debates, opiniões e sugestões. Este procedimento foi realizado em duas etapas: i) uma para o Modelo de Análise; e ii) outra para o Modelo de Projeto. Nestas aulas, houve também a solidificação do discernimento entre os dois modelos. Cabe ressalta que, em todas as turmas, sem exceção, houve alto grau de participação, proporcionando aos alunos uma experiência enriquecedora. Nestas discussões, o professor da disciplina atuou apenas como um moderador da discussão.

A condução da avaliação dos alunos não teve como objetivo a obtenção de nota para passar, embora seja isso que boa parte dos alunos querem (independente da disciplina cursada), mas o aprendizado do aluno. Assim, o procedimento de avaliação adotado foi em etapas incrementais: i) Apresentação do Modelo de Análise – para esclarecer dúvidas; ii) Entrega do Trabalho 1 – para correção; iii) Devolução do Trabalho 1 – para

fazer comentários do trabalho, depois de corrigido; iv) Apresentação do Modelo de Projeto – para esclarecer dúvidas; v) Entrega do Trabalho 2 – para correção; vi) Devolução do Trabalho 2 – para fazer comentários do trabalho, depois de corrigido; e vii) Entrega do Trabalho 3 – para correção. A Tabela 3 apresenta relação dos artefatos de software a serem gerados nos trabalhos.

Tabela 3 – Relação dos Artefatos de Software para os Trabalhos

Trabalho	Artefatos de Software
Trabalho 1	Enunciado do Problema + Modelo de Análise (Diagrama de Casos de Uso (completo), Descrição dos Casos de Uso (Manter Cadastro + Gerar Informações + Relatório) + Diagrama de Classes (Completo) + Dicionário de Atributos e Métodos (para as classes envolvidas nos casos de uso escolhidos) + Diagrama de Seqüência + Diagrama de Estados + Diagrama de Atividades) → Versão Impressa.
Trabalho 2	Enunciado do Problema Melhorado + Modelo de Análise Melhorado + Modelo de Projeto (Diagrama de Classes (Completo) + Dicionário de Atributos e Métodos (para as classes envolvidas nos casos de uso escolhidos) + Diagrama de Seqüência + Diagrama de Estados + Diagrama de Atividades) + Modelo de Dados (Esquema das Tabelas) → Versão Impressa.
Trabalho 3	Enunciado do Problema Melhorado + Modelo de Análise Melhorado + Modelo de Projeto Melhorado + Modelo de Dados Melhorado + Software → Versão Eletrônica.

Após a apresentação das suas dificuldades, os alunos complementavam seus artefatos de software e os entregava para correção. Após a correção, os trabalhos eram repassados aos alunos apontando os pontos fracos do trabalho para melhorar e entregar novamente. Assim, o Modelo de Análise foi avaliado três vezes, o Modelo de Projeto foi avaliado duas vezes e a implementação foi avaliada uma vez.

A funcionalidade desenvolvida para o software seguiu a proposta de [Costa, 2005] que coloca que os casos de uso de um sistema de informação podem ser organizados em três categorias básicas: i) manutenção de cadastros (inclusão, exclusão, consulta e alteração); ii) geração de informações (combinar dados armazenados); e iii) geração de relatórios (na tela e impressos). Assim, tendo em vista que o desenvolvimento do software completo poderia levar mais de um semestre letivo, optou-se pelo desenvolvimento de um caso de uso de cada categoria.

5 Dados Coletados

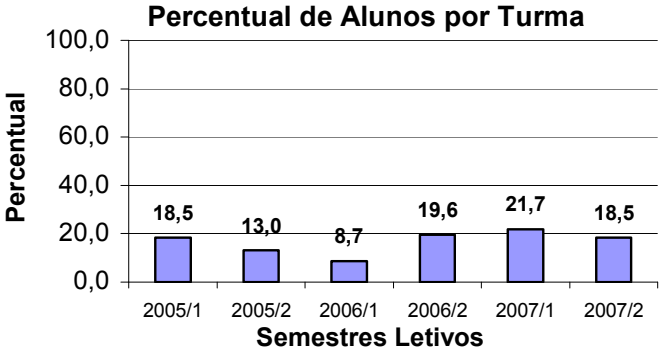
A seguir, são apresentados vários dados coletados a partir dos seminários e estudos de caso realizados pelos alunos e da forma de avaliação dos alunos.

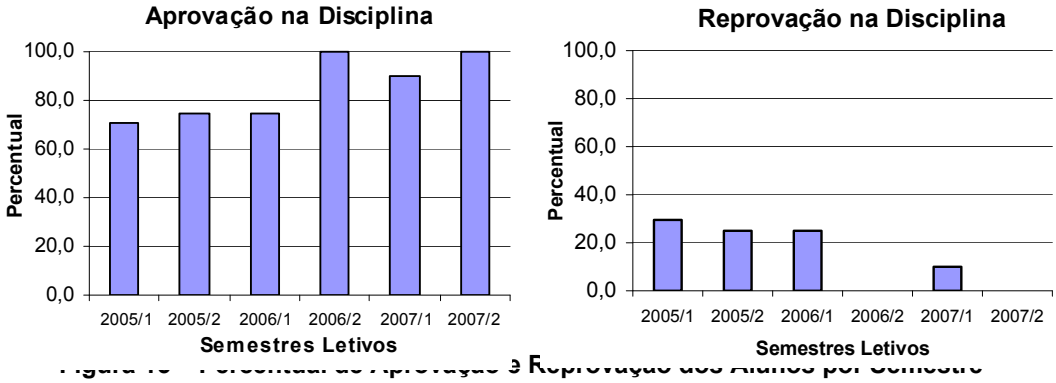
A Figura 14 apresenta a quantidade de alunos por turma. Pode-se notar que, no início, houve grande procura pela disciplina por ser novidade (os dois primeiros semestres); contudo, no semestre seguinte, a demanda caiu. Os três últimos semestres a procura aumentou, pois o grupo de Pesquisa em Engenharia de Software (PqES) do DCC/UFLA foi consolidado na universidade, surgindo vários projetos na área. Durante os três anos de oferta, esta disciplina teve o total de 91 alunos, ou seja, aproximadamente 45% de alunos do curso de graduação em Ciência da Computação da UFLA, considerando a entrada de 25 alunos/semestre e o tempo médio de formação do aluno de quatro anos (200 alunos).

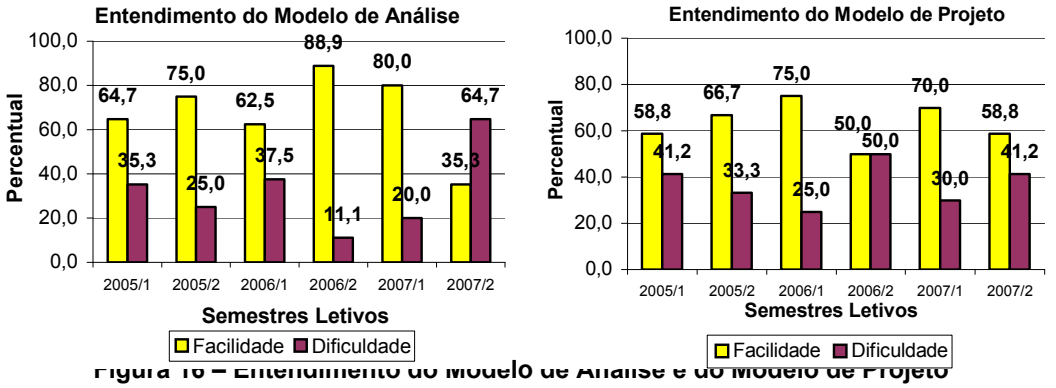
A Figura 15 apresenta o percentual de alunos aprovados e reprovados. Pode-se notar que o índice de aprovação foi relativamente bom; contudo, com a consolidação do grupo PqES, os alunos do grupo cursaram a disciplina e o índice de aprovação

aumentou consideravelmente (mais de 90%). No total, o percentual de aprovação foi 86,96% e o percentual de reprovação foi 13,04%.

A Figura 16 apresenta o percentual de alunos com facilidade e com dificuldade no entendimento no Modelo de Análise e no Modelo de Projeto. Fato interessante apresentado por esta figura ocorre em 2007/2, onde a turma deste semestre letivo apresentou relativa dificuldade no entendimento do Modelo de Análise, enquanto as turmas dos semestres letivos anteriores não tinham tamanha dificuldade. Estas turmas possuíram números relativamente expressivos onde elas conseguiram mais de 62% de entendimento do Modelo de Análise. Acompanhando a dificuldade da turma de 2007/2, pode-se notar dificuldade no entendimento do Modelo de Projeto (índice relativamente alto em comparação com as demais turmas).



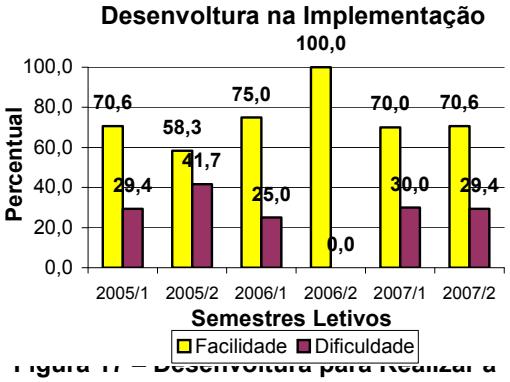




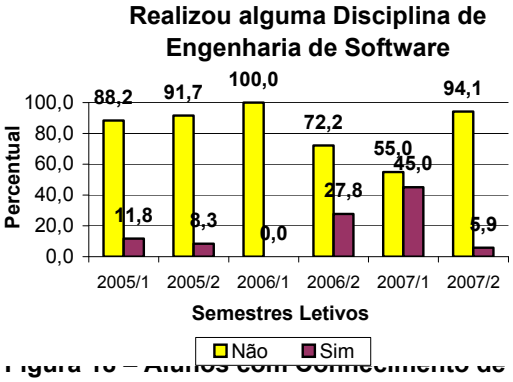
A Figura 17 apresenta o percentual de alunos com facilidade e com dificuldade para realizar a implementação do software, seguindo o Modelo de Projeto e utilizando a linguagem de programação Java. Pode-se perceber que, durante a apresentação do software para o professor (explicação da implementação), os alunos tiveram

maturidade para ler e interpretar a documentação gerada durante o curso. Além disso, foi sensível o amplo conhecimento dos alunos na linguagem de programação adotada. Considerando os seis semestres letivos de oferta da disciplina, em torno de 75% lograram êxito na implementação.

A Figura 18 apresenta o percentual de alunos que cursou alguma disciplina na área de Engenharia de Software. Nesta figura, pode-se perceber que a maioria dos alunos não teve contato algum com conteúdo da área de Engenharia de Software, contudo o índice de aprovação foi muito bom ao final da disciplina, levando a crer que a disciplina foi importante para a formação do aluno. Considerando o semestre letivo 2007/2, onde 94,1% dos alunos não tinham visto algo sobre Engenharia de Software, o índice de aprovação foi de 100%.



Implementação do Software



Engenharia de Software

A Figura 19 apresenta a média final das turmas. Pode-se notar que as médias não ficaram abaixo da média de aprovação determinada pela universidade (60 pontos). Assim sendo, mesmo com as dificuldades encontradas e relatadas neste trabalho, acredita-se que os alunos tenham aproveitado de maneira adequada a disciplina. Cabe considerar o semestre letivo 2007/2, onde foi detectado que 74% dos alunos não tinham visto algo sobre Engenharia de Software e o índice de aprovação foi de 100%, a média da turma foi a segunda maior com 85,71 pontos.

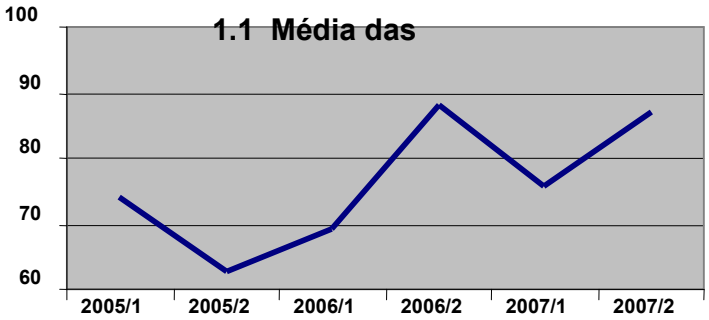


Figura 19 – Média Final das Turmas

6 Trabalhos Relacionados

[Soares, 2004] descreve uma experiência de ensino da disciplina Engenharia de Software em um curso de graduação em Ciência da Computação. A proposta foi mostrar como a disciplina transcorreu, considerando uma abordagem de proposta de

trabalhos práticos em equipes, como foi realizada a avaliação e como essa abordagem motivou os alunos.

[Silva *et al*., 2004] relata a experiência adquirida com um método de ensino aplicado à disciplina Princípio de Engenharia de Software. O objetivo deste método foi despertar o interesse dos alunos para Engenharia de Software, apresentando aspectos teóricos e oferecendo experiência prática dos conceitos. Para isso, os autores adotaram *eXtreme Programming*. [Martins, 2002] apresenta um relato da experiência de ensino de Engenharia de Requisitos em um curso de Mestrado. Para isso, o autor apresentou alguns indicativos sobre a área de Engenharia de Requisitos que apareceram durante a ministração da disciplina. Estes indicativos foram coletados em seminários e estudos de caso pelos alunos e da aplicação de um questionário no final da disciplina.

[Tomayko, 1987] apresenta um guia ao professor de cursos introdutórios em engenharia de software, contendo um estudo de caso de um curso baseado em um projeto grande. Para isso, o autor distingue quatro modos em que esta disciplina pode ser ministrada: i) aulas expositivas; ii) aulas expositivas e seminários; iii) aulas expositivas e estudos de casos com pequenas equipes; e iv) aulas expositivas e estudos de casos com grandes equipes. [Hazzan e Dubinsky, 2003] enfocam no ensino de metodologias de desenvolvimento de software, apresentando dez princípios de ensinar cada metodologia e examinando cada um do ponto de vista organizacional e pedagógico. Os princípios pedagógicos são demonstrados usando a metodologia de Programação Extrema.

Estes trabalhos relacionados apresentam grande preocupação em ensinar conceitos aliados a atividades práticas. Embora estas idéias também tenham norteado o desenvolvimento deste trabalho, o seu diferencial é no seu escopo, pois este trabalho tem o foco estritamente no tema modelagem e implementação de software.

7 Conclusões

Foi constatado que resultados relevantes foram alcançados na aprendizagem e na prática da modelagem de software, visto que o índice de aprovação foi alto nos seis semestres em que a disciplina foi oferecida.

A forma de condução de avaliação dos alunos contribuiu para seu aprendizado, o que pode ser fortemente notado no semestre letivo 2007/2. Este semestre letivo, em especial, é considerado um semestre de grande sucesso; acredita-se que seja em decorrência de dois fatores: i) maturidade dos alunos na escolha da disciplina; e ii) maturidade do professor, pois a lecionou por cinco semestres anteriores consecutivos.

Podem ser considerados os seguintes passos para dar continuidade a este trabalho: i) conscientizar o colegiado do curso da importância desta disciplina e, assim, colocá-la na estrutura curricular; ii) redistribuir o trabalho entre os alunos após a construção dos modelos; e iii) abordar outros tópicos da Engenharia de Software. Além disso, técnicas de *reviews* podem ser empregadas, ou seja, um grupo de alunos tenta entender a documentação gerada por um outro grupo para realizar a implementação.

Bibliografia

Costa, H. A. X. (2000) BDOO: Uma Visão Geral. Infocomp – Journl of Computer Science. v. 2. n. 1 p.27-32.

Costa, H. A. X. (2005) Critérios e Diretrizes de Manutenibilidade para a Construção do Modelo de Projeto Orientado a Objetos. 199 p. Tese de Doutorado. Escola Politécnica da Universidade de São Paulo.

Ementa (2008) Ementa da Disciplina Modelagem e Implementação de Software. Acessado em: agosto/2008. Localização: <http://www.comp.ufla.br/curso/ementas/com203.pdf>.

Estrutura Curricular (2008) Estrutura Curricular do Curso de Graduação em Ciência da Computação da Universidade Federal de Lavras. Acessado em: Agosto/2008. Localização: <http://www.prg.ufla.br/curri_pleno/2008-2/Ciência%20da%20Computação.pdf>.

Hazzan, O.; Dubinsky, Y. (2003) Teaching a Software Development Methodology: The Case of Extreme Programming. Proceedings of the 16th Conference on Software Engineering Education and Training (CSEE&T 2003), Espanha, pp. 176-184.

Jacobson, I., Booch, G., Rumbaugh, J. (1999) The Unified Modeling Language User Guide. Addison Wesley.

Jacobson, I., Booch, G., Rumbaugh, J. (2005) Unified Modeling User Guide. Addison Wesley.

Martins, L. E. G. (2002) Relato de Experiência de Ensino de Engenharia de Requisitos em um Curso de Mestrado em Sistemas de Informação. Workshop on Requirements Enginnering. Valência, Espanha.

PRG (2007) Resolução CEPE Nº 042 de 21 de março de 2007 – Estabelece normas gerais do Ensino de Graduação da UFLA. Localização: <http://www.prg.ufla.br/legislacao.htm>.

SBC (2005) Currículo de Referência da SBC para Cursos de Graduação em Bacharelado em Ciência da Computação e Engenharia de Computação – Proposta versão 2005. Localização: <http://www.sbc.org.br/index.php?language=1&content=downloads&id=198>.

Silva, L. F. da; Leite, J. C. S. do P.; Breitman, K. K. (2004) Ensino de Engenharia de Software: Relato de Experiências. Anais do Workshop de Educação em Informática. p.994-1005, Salvador.

Soares, M dos S. (2004) Uma Experiência de Ensino de Engenharia de Software Orientada a Trabalhos Práticos. Anais do Workshop sobre Educação em Informática da Região RJ/ES.

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

Utilizando Experimentação para Apoiar a Pesquisa em Educação em Engenharia de Software no Brasil

Rodrigo Pereira dos Santos, Paulo Ségio Medeiros dos Santos,
Cláudia Maria Lima Werner, Guilherme Horta Travassos

Programa de Engenharia de Sistemas e Computação (PESC)
COPPE/UFRJ, Universidade Federal do Rio de Janeiro, Brasil
Caixa Postal 68511 – CEP 21945-970 – Rio de Janeiro, RJ

{rps, pasemes, werner, ght}@cos.ufrj.br

Abstract. The Software Engineering (SE) research community has an important role in the process of human resources formation for the area. Efforts to improve SE education have been made by the community. However, obtained results often remain isolated and localized, being necessary to make a diagnosis and to put together what are the main problems, solutions and challenges of SE education, as well as the national scenario peculiarities. This paper aims to present a proposal of a strategy to support large scale research in SE education based in experimentation that contributes to the organization of a body of knowledge regarding SE education in Brazil.

Keywords: Software Engineering Education; Experimentation in Computer Science; Systematic Review; Large Scale Collaborative Research.

Resumo. A comunidade de pesquisa em Engenharia de Software (ES) tem um papel significativo no processo da formação dos recursos humanos da área. Esforços para melhorar a educação em ES têm sido realizados pela comunidade. No entanto, os resultados alcançados muitas vezes permanecem isolados e localizados, tornando-se necessário diagnosticar e reunir quais são os principais problemas, soluções e desafios de educação em ES, além das peculiaridades do cenário nacional. Este trabalho visa apresentar uma proposta de estratégia para apoiar a pesquisa em larga escala em educação em ES, baseada em experimentação, que contribua para a organização de um corpo de conhecimento em educação em ES no Brasil.

Palavras-chave: Educação em Engenharia de Software; Experimentação em Ciência da Computação; Revisão Sistemática; Pesquisa Colaborativa em Larga Escala.

---

* Este trabalho foi desenvolvido com apoio do CNPq, FAPERJ e CAPES.

1 Introdução

O processo de ensino e aprendizagem de ES tem passado por questionamentos [Hilburn & Towhidnejad, 2007]. A academia usualmente ensina conceitos e fundamentos de ES através de disciplinas teóricas, com aulas expositivas e leituras complementares, que contemplam a competência prática mediante o desenvolvimento de um pequeno projeto em um curto espaço de tempo [Huang & Distante, 2006]. Assim, quando expostos à indústria, os recém-formados encontram um cenário no qual técnicas e métodos aprendidos são pouco aplicados [Nauman & Uzair, 2007]. Muitas vezes, vêem-se obrigados a utilizar práticas *ad hoc*, aprendidas com a experiência profissional, que podem impactar negativamente o estabelecimento de princípios e boas práticas de engenharia no processo de desenvolvimento de software. Segundo Meyer (2001), a academia não deve assumir toda essa responsabilidade, pois a universidade não é uma empresa, mas precisa preparar os estudantes para os reais desafios. Há um consenso de que o ensino de ES, tradicional e focado em metodologias, deve ser transformado para refletir a demanda por software mais complexo [Baker *et al.*, 2005]. Frente ao volume de documentos e processos para um bom projeto de software, estudantes se desinteressam pela disciplina de ES e acham-na teórica e burocrática, preferindo escrever programas e vê-los funcionando a documentar formalmente os seus sistemas [De Lucena *et al.*, 2006].

Essas e outras questões fazem parte dos grandes desafios de educação em Computação para os próximos dez anos, conforme apresentado em [McGettrick *et al.*, 2004]. Esse documento destaca que estudantes percebem os cursos na área de Computação como "dominados" pela programação e aponta o desafio de inovação em reconhecer e acomodar as competências e habilidades de estudantes em ES, devido à sua amplitude de pesquisa e de aplicação. Tais considerações também impactam os Grandes Desafios da Pesquisa em Computação no Brasil, pois se concentram na busca de soluções melhor sedimentadas para o processo de formação de recursos humanos para a área. Artigos recentes [SBC, 2006] enfatizam os problemas da falta de qualidade em software e consideram suas implicações sobre a indústria nacional. Deve existir a preocupação com o desenvolvimento de software em equipe, de forma distribuída, dinâmica, com aplicação de engenharia e baseado em evidência ao longo das disciplinas relacionadas a ES. Isso atinge diretamente o desafio *desenvolvimento tecnológico de qualidade*.

Recentemente, a academia despendeu muito esforço em atacar esses desafios, encontrando novas maneiras de ensinar a ES e utilizando diferentes abordagens (e.g., Dantas *et al.*, 2004; Huang & Distante, 2006). Entretanto, se tratado como um conjunto de estratégias isoladas, esse esforço tende a se dissipar, contribuindo para um cenário de ensino e aprendizagem de ES divergente e localizado, e não baseado em evidência e na utilização dos melhores recursos educacionais [Hiebert *et al.*, 2002]. Portanto, identificar mecanismos que permitam a identificação e organização deste conhecimento, colocando-o à disposição da comunidade envolvida em educação em ES, torna-se relevante. Para isso, o paradigma experimental, que envolve a coleção e análise de dados e evidências, pode ser utilizado para caracterizar, avaliar e revelar relacionamentos entre tecnologias, práticas e resultados do processo de ensino e aprendizagem de ES. Com isso, espera-se que os resultados experimentais possam compor um corpo de conhecimento e levar a teorias amplamente aceitas e bem formadas sobre educação em ES.

O presente trabalho explora a experimentação como uma estratégia para apoiar a pesquisa em larga escala sobre educação em ES, por meio do desenvolvimento de um protocolo de investigação científica, a ser acordado e trabalhado cooperativamente pela

comunidade, que guie o processo de construção de um corpo de conhecimento unificado sobre educação em ES no Brasil. O artigo está organizado da seguinte forma: a Seção 2 apresenta uma caracterização inicial baseada numa revisão informal da literatura em bibliotecas digitais (ACM e IEEE) e anais do Workshop sobre Educação em Computação (WEI), com o intuito de identificar aspectos que permeiam o processo de ensino e aprendizagem de ES (problemas, soluções, desafios e cenário nacional). A Seção 3 descreve a estratégia de pesquisa proposta para esta pesquisa. A Seção 4 exemplifica uma proposta de protocolo, baseada nessa estratégia, no contexto do ensino e aprendizagem de ES. Por fim, a Seção 5 apresenta as considerações finais.

2 Caracterização Inicial do Problema

Historicamente, a ES se desenvolveu como uma sub-área da Ciência da Computação (CC), o que reflete o fato das primeiras manifestações de educação em ES (e.g., SE2004 [IEEE/ACM Joint Task Force on Computing Curricula, 2004]) terem ocorrido em cursos inseridos nos programas de CC [Lethbridge *et al.*, 2007]. Soma-se a isso o estabelecimento de programas de graduação e de pós-graduação pelo mundo. Deve-se considerar a existência de alguns problemas no ensino e estabelecimento da ES [Hilburn & Towhidnejad, 2007] [Nauman & Uzair, 2007]: (i) a ES é relativamente nova como disciplina acadêmica; (ii) muitas instituições apresentam carências em prover experiências industriais para os estudantes na prática de ES; (iii) mesmo para disciplinas mais maduras de engenharia, existe o problema de como preparar os estudantes para a prática profissional em um ambiente acadêmico; e (iv) dificuldades para desenvolver e estimular as diversas habilidades necessárias para um engenheiro de software, devido à natureza abrangente e multidisciplinar da ES. Como conseqüência, muitos estudantes concluem seu curso de graduação sem terem participado de um projeto (e processos) de ES próximo ao que encontrarão no mundo real.

O ensino da ES deve preparar os estudantes para participações efetivas em ambientes colaborativos e interdisciplinares. Algumas instituições apresentam estratégias e ferramentas que tentam trabalhar o ensino de ES e tópicos relacionados, por meio de tarefas atrativas de laboratório (e.g.: Huang & Distante, 2006; Figueiredo *et al.*, 2007). Elas têm em comum a utilização de recursos de ensino, às vezes externos ao computador, que desafiam e mobilizam os alunos para tarefas mais próximas de objetivos reais. Mais especificamente em projetos responsáveis pelo desenvolvimento de ferramentas de apoio ao ensino de ES, pode-se citar: *Draw-Bot* [Mohrenschildt & Peters, 1998], *Problems and Programmers* [Baker *et al.*, 2005] e *Projeto Unibral* [De Lucena *et al.*, 2006]. Paralelo a isso, de acordo com Conn (2002), os profissionais de ES estão insatisfeitos com a falta de preparo dos estudantes universitários que ingressam no mercado de trabalho, o que leva a indústria a ter que complementar a sua educação com treinamentos que lhes forneçam o conhecimento necessário para suprir esta deficiência.

Em meio a esses problemas e soluções, surgem desafios para melhorar a educação em ES [Shaw, 2000] [Lethbridge *et al.*, 2007]: (i) tornar os cursos de ES mais atrativos aos estudantes; (ii) focar na educação de ES apropriadamente, entendendo suas dimensões; (iii) apresentar as práticas industriais aos estudantes; (iv) definir padrões de currículo que se preocupem com a evolução da área; (v) prover educação para profissionais da indústria; (vi) tornar a educação em ES baseada em evidência; (vii) assegurar que educadores em ES tenham uma bagagem necessária para essa tarefa; (viii) aumentar o prestígio e a qualidade da pesquisa educacional em ES; (ix) identificar papéis no desenvolvimento de software e prover educação apropriada para cada um;

(x) implantar uma "atitude de engenharia" em cursos de ES; e (xi) estabelecer credenciais que representem precisamente as diferentes habilidades dos profissionais.

No cenário nacional, buscou-se por trabalhos relacionados ao processo de ensino e aprendizagem de ES, no WEI, um evento focado em educação em CC no Brasil. Foram identificados 14 artigos que tratam do ensino de ES nas últimas 4 edições do WEI (2005-2008), dos quais: (i) 9 tratam de metodologias no ensino de ES, contemplando os tópicos gerência de projetos, padrões de projeto, metodologias ágeis, análise e projeto de sistemas, além de aspectos interdisciplinares (e.g., ES com a disciplina Banco de Dados); (ii) 2 discorrem acerca de relatos de experiência; (iii) 2 apresentam ferramentas e/ou jogos para tornar o aprendizado mais lúdico e dinâmico; e (iv) 1 discursa sobre questões relativas ao perfil do profissional de ES. Esses artigos não se aprofundam em problemas e desafios específicos da realidade brasileira de ES; naqueles que descrevem metodologias e relatos de experiência, os resultados apresentados possuem caráter local e não convergem para um ideal maior de se estabelecer um corpo de conhecimento (i.e., guia de referência para educação em ES) e uma comunidade de educadores de ES no Brasil, sobretudo pelo fato de que o WEI é um evento mais geral para a CC.

3 Proposta de Estratégia para a Pesquisa Colaborativa

Com base na revisão informal da literatura (Seção 2), será apresentada uma proposta de pesquisa colaborativa e em larga escala sobre educação em ES focada em experimentação, pois [Perry *et al.*, 2000]: (i) o conhecimento pode ser documentado e codificado mais rapidamente; (ii) as idéias de pesquisas inválidas ou de baixo retorno científico podem ser descartadas; (iii) as áreas de alto retorno científico podem ser reconhecidas rapidamente; e (iv) as questões práticas importantes podem ser consideradas.

A estratégia proposta envolve a execução de estudos secundários (revisão sistemática) e primários (*survey*). A maioria das pesquisas científicas tem seu começo por meio de uma revisão de literatura executada de forma *ad hoc*, conforme a Seção 2. Entretanto, caso esta revisão não esteja completa e justa, terá pouco valor científico. Esta é a principal razão pela qual se deve considerar o uso de uma revisão sistemática da literatura, uns dos meios existentes para identificar, avaliar e interpretar as informações pertinentes a uma questão de pesquisa em particular [Kitchenham, 2004]. Assim, a Figura 20 apresenta a estratégia de pesquisa, definida com base na proposta de Biolchini *et al.* (2007) e previamente utilizada em outros contextos (e.g., Dias Neto, 2008). Essa estratégia é baseada em quatro passos e nos dois tipos de estudos supracitados e será utilizada para construir um corpo de conhecimento sobre educação em ES:

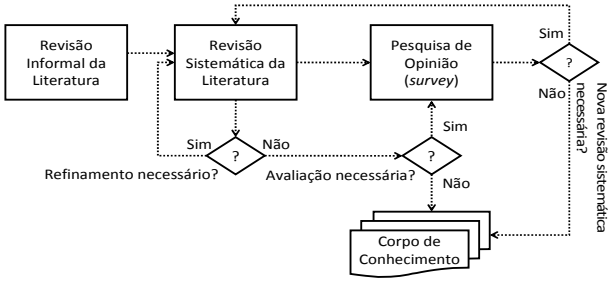


Figura 20 – Estratégia de Pesquisa [Spínola, 2008]

(1) Uma **revisão informal da literatura** é realizada para identificar conceitos básicos que apóiam a definição de um protocolo de revisão sistemática preciso e abrangente;
(2) O **protocolo da revisão sistemática** é elaborado e executado. Baseado nos resultados, o pesquisador define se é necessário refinar o estudo. Caso seja, repete-se este passo. Toma-se ainda uma decisão com relação à necessidade de se avaliar a informação coletada. Se sim, o passo (3) será executado. Caso contrário, o passo (4) será executado;
(3) Uma **pesquisa de opinião** com os especialistas da área de ES é planejada e executada para avaliar o conhecimento adquirido a partir da revisão sistemática (passo (2)). Os resultados poderão apoiar ou não a informação reunida no passo (2). Se a informação não for confirmada, volta-se ao passo (2). Caso contrário, segue-se para o passo (4);
(4) O **corpo de conhecimento** é consolidado a partir das informações adquiridas pela execução da revisão sistemática (e por meio da pesquisa de opinião com especialistas).

Dada a execução da estratégia de pesquisa, os pesquisadores têm acesso a um conjunto de informações que representa um corpo de conhecimento inicial, criado com base na informação extraída de trabalhos examinados e de parceiros de pesquisa (*survey*), além de agregar uma comunidade que o representa. Esse corpo de conhecimento deve ser fundamentado por uma análise do processo de ensino e aprendizagem em ES e abranger questões de pesquisa não especulativas e focadas na realidade nacional, incluindo uma combinação de práticas de ensino e técnicas de ES, bem caracterizada e baseada em evidência. Partindo do pressuposto de que as pesquisas na área de educação em ES no Brasil ainda são incipientes e localizadas, essa estratégia será utilizada para estabelecer uma comunidade de pesquisadores na área através do Fórum de Educação em Engenharia de Software (FEES). O processo de construção do corpo de conhecimento inicial, no contexto do FEES, servirá para estruturar e focar a discussão dessa comunidade, além de primar pela pesquisa colaborativa e em larga escala, onde cada um de seus grupos de pesquisa constituintes possa contribuir para os avanços na educação em ES no escopo de seu respectivo tópico de pesquisa (e.g., modelagem, qualidade, teste, reutilização etc.). Isso se faz importante para o desenvolvimento de uma comunidade de educadores de ES no Brasil, utilizando o FEES como um canal de comunicação que (i) a mantenha sintonizada com os avanços de pesquisa em educação em ES; (ii) viabilize estudos experimentais, visando avaliar e solidificar estes avanços na realidade brasileira; e (iii) contribua para manter o cenário nacional de educação em ES integrado, visando a maturidade e o estabelecimento da disciplina de ES no país.

No presente trabalho, o passo (1) foi exemplificado na Seção 2 e consistiu na identificação inicial de aspectos que permeiam a educação em ES, sob a responsabilidade dos pesquisadores autores deste artigo, para entendimento mínimo do estado da arte sobre educação em ES. Na Seção 4, o passo (2) será iniciado (i.e., apresentação do modelo inicial para o protocolo de revisão), a fim de incitar a sua discussão pelos pesquisadores e profissionais de ES e avaliar a sua viabilidade, servindo de atrativo para os grupos interessados em definir quais tópicos de educação em ES serão objetos das revisões sistemáticas e quais grupos ficarão responsáveis por cada um deles. A idéia principal da continuidade do passo (2) está em prover revisões sistemáticas colaborativas, especializadas e distribuídas, entre os diferentes grupos de pesquisa que atuam no mesmo tópico, a fim de produzir um estudo abrangente e desenvolvido por especialistas. Como sugestão, espera-se que esse passo possa ser conduzido no intervalo entre o primeiro e o segundo FEES, com participação dos pesquisadores autores deste artigo, usando um processo e uma infra-estrutura de apoio a revisões sistemáticas colaborativas e distribuídas. Considerando a evolução bem sucedida da

estratégia, espera-se também que resultados iniciais das revisões sejam apresentados no II FEES, em 2009.

Possivelmente, os aspectos que permeiam o processo de ensino e aprendizagem de ES identificados refletirão muito mais o cenário internacional, devido à ausência de bases de dados bem sedimentadas que reúnam artigos apresentados nos eventos brasileiros. Para minimizar essa questão, ocorreria a seleção manual de trabalhos nacionais, com base na experiência dos pesquisadores. Assim, a partir dos resultados apresentados no II FEES, parte-se para o passo (3), no intervalo entre o segundo e o terceiro FEES, isto é, a elaboração de uma pesquisa de opinião geral, baseada nas informações contidas no conjunto das revisões sistemáticas realizadas, e sua distribuição aos educadores de ES no Brasil, almejando abranger as diversas regiões, universidades e formações na área de Computação. Busca-se (com o *survey*) verificar se as preocupações com a educação em ES, apontadas no exterior, se refletem ou não no cenário brasileiro e, mais ainda, identificar quais são as suas peculiaridades e seus aspectos centrais.

Após a execução da pesquisa de opinião, passa-se para a tabulação e análise dos dados. Se ocorrerem discrepâncias significativas entre os aspectos levantados pelas revisões realizadas e a opinião dos educadores, esse resultado seria discutido no III FEES, em 2010, verificando a possibilidade de se voltar para a etapa (2) (i.e., novas revisões sistemáticas com base em melhorias sobre o protocolo). Caso contrário, serão apresentados os resultados extraídos da pesquisa de opinião, o que contribuiria para o delineamento das questões de pesquisa futuras a serem tratadas no FEES pela comunidade de pesquisa em educação em ES construída. Dessa forma, o passo (4) seria alcançado, com a organização de um corpo de conhecimento em educação em ES bem fundamentado, consolidado e construído para apoiar o dinamismo e a evolução das pesquisas de ES com relação ao seu ensino na academia. Esse passo é essencial para que a maturidade da área de ES como um todo seja alcançada, pois atua justamente na formação de seus recursos humanos, além de propiciar a solidificação da pesquisa em educação em ES, abrindo portas para estudos experimentais e geração de melhorias em escala nacional, contando com o apoio da comunidade de educadores desenvolvida.

4 Utilizando a Estratégia de Pesquisa em Educação em ES

A partir da caracterização inicial do problema (Seção 2), tem-se um ponto de partida para o planejamento da revisão sistemática, mediante a definição inicial de um protocolo científico. Esta seção visa exemplificar a primeira etapa do planejamento, conforme o *template* detalhado em [Biolchini *et al.*, 2007], a qual trata da *formulação da questão de pesquisa*. A idéia aqui é definir uma questão de pesquisa específica para que cada grupo de pesquisadores possa estruturar e conduzir a sua revisão sistemática, atendendo a um protocolo científico de referência. Dessa forma, a tarefa dos grupos de pesquisadores envolverá tanto o planejamento da revisão sistemática quanto à sua execução, onde grupos de uma mesma sub-área trabalharão em conjunto.

4.1 Formulação da Questão de Pesquisa

A seguir, é exibida a primeira etapa do planejamento do protocolo de revisão, intencionando servir de base para discussão e aprimoramento do protocolo de pesquisa.

Foco da Questão: O *objetivo* é analisar o processo de ensino e aprendizagem de Engenharia de Software *com o propósito de* ***caracterizar*** *com respeito à* identificação de problemas, soluções e desafios, além de peculiaridades do cenário nacional, *a partir do ponto de vista dos* pesquisadores de Engenharia de Software no Brasil, considerando suas especialidades *no contexto* das disciplinas e dos cursos de Engenharia de Software.

Qualidade e Amplitude da Questão:

Problema: A área de ES é constituída por várias sub-áreas e é normalmente apresentada em disciplinas com um denso conteúdo teórico-conceitual e ensinadas de forma tradicional. Em conseqüência, essas disciplinas nem sempre contribuem para uma boa formação de engenheiros de software, considerando as necessidades da indústria. Nesse sentido, a academia tem investido esforços na tentativa de minimizar esse quadro, experimentando novas metodologias e estratégias de ensino, ferramentas e processos. Entretanto, pelo fato de serem muitas vezes pontuais e localizados, esses esforços tendem a se dissipar e não contribuir para a formação de um corpo de conhecimento. Dessa forma, é necessário realizar um diagnóstico do quadro relacionado ao processo de ensino e aprendizagem de ES, identificando seus problemas, soluções e desafios, além das peculiaridades do cenário nacional.

Questões de Pesquisa:

Questão Principal P(<SUB-ÁREA**>):** Quais são os principais problemas, soluções e desafios no processo de ensino e aprendizagem de Engenharia de Software com relação à dimensão <SUB-ÁREA>? [*Cada revisão sistemática terá uma questão de pesquisa específica, conforme a sub-área a ser investigada*]

Intervenção: Problemas, soluções e desafios em educação em ES.

Controle: Shaw (2000), Baker *et al.* (2005), De Lucena *et al.* (2006), Huang & Distante (2006), Hilburn & Towhidnejad (2007), Lethbridge *et al.* (2007) e Nauman & Uzair (2007).

Comparação: Nenhuma.

Efeito: Identificação de problemas, soluções e desafios em educação em ES.

Medida de Resultado: Caracterização de problemas, soluções e desafios em educação em ES.

População: Pesquisas em ensino e aprendizagem de ES na dimensão <SUB-ÁREA>, representadas por artigos publicados na literatura técnica.

Aplicação: Educadores de ES em cursos de graduação (e pós) em Computação.

***Design* Experimental:** Estudos e experiências em geral.

Palavras-chave e Sinônimos (em inglês, devido ao idioma das bases eletrônicas ACM e IEEE)**:**

P*opulação*:

- Education: teaching, learning, training;
- Software Engineering: System Engineering;
- <SUB-ÁREA>: <SINÔNIMOS>.

I*ntervenção*:

- Problem: issue;
- Solution: resolution, success experience;
- Challenge.

4.2 Demais Etapas do Planejamento da Revisão Sistemática

Seguindo o *template* definido em [Biolchini *et al.*, 2007], as próximas quatro etapas são: (i) *seleção das fontes*: visa selecionar as fontes onde as buscas por estudos primários serão executadas a partir de *strings* de busca definidas; (ii) *seleção dos estudos*: consiste em descrever o processo e os critérios para a seleção e avaliação dos estudos; (iii) *extração da informação*: foca na extração de informações relevantes, de acordo com os critérios de inclusão e exclusão; para cada estudo selecionado após a execução do processo de seleção, os seguintes dados devem ser extraídos e tabulados: título, autores, fonte, problemas em educação em ES, soluções apresentadas (de sucesso ou não) e desafios, além de peculiaridades do cenário nacional, quando se aplicar; e (iv) *sumarização dos resultados*: visa analisar e apresentar os dados resultantes dos estudos selecionados, agregando-os com o intuito de responder à questão de pesquisa, por meio de uma análise estatística (e.g., porcentagem de ocorrência de um dado problema nos estudos

selecionados e sua correlação com variáveis como tipo e região das universidades envolvidas nos estudos). Um passo importante consiste na elaboração das *strings*, construídas com base nas palavras-chave (e sinônimos) e organizadas conforme a estrutura PICO (***P****opulation*, ***I****ntervention*, ***C****omparision* e ***O****utcome*) [Biolchini *et al.*, 2007]. Como a comparação (*comparision*) não foi definida e os termos do resultado (*outcome*) são os mesmos da intervenção (*intervention*), não foram definidas palavras-chave para ambos. Essa estrutura é combinada logicamente e as palavras-chave usadas para formar expressões que farão parte da combinação. Os sinônimos são arranjados para maximizar a quantidade de estudos relevantes obtida. Utilizando as palavras-chave definidas, teremos:

```
(software engineering education OR software engineering teaching OR
software engineering learning OR software engineering training OR
system engineering education OR system engineering teaching OR system
engineering learning OR system engineering training) AND (<SUB-ÁREAS>
OR <SINÔNIMO_1> OR ...)
AND
(problem OR issue OR solution OR resolution OR success experience OR
challenge)
```

5 Considerações Finais

A ES normalmente é apresentada nos cursos da área de Computação por meio de disciplinas de conteúdo teórico-conceitual e ensinadas de forma tradicional, o que faz com que nem sempre contribuam para uma boa formação de engenheiros de software, considerando as necessidades da indústria [Nauman & Uzair, 2007]. Apesar dos esforços da academia em minimizar essa realidade [Shaw, 2000] [Lethbridge *et al.*, 2007], é necessário realizar um diagnóstico do quadro relativo ao processo de ensino e aprendizagem de ES, identificando seus principais problemas, soluções e desafios, além das peculiaridades do cenário nacional.

Dessa forma, o presente trabalho explorou a experimentação como uma estratégia para apoiar a pesquisa em educação de ES. Um ponto fundamental da estratégia é a criação e manutenção de uma comunidade de pesquisadores em educação em ES, identificando questões de pesquisa não especulativas e focadas na realidade nacional. O FEES tem um papel importante para esta estratégia. Como próximos passos, espera-se prosseguir com a execução da estratégia de pesquisa. Assim, espera-se que os primeiros resultados sejam apresentados no II FEES, em 2009.

Referências

BAKER, A.; NAVARRO, E.O.; VAN DER HOEK, A. An Experimental Card Game for Teaching Software Engineering Processes. Journal of Systems and Software, New York, v. 75, n. 1-2, p. 3-16, Feb. 2005.

BIOLCHINI, J.C.A.; MIAN, P.G.; NATALI, A.C.C.; CONTE, T.U.; TRAVASSOS, G.H. Scientific Research Ontology to Support Systematic Review in Software Engineering. Advances Engineering Informatics, v. 21, n. 2, p. 133-151, Apr. 2007.

CONN, R. Developing Software Engineers at the C-130J Software Factory. IEEE Software, Los Alamitos, v. 19, n. 5, p. 25-29, Sep. 2002.

DANTAS, A.; BARROS, M; WERNER, C. Treinamento Experimental com Jogos de Simulação para Gerentes de Projeto de Software. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE – SBES '04, 18., 2004, Brasília. Anais ... Porto Alegre: SBC, 2004. p. 23-38.

DE LUCENA, V.F.; BRITO, A.; GOHNER, P. A Germany-Brazil Experience Report on Teaching Software Engineering for Electrical Engineering Undergraduate Students. In: CONFERENCE ON SE EDUCATION & TRAINING – CSEET '06, 19., 2006, Turtle Bay. Proceedings … Washington: IEEE Computer Society, 2006. p. 69-76.

DIAS NETO, A.C. Estratégia para Apoiar a Seleção e Avaliação de Abordagens de Teste de Software Baseado em Modelos. 2008. 104f. Exame de Qualificação (Doutorado) – Universidade Federal do Rio de Janeiro, COPPE - Programa de Engenharia de Sistemas, Rio de Janeiro, 2008.

FIGUEIREDO, E.; LOBATO, C.; DIAS, K.; LEITE, J.; LUCENA, C. Um Jogo para o Ensino de Engenharia de Software Centrado na Perspectiva de Evolução. In: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO – WEI '07, 15., CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 2007, Rio de Janeiro. Anais ... Porto Alegre: SBC, 2007. p. 37-46.

HIEBERT, J.; GALLIMORE, R.; STIGLER, J.W. A Knowledge Base for the Teaching Profession: What Would It Look Like and How Can We Get One?. Educational Research, v. 31, n. 5, p. 3-15, Jun./Jul. 2002.

HILBURN, T.B.; TOWHIDNEJAD, M. A Case for Software Engineering. In: CONFERENCE ON SOFTWARE ENGINEERING EDUCATION & TRAINING – CSEET '07, 20., 2007, Dublin. Proceedings … Washington: IEEE Computer Society, 2007. p. 107-114.

HUANG, S.; DISTANTE, D. On Practice-Oriented Software Engineering Education. In: CONFERENCE ON SOFTWARE ENGINEERING EDUCATION & TRAINING WORKSHOPS – CSEETW '06, 19., 2006, Turtle Bay. Proceedings … Washington: IEEE Computer Society, 2006. p. 15.

IEEE/ACM Joint Task Force on Computing Curricula. **Software Engineering 2004, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering**. IEEE Computer Society and ACM. Available at: http://sites.computer.org/ccse/. Accessed: 22 sep. 2008.

KITCHENHAM, B. **Procedures for Performing Systematic Reviews**. United Kingdom: Keele University, Department of Computer Science, 2004. 28 p. (Technical Report TR/SE-0401).

LETHBRIDGE, T.C.; DIAZ-HERRERA, J.; LEBLANC, R.J.; THOMPSON, J.B. Improving Software Practice through Education: Challenges and Future Trends. In: FUTURE OF SOFTWARE ENGINEERING, INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING – ICSE '07, 29., 2007, Minneapolis. Proceedings … Washington: IEEE Computer Society, 2007, p. 12-28

MCGETTRICK, A.; BOYLE, R.; IBBETT, R.; LLOYD, J.; LOVEGROVE, G.; MANDER,K. **Grand Challenges in Computing Education**.The British Computer Society,2004. 26p.

MEYER, B. Software Engineering in the Academy. Computer, Los Alamitos, v. 34, n. 5, pp. 28-35, May 2001.

MOHRENSCHILDT, M.V.; PETERS, D.K. The Draw-Bot: A Project for Teaching Software Engineering. In: ANNUAL FRONTIERS IN EDUCATION CONFERENCE – FIE '98, 28., 1998, Tempe. Proceedings … Washington: IEEE Computer Society, v. 3, 1998. p. 1022-1027.

NAUMAN, M.; UZAIR, M. SE and CS Collaboration: Training Students for Engineering Large, Complex Systems. In: CONFERENCE ON SOFTWARE ENGINEERING

EDUCATION & TRAINING – CSEET '07, 20., 2007, Dublin. Proceedings … Washington: IEEE Computer Society, 2007. p. 167-174.

PERRY, D.E.; PORTER, A.A.; VOTTA, L.G. Empirical Studies of Software Engineering: A Roadmap. In: FUTURE OF SOFTWARE ENGINEERING, INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING – ICSE '00, 22., 2000, Limerick. Proceedings … New York: ACM, 2000, p. 345-355.

SBC. **Grandes Desafios da Pesquisa em Computação no Brasil – 2006-2016**. Sociedade Brasileira de Computação, 2004. 22p.

SHAW, M. Software Engineering Education: A Roadmap. In: FUTURE OF SOFTWARE ENGINEERING, INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING – ICSE '00, 22., 2000, Limerick. Proceedings … New York: ACM, 2000, p. 371-380.

SPÍNOLA, R.O. Arcabouço para Especificação e Garantia da Qualidade de Requisitos de Ubiqüidade em Projetos de Software Ubíquo. 2008. 117f. Exame de Qualificação (Doutorado) – Universidade Federal do Rio de Janeiro, COPPE - Programa de Engenharia de Sistemas, Rio de Janeiro, 2008.

Elicitação de Requisitos - Evidências de uma Problemática na Formação dos Estudantes de Computação *

Andréa Pereira Mendonça[1,2] Evandro de Barros Costa[1,3]
Dalton Dario Serey Guerrero[1]

[1]Departamento de Sistemas e Computação – Universidade Federal de Campina Grande (UFCG)

[2]Coordenação de Informática - Centro Federal de Educação Tecnológica do Amazonas (CEFET - AM)

[3]Instituto de Computação – Universidade Federal de Alagoas (UFAL)

andrea@dsc.ufcg.edu.br, ebc.academico@gmail.com, dalton@dsc.ufcg.edu.br

Abstract. Computing students have difficulties to deal with requirements elicitation. This problem was highlighted by two empirical research - a case study carried out with students in the beginning of the course and a survey research conducted with computing teachers and IT professionals. In this paper, we present these research results and propose a pedagogical approach that supports the development of problem specification skills and good practices on software development. The approach proposed will be evaluated in the context of novice programming students.

Keywords: Requirements Elicitation, Empirical Researches, Programming Learning.

Resumo. Estudantes de computação apresentam dificuldades em lidar com elicitação de requisitos. Esta problemática foi evidenciada por meio de duas pesquisas empíricas - um estudo de caso com estudantes de programação, além de uma pesquisa do tipo *survey* com professores de computação e profissionais de TI. Neste artigo, apresentamos os resultados destas pesquisas e propomos uma abordagem pedagógica que possibilita o desenvolvimento das habilidades para especificação de problemas, assim como das boas práticas de desenvolvimento de software. A abordagem proposta será avaliada no contexto de estudantes iniciantes de programação.

Palavras-chave: Elicitação de Requisitos, Pesquisas Empíricas, Aprendizagem de Programação.

*A primeira autora recebe bolsa da FAPEAM - Fundação de Amparo à Pesquisa do Estado do Amazonas.

1 Introdução

No processo de desenvolvimento de *software*, a especificação do problema não é uma atividade simples, principalmente se considerarmos que a declaração de um problema tipicamente contém ambigüidades, contradições, falta de informações e/ou informações irrelevantes.

Transformar declarações dessa natureza em uma definição clara do que deve ser feito exige um conjunto de habilidades não triviais, tais como: identificação de informações relevantes, estratégias para aquisição e registro de informações, modelagem, diálogo e análise das restrições do problema. No modelo tradicional dos currículos de computação, tais habilidades são desenvolvidas nas disciplinas de Análise, Engenharia de Software ou denominações similares [MEC/SESU 1999].

Desta forma, é comum que nas séries iniciais, os estudantes lidem com o domínio da solução, priorizando a construção de programas e apenas, posteriormente, passam a lidar com o domínio do problema, tratando, por exemplo, com elicitação dos requisitos. Esta estratégia curricular não apenas contraria a forma canônica do processo de resolução de problemas, no qual, primeiramente, entende-se o problema e somente depois soluciona-o, como também não possibilita o satisfatório desenvolvimento das habilidades para especificação de problemas.

Além de analisar relatos da literatura [Winbladh 2004], nós realizamos duas investigações que apontam deficiências dos alunos em elicitar e especificar problemas - (i) um estudo de caso com alunos iniciantes de programação [Mendonça 2008b] e (ii) uma pesquisa do tipo *survey* com professores de computação e profissionais de TI [Mendonça 2008a].

A partir dos resultados dessas investigações e da análise da literatura [Deek 1997, Janzen and Saiedian 2006, Winbladh 2004], propusemos uma abordagem pedagógica, a ser implementada no contexto de alunos iniciantes, que possibilita o desenvolvimento das habilidades para especificação de problemas em conjunto com as boas práticas de desenvolvimento de *software*, como por exemplo, interação com cliente e escrita de casos de testes.

Nas Seções 2 e 3, respectivamente, apresentamos os resultados obtidos com o estudo de caso e a pesquisa *survey*. Na seção 4 descrevemos a abordagem pedagógica proposta, assim como a metodologia para avaliá-la e posteriormente, apresentamos as considerações finais sobre a pesquisa em pauta.

2 Estudo de Caso com Alunos Iniciantes de Programação

No período de 18/02/2008 a 10/04/2008 foi realizado um estudo de caso com alunos iniciantes de programação, do curso de Ciência da Computação, da Universidade Federal de Campina Grande.

A investigação foi realizada em duas fases: (i) estudo piloto realizado com 4 alunos e o estudo de caso, propriamente dito, realizado com 10 alunos. Cada aluno foi observado individualmente e cada sessão durava em média duas horas. Após a resolução de um problema de programação era realizada uma entrevista com o aluno. O planejamento, execução e registro do estudo de caso seguiu as orientações de Kitchenham *et al.* [Kitchenham et al. 1995] e Yin [Yin 1984]. A descrição detalhada da

investigação incluindo trechos de diálogos, versões de código e análise dos resultados obtidos estão descritos em Mendonça [Mendonça 2008b].

O estudo de caso teve por objetivo responder as seguintes questões de pesquisa: (Q1) Quais estratégias os alunos iniciantes empregam para especificar problemas? (Q2) Quais as dificuldades enfrentadas por estes alunos ao lidar com especificação de problemas? (Q3) Como os alunos iniciantes conduzem o diálogo para obtenção e esclarecimento das informações sobre o problema?

O procedimento empregado na investigação é descrito como segue: o aluno recebia um problema "mal definido"[6] e deveria solucioná-lo. A especificação completa do problema era de conhecimento apenas do investigador que fazia o papel de cliente. Para este problema foram definidos previamente 8 casos de testes. Caso o programa feito pelo aluno não passasse nos casos de testes previstos, o aluno deveria corrigi-lo. Cada entrega do programa ao investigador era contabilizada como uma versão. Não havia limitações quanto ao tempo e o aluno podia fazer perguntas a qualquer momento. A atividade era finalizada quando o programa passava em todos os casos de testes previstos.

2.1 Resultados da Investigação

Em média os alunos levaram 86 minutos para concluir o programa com aprovação em todos os casos de testes. Nenhum deles conseguiu elicitar todos os requisitos do problema na primeira versão do programa. Somente 20% dos alunos conseguiram finalizá-lo na segunda versão, sendo que 50% deles precisaram gerar mais de 4 versões para atender a todos os requisitos. Realizaremos a descrição dos resultados tomando como referência as questões de pesquisa.

- [Q1] - Quais estratégias os alunos iniciantes empregam para especificar problemas?

Verificamos na investigação que a maioria dos alunos, após a leitura do enunciado do problema, partiam diretamente para a construção do programa ou para escrita de um algoritmo. Assim, não realizavam uma análise exploratória do problema e a especificação do mesmo era construída no decorrer da codificação da solução.

- [Q2] - Quais as dificuldades enfrentadas pelos alunos ao lidar com especificação de problemas?

Os alunos apresentaram dificuldades em realizar leitura exploratória, interpretação do enunciado do problema, análise das restrições do problema, aquisição e registro das informações.

- Leitura Exploratória. Os alunos não exploravam previamente o enunciado do problema.

Os alunos apresentaram dificuldades em identificar informações faltantes, principalmente as que dizem respeito às entradas, saídas e restrições do problema. Em geral, questionavam apenas sobre as regras dos cálculos que deveriam ser feitos. Além disso, não exploravam o significado dos termos envolvidos no problema, mesmo que os desconhecessem.

- Interpretação. A maioria dos alunos ignorava a existência de um cliente e atribuía um significado pessoal para a informação.

6 Problema cuja descrição não é completa, podendo conter ambigüidades, contradições, falta de informações e/ou informações irrelevantes.

A maioria dos alunos construiu o programa seguindo uma interpretação pessoal do que devia ser feito, ignorando a perspectiva do cliente. Por exemplo, os alunos não questionavam sobre as saídas e em virtude disso duas situações ocorriam: ou imprimiam apenas o dado que era explicitamente descrito no enunciado, ou imprimiam as informações que eles próprios julgavam necessárias, impondo uma interpretação pessoal sobre o problema.

- Aquisição de informações. As dificuldades com leitura exploratória e interpretação comprometeram a aquisição de informações.

Obviamente, se o aluno não percebe a necessidade de informações ele também não desenvolve as estratégias para adquiri-las. As estratégias para aquisição de informações foram pautadas no diálogo e na submissão do programa. A maioria dos alunos submetiam o programa esperando que, quando houvesse erros, indicações do que precisava ser corrigido fossem apontadas. Essa é uma característica que, entre outras coisas, revela a falta de reflexão do aluno sobre o problema. Descreveremos as estratégias de diálogo posteriormente, em resposta a próxima questão de pesquisa.

- Análise das restrições do problema. Os alunos preocupavam-se com a verificação dos casos óbvios.

Os alunos apresentaram dificuldades em especificar as restrições do problema. Os testes, que poderiam ajudar nessa descoberta, eram concebidos para os casos óbvios e não exploravam as restrições e situações de erros no programa. Na maioria dos casos, os alunos concebiam dois ou três testes e o faziam no final da implementação.

- Registro das Informações. Os alunos não re-elaboravam a definição do problema.

Os registros das novas informações eram feitos através de pequenas anotações em papel sem preocupação com estilo e organização. As anotações utilizavam abreviações, simbologia matemática e setas relacionando blocos de informação. Em função do mau registro da informação, as dúvidas se repetiam em diferentes momentos da resolução de problemas. Os alunos não tinham a consciência de que o registro dos requisitos estabelecem um "contrato" definindo *o que* deve ser feito.

- [Q3] - Como os alunos iniciantes conduzem o diálogo para obtenção e esclarecimento das informações sobre o problema?

No geral, o diálogo ocorria sob demanda, com poucas perguntas sendo feitas inicialmente e as demais sendo realizadas à medida que o aluno codificava o programa. Nem todas as perguntas feitas eram direcionadas à especificação do problema, algumas delas visavam esclarecer aspectos de implementação. Em virtude de não perceberem as informações que precisavam ser esclarecidas, os alunos questionavam pouco. Dada a quantidade de perguntas imprecisas, constatamos a inabilidade dos alunos em elaborar perguntas objetivas para o esclarecimento das informações.

3 *Survey* com Professores de Computação e Profissionais de TI

Uma investigação mais abrangente foi realizada por meio de uma pesquisa do tipo *survey* no período de 06/04 a 08/05/2008. O *survey* foi do tipo *case control* [Kitchenham and Pfleeger 2002a], destinado a professores de computação e profissionais de TI. A coleta de dados ocorreu por meio de questionários, não supervisionados, disponibilizados na Web. A investigação contou com a participação de 205 professores

e 196 profissionais de TI de todas as regiões do país. O planejamento, execução, análise e registro dos resultados seguiu as orientações de Kitchenham and Pfleeger [Kitchenham and Pfleeger 2002a, Kitchenham and Pfleeger 2002b, Kitchenham and Pfleeger 2003] e foram descritos detalhadamente em Mendonça [Mendonça 2008a].

3.1 Resultados e Discussão

Ao avaliarem os alunos egressos dos cursos de computação com relação ao desenvolvimento das habilidades para especificação de problemas, 51,5% dos profissionais acreditam que poucos saem com essas habilidades e 58,6% dos professores acreditam que apenas parte dos alunos saem com essas habilidades.

Do total de professores, 53,9% deles afirmaram que os alunos freqüentemente apresentam dificuldades em compreender os problemas propostos na disciplina. Do mesmo modo, 52,6% dos profissionais, afirmaram que freqüentemente os desenvolvedores de software com os quais trabalham apresentam dificuldades em compreender os problemas propostos pelos clientes.

As sugestões deixadas pelos professores concentram-se em três aspectos: (i) a necessidade de conscientização dos próprios professores para o problema em questão; (ii) descrição das dificuldades percebidas nos alunos com relação à interpretação de textos, expressão oral e escrita; (iii) a necessidade de orientação pedagógica que auxilie o professor a lidar com esse problema no contexto de sala de aula.

Verificamos com a pesquisa que, tanto no contexto acadêmico quanto no profissional, são freqüentes as dificuldades com especificação de problemas. A investigação demonstrou ainda que professores e profissionais de TI consideram que as habilidades para entendimento de problemas não foram suficientemente desenvolvidas na graduação, haja vista a avaliação dos egressos feita por eles. Tal fato, evidencia a necessidade de melhorias na formação dos estudantes de computação.

4 Proposta de uma Abordagem Pedagógica

Nesta seção descrevemos uma abordagem pedagógica que tem por objetivo desenvolver as habilidades para especificação de problemas e as boas práticas de programação. A abordagem pressupõe que o entendimento do problema ocorre de maneira iterativa e incremental em duas fases que chamaremos de *entendimento inicial* e *refinamento da compreensão* (Figura 1). Problemas "mal definidos" devem ser propostos a fim de permitir aos estudantes o desenvolvimento das habilidades de leitura exploratória, análise, interpretação e diálogo.

Na fase de *entendimento inicial* o estudante toma conhecimento do enunciado do problema e deve produzir um artefato, que chamamos de *especificação inicial*. Este artefato deve conter os requisitos desejados pelo professor (cliente), os quais são elicitados por meio de diálogo, obedecendo o processo natural de perguntas e respostas.

A intenção nessa fase é que aluno vivencie um contexto mais próximo do mundo real e exercite as atividades de elicitação e especificação de requisitos. O documento de especificação não obedece um *template*. Os alunos devem expressar a especificação do problema em um formato textual, expandindo o enunciado anteriormente apresentado pelo professor (cliente). A opção por adotar um documento textual sem um formato

pré-definido justifica-se pela simplicidade nessa etapa. Orientações quanto ao estilo e organização do documento de especificação são fornecidos aos alunos.

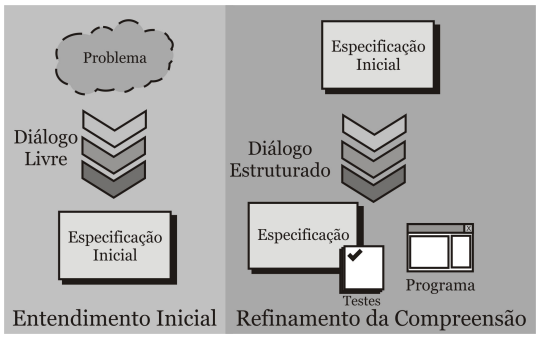


Figura 1 – Abordagem Pedagógica.

Como foi observado no estudo de caso, os alunos apresentaram dificuldades na elaboração de perguntas. Na fase de entendimento inicial, na qual o diálogo é livre, é importante que o aluno faça questionamentos com qualidade. Para isto, estratégias de diálogo foram desenvolvidas com o objetivo de orientar os estudantes na reflexão sobre o problema, na habilidade de expressar claramente as dúvidas e na aquisição de autonomia com relação a própria aprendizagem. A estratégia de diálogo está estruturada em três dimensões: *propósito*, *forma* e *conteúdo* (Figura 2) e foi inspirada no trabalho de Lane [Lane 2004].

Figura 2 – Estratégias de Diálogo.

O *propósito* diz respeito ao objetivo pedagógico que se deseja alcançar. A *forma* indica a estrutura da interação e o *conteúdo* refere-se ao teor da interação. Por exemplo, no caso em que o aluno faz uma pergunta sobre um determinado requisito, o *propósito* do diálogo é o de esclarecer informações, isto pode ser feito na *forma* de uma resposta direta com informações sobre o problema (*conteúdo*).

Quando o aluno consegue estabelecer um acordo com o professor (cliente) sobre o que deve ser feito, expressando-o por meio do documento de *especificação inicial*, ele passa para a fase de *refinamento da compreensão* que ocorre gradativamente à medida que o aluno vai construindo a solução do problema. Nesta fase, o diálogo continua sendo a "força motriz" do processo de elicitação de requisitos, porém ele deve, agora, processar-se de maneira mais estruturada, na forma de *casos de testes*. Testes, portanto, são entendidos como uma pergunta, escritos em uma sintaxe estruturada, que questiona o professor (cliente) quanto aos requisitos e restrições do problema.

Nesta fase, motivamos a prática de mais uma atividade da engenharia de *software* - testes. Entretanto, não a utilizamos seguindo os rigores de TDD (*Teste Driven Development*) [Beck 2002], isto é, que os testes sempre precedam a codificação. Nesta abordagem, incentivamos a criação de casos de testes de maneira contínua e antecipada, porém não adotamos o rigor de que os mesmo sejam concebidos sempre antes da codificação. O mais importante nesta fase é que os alunos utilizem testes como

práticas efetivas que auxiliam na confirmação e descoberta de requisitos, na reflexão sobre corretude e situações de erro que podem ocorrer no programa.

Para estabelecer uma continuidade nas atividades desenvolvidas pelos alunos, eles são motivados a expressarem em casos de testes as declarações de requisitos presentes no documento de especificação inicial. Isto contribui para que o aluno passe a dialogar de maneira mais estruturada e que continue especificando novos requisitos.

Nessa fase, a compreensão do problema é mensurada pelo número de testes cobertos pelo aluno. Caso ele submeta o programa a avaliação do professor sem que todas as funcionalidades tenham sido atendidas, um novo ciclo de *refinamento da compreensão* é iniciada. Nesta fase, além dos testes, o aluno deve complementar o documento de especificação e produzir o programa.

Nas duas fases da abordagem, o estudante é motivado a refletir sobre o problema antes de simplesmente codificá-lo. Para produzir o documento de especificação inicial, por exemplo, o estudante deverá explorar o enunciado, detectar as informações faltantes, contraditórias ou ambíguas, deve esclarecer tais informações por meio do diálogo e registrá-las de maneira organizada no documento de requisitos. Na segunda fase, além de refinar essas atividades, o estudante desenvolverá a habilidade de testes e, conseqüentemente, a análise das restrições do problema. Acreditamos que essa intervenção pedagógica irá minimizar as deficiências apresentadas pelos alunos, conforme descritas no estudo de caso (Seção 2).

4.1 Avaliação

A avaliação da abordagem proposta será feita por meio de dois estudos de caso: o primeiro, com o objetivo de avaliar os efeitos da abordagem, as dificuldades de aplicação, a adequação das variáveis de controle e das formas de medir os resultados. Esta investigação funcionará como um estudo piloto que orientará a formalização do segundo estudo de caso, a ser realizado com dois grupos - um de controle e outro experimental.

O primeiro estudo de caso será realizado com a turma de alunos iniciantes do Curso de Ciência da Computação da Universidade Federal de Campina Grande no período de setembro de 2008 a fevereiro de 2009. Python será a linguagem de programação utilizada e os testes serão feitos usando expressões *assert*, próprias da linguagem.

5 Considerações Finais

Neste artigo apresentamos evidências de uma problemática na formação do estudante de computação - falta de habilidades para especificação de problemas. Estas evidências foram obtidas por meio de duas investigações: um estudo de caso com alunos iniciantes de programação e um *survey* realizado com professores e profissionais da área. A dificuldade com elicitação de requisitos, entretanto, não é característica de estudantes brasileiros. Outros autores relatam problemas semelhantes, em diferentes países, inclusive, alertando para o *gap* entre a formação dos estudantes e as exigências da indústria de software [Winbladh 2004, O'Leary et al. 2006, Garg and Varma 2008].

Embora os resultados obtidos não possam ser, facilmente, generalizados em termos estatísticos, eles contribuíram para uma discussão inicial sobre o tema e permitiram observar comportamentos e algumas dificuldades dos alunos ao realizarem atividades de especificação. A observação desses fenômenos é especialmente importante no

contexto de computação porque grande parte dos problemas são resolvidos fora do horário de sala de aula e em virtude disto, os professores realizam pouco acompanhamento do processo de resolução, tendo mais acesso ao produto final que é entregue pelo aluno na forma de um programa ou algoritmo.

A fim de contribuir para a melhoria do processo ensino-aprendizagem, propomos uma abordagem pedagógica que possibilita ao aluno, já na série inicial, conviver com problemas "mal definidos" e desenvolver habilidades para solucioná-los, exercitando atividades da engenharia de software, tais como, testes, elicitação e especificação de requisitos.

O diferencial desta abordagem reside no fato de congregar diferentes atividades de desenvolvimento de *software* e conceber o entendimento do problema como uma atividade iterativa e incremental, no contexto de alunos iniciantes. Outros trabalhos já enfatizaram o uso de testes [Janzen and Saiedian 2006] e especificação [Brown 1988] com alunos iniciantes, mas não o fizeram de forma conjunta. Deek [Deek 1997] propôs um modelo para resolução de problemas e desenvolvimento de programas que congrega todas as etapas de desenvolvimento de *software*. Entretanto, o modelo proposto por ele assemelha-se ao desenvolvimento em cascata, em que o entendimento e solução dá-se de maneira seqüencial e progressiva, contrariando a perspectiva incremental e iterativa do processo de desenvolvimento de *software*. Além disso, na fase de formulação do problema, o autor baseia-se na estratégia de extrair do enunciado os elementos relevantes do problema. Nossa perspectiva é de que o entendimento baseia-se não somente na extração de informações do enunciado, mas também em um processo de interação com o cliente (professor) por meio do diálogo.

Acreditamos que a abordagem proposta traz três contribuições significativas: (i) proporcionar ao aluno iniciante o desenvolvimento de outras habilidades além da codificação, consolidando, já na série inicial, algumas práticas da engenharia de *software*; (ii) sistematizar um procedimento de ensino-aprendizagem que pode ser replicado em outras universidades; (iii) colaborar por meio dos resultados obtidos para uma reflexão futura sobre a integração gradativa do currículo, na qual os alunos não precisam mais esperar até as disciplinas de Análise ou Engenharia de Software para desenvolver habilidades de especificação.

Cabe por fim ressaltar que a programação é considerada um dos sete grandes desafios em educação para computação [McGettrick et al. 2004]. Dentre os desafios citados estão o de desenvolver estudos sistemáticos para melhorar a compreensão das habilidades humanas neste domínio e proporcionar metodologias de ensino que desenvolvam as habilidades necessárias para o desenvolvimento de programas. Acreditamos que com a avaliação da abordagem proposta estaremos colaborando para a construção de soluções e superação desses desafios.

Referências

Beck, K. (2002). Test-Driven Development By Example. Addison Wesley.

Brown, D. A. (1988). Requiring CS1 students to write requirements specifications: a rationale, implementation suggestions, and a case study. In SIGCSE '88: Proceedings of the nineteenth SIGCSE technical symposium on Computer science education, pages 13–16, New York, NY, USA. ACM.

Deek, F. P. (1997). An integrated environment for problem solving and problem development. PhD thesis, New Jersey Institute of Technology, Newark.

Garg, K. and Varma, V. (2008). Software engineering education in india: Issues and challenges. IEEE 21st Conference on Software Engineering Education and Training (CSEET '08), pages 110–117.

Janzen, D. S. and Saiedian, H. (2006). Test-driven learning: intrinsic integration of testing into the cs/se curriculum. In SIGCSE '06: Proceedings of the 37th SIGCSE technical symposium on Computer science education, pages 254–258, New York, NY, USA.

Kitchenham, B. and Pfleeger, S. L. (2003). Principles of survey research part 6: data analysis. SIGSOFT Softw. Eng. Notes, 28(2):24–27.

Kitchenham, B., Pickard, L., and Pfleeger, S. L. (1995). Case studies for method and tool evaluation. IEEE Softw., 12(4):52–62.

Kitchenham, B. A. and Pfleeger, S. L. (2002a). Principles of survey research part 2: designing a survey. SIGSOFT Softw. Eng. Notes, 27(1):18–20.

Kitchenham, B. A. and Pfleeger, S. L. (2002b). Principles of survey research: part 3: constructing a survey instrument. SIGSOFT Softw. Eng. Notes, 27(2):20–24.

Lane, H. C. (2004). Natural Language Tutoring and the Novice Programmer. PhD thesis, University of Pittsburg. Department of Computer Science.

McGettrick, A., Boyle, R., Ibbett, R., Lloyd, J., Lovegrove, G., and Mander, K. (2004).

Grand challenges in computing education. Technical report, British Computer Society.

MEC/SESU (1999). Diretrizes curriculares de cursos da Área de computação e informática. Technical report.

Mendonça, Andréa P. (2008a). Entendimento de problemas - um survey com profissionais de TI, professores e alunos de computação. Technical Report DSC/004/2008, Departamento de Sistemas e Computação. Coordenação de Pós-Graduação em Ciência da Computação. Universidade Federal de Campina Grande.

Mendonça, Andréa P. (2008b). Entendimento de problemas: Uma investigação exploratória com alunos iniciantes de programação. Technical Report DSC/005/2008, Departamento de Sistemas e Computação. Coordenação de Pós-Graduação em Ciência da Computação. Universidade Federal de Campina Grande.

O'Leary, C., Lawless, D., Gordon, D., Haifeng, L., and Bechkoum, K. (2006). Developing a software engineering curriculum for the emerging software industry in china. In CSEET '06: Proceedings of the 19th Conference on Software Engineering Education & Training, pages 115–122, Washington, DC, USA. IEEE Computer Society.

Winbladh, K. (2004). Requirements engineering: closing the gap between academic supply and industry demand. Crossroads, 10(4):4–4.

Yin, R. K. (1984). Case Study Reseach Design and Methods. Sage Publications.

Qualidade, Confiabilidade e Segurança nas Disciplinas do Processo Unificado

Strauss Cunha Carvalho[1], Felipe Rafael Motta Cardoso[1], Adílson Marques da Cunha[1], Luis Alberto Vieira Dias[1]

[1]Instituto Tecnológico de Aeronáutica (ITA)
Praça Marechal Eduardo Gomes, 50, Vila das Acácias
Cep 12228-901, São José dos Campos, SP, Brasil

{strauss@ita.br, felipem@ita.br, cunha@ita.br, vdias@ita.br

Abstract. *This paper describes an Academic Experience performed by graduate and postgraduate students at the Brazilian Aeronautical Institute of Technology (ITA), during the second semester of 2007. It happened during a System Prototype construction for Real Time Embedded System and Quality, Reliability and Safety courses. At that time, Rational Unified Process (RUP) disciplines ,the Real Time Unified Modeling Language (RT-UML) and the Integrated Computer Aided Software Engineering Environment (I-CASE-E) were used in a case study named Unmanned Aircraft Vehicle – Control Station – Ecological Satellite Monitoring (VANT-EC-SAME).*

Keywords. *Real Time Embedded System, Software Quality, Reliability and Safety, Rational Unified Process (RUP), Real Time Unified Modeling Language (RT-UML), Integrated Computer Aided Software Engineering Environment (I-CASE-E).*

Resumo. *Este artigo descreve uma experiência acadêmica realizada por alunos de graduação e pós-graduação do Instituto Tecnológico de Aeronáutica (ITA), durante o segundo semestre de 2007. Essa experiência ocorreu durante a elaboração de um Protótipo de Sistema para as matérias Sistemas Embarcados de Tempo Real e Qualidade, Confiabilidade e Segurança (Safety) de Software. Naquela ocasião, as disciplinas do Processo Unificado da Rational, o padrão da Linguagem de Modelagem Unificada para Sistemas de Tempo Real e o Ambiente Integrado de Engenharia de Software Ajudada por Computador foram utilizadas em um Estudo de Caso denominado Veiculo Aéreo Não Tripulado – Estação de Controle – Satélite Artificial para Monitoramento Ecológico (VANT-EC-SAME).*

Palavras-Chave. *Sistemas Embarcados de Tempo Real; Qualidade, Confiabilidade e Segurança de Software; Processo Unificado da Rational, Linguagem de Modelagem Unificada para Sistemas de Tempo Real; Ambiente Integrado de Engenharia de Software Ajudada por Computador.*

1 Introdução

A crescente necessidade por produtos e/ou serviços de software vêm ocasionando um aumento da demanda de desenvolvimento de sistemas de informação. Nos últimos anos, para se atender com qualidade a construção de Sistemas Computadorizados, alguns processos de desenvolvimento de software vêm sendo utilizados com sucesso.

Este artigo descreve uma experiência acadêmica realizada por alunos de graduação e pós-graduação do Instituto Tecnológico de Aeronáutica – ITA, durante o segundo semestre de 2007 utilizando o Processo Unificado da Rational (*Rational Unified Process - RUP*) [1].

Essa experiência ocorreu durante a elaboração de um Protótipo de Sistema para as matérias CE-235 Sistemas Embarcados de Tempo Real [2] e CE-230 Qualidade, Confiabilidade e Segurança (*Safety*) de Software [2]. Nelas, as disciplinas do RUP, o padrão da Linguagem de Modelagem Unificada para Sistemas de Tempo Real (*Real Time - Unified Modeling Language - RT-UML*) [3] e o Ambiente Integrado de Engenharia de Software Ajudada por Computador (*Integrated Computer Aided Software Engineering Environment – I-CASE-E*) [4] foram utilizadas em um Estudo de Caso denominado Veiculo Aéreo Não Tripulado – Estação de Controle – Satélite Artificial para Monitoramento Ecológico (VANT-EC-SAME).

Este artigo evidencia a utilização das ferramentas do I-CASE-E para a obtenção de resultados nas seguintes disciplinas do RUP: Requisitos; Análise e Design; Implementação; Teste; Gerenciamento de Configuração; e Gerenciamento de Projeto.

Ele encontra-se dividido em 5 Seções: A introdução; A Seção 2 que descreve o cenário da pesquisa, os métodos e os processos adotados para o desenvolvimento de um protótipo de software; A Seção 3 que apresenta um Estudo de Caso adotado nas disciplinas ministradas; A Seção 4 que descreve os principais resultados obtidos; As conclusões são apresentas na Seção 5.

2 Descrição do Cenário

Um Estudo de Caso foi realizado envolvendo um **V**eículo **A**éreo **N**ão **T**ripulado, operando a partir de uma **E**stação de **C**ontrole e de um **Sa**télite Artificial de **M**onitoramento **E**cológico, denominado **VANT-EC-SAME** cujo objetivo é a realização de Missões de Reconhecimento e Sensoriamento Remoto de recursos naturais e ambientais tais como: reservatórios de águas, reservas florestais, jazidas minerais.

Um VANT deveria propiciar a obtenção de informações, a partir de imagens a serem enviadas para uma Estação de Controle (EC). Plataformas de Coleta de Dados (PCD) deveriam obter dados meteorológicos e hidrológicos, enviando-os para uma EC. Tal EC poderia receber dados de VANTs, PCDs e/ou Satélites, por meio de comandos enviados aos PCDs e aos VANTs e monitoraria os Satélites, que periodicamente enviaram imagens para a EC.

3 O Estudo de Caso

O Estudo de Caso do VANT-EC-SAME foi considerado um Sistema de Software de Computador (SSC). Ele foi dividido em Itens de Configuração de Software de Computador (ICSCs), que por sua vez foram divididos em Componentes de Software de Computador (CSCs), finalmente divididos em Unidades de Software de Computador (USC). Essas sub-divisões caracterizam a utilização da primeira das seis melhores práticas de desenvolvimento de software recomendadas pela Rational, o Desenvolvimento Iterativo e Incremental.

Grupos de 3 a 5 alunos foram alocados para os CSCs, a fim de realizarem a 2º Iteração da 1º Fase de Iniciação e/ou a 1º Iteração da fase de Elaboração do RUP.

Para os ICSCs, os artefatos do RUP dos 1ª e 2ª Níveis de Integração foram refinados em uma 2ª Iteração da Fase de Elaboração dos CSCs e em uma 1ª Iteração da Fase de Construção. O último Nível de Integração (3ª Nível) foi realizado, a partir da junção dos ICSCs, deste processo, resultou o protótipo do VANT-EC-SAME.

4 Principais Resultados

Os fatores que contribuíram para que os resultados fossem considerados satisfatórios foram: À adoção do *RUP*; O uso das ferramentas I-CASE-E; e à integração ocorrida entre as matérias CE-230 Sistemas Embarcados de Tempo Real e CE-235 Qualidade, Confiabilidade e Segurança (*Safety*) de Software. A estratégia adotada mostrou-se apropriada na medida em que propiciou o relacionamento das experiências adquiridas pelos alunos no desenvolvimento de Sistemas Embarcados de Tempo Real com as suas respectivas medições quanto a Qualidade do Software. Enquanto a 1ª matéria voltou-se para a análise e desenvolvimento, a 2ª matéria voltou-se mais para as medições e a garantia da qualidade do que foi produzido.

Dos itens 4.1 até 4.6, são reportados os principais resultados obtidos, bem e os artefatos produzidos nas 6 disciplinas do RUP aplicadas no Estudo de Caso do VANT-EC-SAME.

4.1 Requisitos

As principais características da Disciplina Requisitos do *RUP* são: 1-Estabelecer a concordância com os clientes sobre o que o sistema deverá propiciar; 2 - Definir as fronteiras do sistema; e 3 - Oferecer aos desenvolvedores uma melhor compreensão das regras de negócio. Para maximizar os resultados dos itens acima citadas, utilizou-se a ferramenta *Rational RequesitePro* [5] e a produção de cinco (5) artefatos, sendo:

1- Glossário (GLO); 2 - Modelo de Casos de Uso (MCU); 3 - Solicitações dos Principais Envolvidos (SPE); 4 - Especificações Suplementares (ESU); e 5 - Plano de Gerenciamento de Requisitos (PGR).

A ferramenta *Rational RequisitePro* propiciou, desde o início do processo de desenvolvimento, por meio da matriz de rastreabilidade, manter sempre informados os integrantes da equipe de desenvolvimento sobre o impacto de um novo requisito ou um requisito já existente, minimizando os desvios no cronograma e orçamento do projeto.

4.2 Análise e Projeto

Seguindo o paradigma da Orientação a Objetos (OO) [4], dentro do Estudo de Caso VANT-EC-SAME, o foco da Disciplina de Análise e Projeto (*Analysis and Design*) é transformar os requisitos em um projeto do sistema a ser criado e desenvolver sua arquitetura. Para satisfazer tais condições, foram elaborados na ferramenta *Rational Rose Real Time* [5] os seguintes diagramas: de classe, de estrutura, de seqüência, de transição e de estado. Tal ferramenta baseia-se na *RT-UML*, uma extensão da *UML*[4] e

vem sendo amplamente utilizada para descrever modelos de sistema computadorizados de tempo real, utilizando os conceitos de cápsulas, protocolos, portas, sinais e conexões.

Foi constatado que a utilização da ferramenta propiciou uma modelagem adaptada à natureza reativa dos sistemas de tempo real.

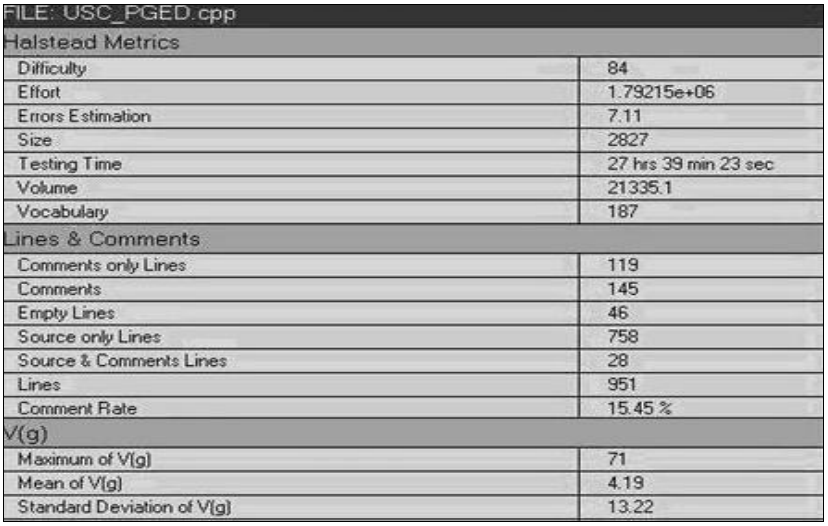
4.3 Implementação

A ferramenta *Rational Rose Real Time* propiciou a implementação baseada no padrão da *RT-UML*, utilizando-se modelagem visual por cápsulas compostas por protocolos e Portas de comunicação, representando a matéria-prima para a geração de código-fonte na linguagem C++. Foram gerados, para o Estudo de Caso do Protótipo de Sistema Embarcado de Tempo Real, VANT-EC-SAME, um total de 52.400 linhas de código em C++, contabilizadas através da ferramenta *Rational Test RealTime (RTRT)* [5].

4.4 Teste

Após a geração do código-fonte em C++, a ferramenta *Rational Test RealTime (RTRT)* foi utilizada para a realização de testes, com o intuito de mensurar os seguintes fatores: dificuldade, esforço, estimativa de erros, tamanho em linhas de código-fonte, percentual de comentários, entre outros. Para a realização dos testes em código-fonte, foi utilizada a estratégia de realização de medições por arquivo de código fonte.

A Figura 02 apresenta o resultado de algumas métricas aplicadas no código-fonte, utilizando-se a ferramenta *RTRT*. Nela, reporta-se os principais resultados obtidos por meio de um teste de sensitividade em um arquivo de código-fonte denominado USC_PGED.CPP, que contendo 951 linhas de código.



FILE: USC_PGED.cpp	
Halstead Metrics	
Difficulty	84
Effort	1.79215e+06
Errors Estimation	7.11
Size	2827
Testing Time	27 hrs 39 min 23 sec
Volume	21335.1
Vocabulary	187
Lines & Comments	
Comments only Lines	119
Comments	145
Empty Lines	46
Source only Lines	758
Source & Comments Lines	28
Lines	951
Comment Rate	15.45 %
V(g)	
Maximum of V(g)	71
Mean of V(g)	4.19
Standard Deviation of V(g)	13.22

Figura 02: Resultado dos Testes nas Linhas de Código do Arquivo USC_PGED.CPP.

O resultado obtido possibilitou as seguintes métricas:

- Halstead Metric **–** Mede a complexidade de um módulo do programa diretamente a partir do código, com ênfase na complexidade computacional;

- Lines / Comments **–** É um indicador percentual de comentário e se refere a legibilidade do código visando avalia o software em relação a manutenibilidade.

-V(g)Complexidade Ciclomática - É baseada na teoria de grafos e essencialmente representa o número de caminhos independentes possíveis em um código e também é um bom indicador do quanto um programa ou função é testável.

Os resultados Obtidos e Recomendados pela ferramenta após à aplicação das métricas em Código Fonte são apresentados na Tabela 01. Dois (2) artefatos foram produzidos na disciplina de Teste, sendo: 1-Caso de Uso de Testes (CUT); e 2 - Plano de Testes (PDT).

Tabela 01: Resultados obtidos e recomendados das Métricas de Teste em Código Fonte.

	Obtido	Recomendado
Halstead Metric	8,4	1 - 10
Lines / Comments	15,45 %	20 %
V(g)Complexidade Ciclomática	4,19	Máximo 10

4.5 Gerenciamento de configuração e mudança

A disciplina de Gerenciamento de Configuração e Mudanças do *RUP* propiciou o controle de versão de artefatos e de atribuições das tarefas aos 19 alunos integrantes do Estudo de Caso do VANT-EC-SAME, respeitando-se as atividades inerentes aos respectivos papéis pré-estabelecidos. A ferramenta utilizada foi o *Rational ClearQuest* [5] na qual propiciou uma gestão de um conjunto de solicitações de mudanças que designaram tarefas para os integrantes do referido Estudo de Caso.

4.6 Gerenciamento de projeto

A ferramenta MS-Project propiciou a elaboração dos cronogramas previsto e realizado, caracterizando assim 17 semanas como o tempo de duração total do Estudo de Caso ao longo do processo de desenvolvimento para atender às premissas da disciplina e prover a gestão das seguintes estimativas para o VANT-EC-SAME:

Cinco (5) artefatos foram produzidos na disciplina de Gerenciamento de Projetos do *RUP*, sendo: 1- Plano de Desenvolvimento de Software (PDS); 2 - Lista de Riscos

(LDR); 3 - Plano de Iteração (PDI); 4- Plano de Garantia da Qualidade (PGQ); e 5 - Caso de Desenvolvimento (CDD.

5 Conclusão

O presente artigo relatou uma experiência acadêmica e prática de utilização do *RUP*, do padrão da *RT-UML* e de um ambiente *I-CASE* no Estudo de Caso do VANT-EC-SAME. Ele propiciou o desenvolvimento de um Sistema Embarcado de Tempo Real com Qualidade, Confiabilidade e Segurança (*Safety*) de Software em duas matérias do Programa de Pós-Graduação em Engenharia Eletrônica e Computação na Área de Informática (PG/EEC-I) do Instituto Tecnológico de Aeronáutica - ITA, no segundo semestre de 2007.

O Estudo de Caso foi realizado por um grupo de 19 alunos, de forma Iterativa e Incremental, seguindo as melhores práticas de desenvolvimento de Software, em um período acadêmico semestral de apenas 17 semanas, produzindo-se: aproximadamente 52.400 linhas de código-fonte, em C++; 12 artefatos do *RUP*, em diferentes versões; e 4 diagramas da *UML,* mostrando a viabilidade de utilização da sistemática adotada em cursos de Engenharia Eletrônica e Computação que vêm sendo ministrados com sucesso no ITA.

6 Referências

[1] Rational Software Corp. **Rational Unified Process –** version 2003
Product Documentation. [S.l.], 2003.
http://www.wthreex.com/rup/ (acesso em 31/07/2008)

[2] A. M. Cunha, **Sistemas Embarcados e de Tempo Real (CE-235) e Qualidade, Confiabilidade e Segurança (Safety) de Software (CE-230).** Notas de Aula. Instituto Tecnológico de Aeronáutica (ITA),
www.comp.ita.br/~cunha, 2007. (acesso em 31/07/2008)

[3] Shang-Wen Cheng, and David Garlan – "**Mapping Architectural Concepts to UMLRT"**.2003

[4] BOOCH, Grady. "**UML - Guia do usuário"**. Rio de Janeiro: Campus, 2000.

[5] Rational Software Corp. **Rational Software** (2003),
http://www306.ibm.com/software (acesso em 31/07/2008)

Estudo de Caso abrangendo o Ensino Interdisciplinar de Engenharia de Software

Adilson Marques da Cunha[1], Gláucia Braga e Silva[1], Juliano de Almeida Monte-Mor[1], Marco Antonio Pizani Domiciano[1,2] e Ricardo Godoi Vieira[1]

[1]Departamento de Engenharia Eletrônica e Computação - Área de Informática (IEC) - Instituto Tecnológico de Aeronáutica (ITA), Brasil

[2]Divisão de GeoInteligência (EGI) - Instituto de Estudos Avançados (IEAv), Brasil

{cunha, glaucia, montemor, vieira}@ita.br, pizani@ieav.cta.br

Abstract. This paper tackles the teaching methodology adopted in a Case Study during the first semester of 2008, at the Electronics and Computer Engineering graduate program of the Brazilian Aeronautics Institute of Technology (ITA). It describes courses and actual projects integration under development at ITA. Initially, an academic project of reduced scope was developed involving courses of Database (DB) Systems Project, Information Technologies (IT), and Software Testing. On this project it was implemented a DB system, an information system, and a software testing prototype by applying different DB, IT and Software Testing tools and techniques.

Keywords: Interdisciplinarity, Case Study, Teaching Methodology, Academic Project, Software Engineering.

Resumo. Este artigo aborda a metodologia de ensino adotada em um Estudo de Caso realizado no 1º Semestre de 2008, no Curso de Pós-Graduação em Engenharia Eletrônica e Computação, do Instituto Tecnológico de Aeronáutica (ITA). Ele descreve uma integração entre disciplinas e projetos reais que se encontram em desenvolvimento no ITA. Inicialmente, definiu-se um projeto acadêmico de escopo reduzido, envolvendo as disciplinas de Projeto de Sistemas de Banco de Dados (BD), Tecnologias da Informação (TI) e Teste de Software. Neste projeto, implementou-se: um protótipo de sistema de BD, um protótipo de sistema de informação, e um protótipo de teste de software, aplicando-se, respectivamente, técnicas e ferramentas de BD, TI e Teste de Software.

Palavras-chave: Interdisciplinaridade, Metodologia de Ensino, Projeto Acadêmico, Estudo de Caso, Engenharia de Software.

1 Introdução

A realização de atividades práticas em disciplinas acadêmicas constitui uma importante complementação da parte teórica dessas, além de propiciar aos alunos o entendimento e a fixação dos conceitos aprendidos.

Para realização de atividades práticas, a adoção de Estudos de Caso baseados em projetos reais, tem sido praticada no Curso de Pós-Graduação em Engenharia Eletrônica e Computação, na área de Informática (PG/EEC-I) do Instituto Tecnológico de Aeronáutica (ITA), principalmente nas seguintes disciplinas ministradas por professores membros do Grupo de Pesquisa em Engenharia de Software (GPES): Projeto de Sistemas de Banco de Dados (CE-240); Qualidade, Confiabilidade e Segurança (*Safety*) de Software (CE-230); Sistemas Embarcados de Tempo Real (CE-235); Tecnologias da Informação (CE-245); e Teste de Software (CE-229).

Dentro deste contexto, este artigo relata as experiências obtidas no ITA quanto à aplicação de uma metodologia de ensino, para condução das atividades práticas nas disciplinas do GPES, abrangendo a definição do Estudo de Caso, o planejamento das atividades, e o acompanhamento e a gestão dos trabalhos desenvolvidos.

O restante deste artigo encontra-se organizado em 6 seções: na seção 2, mostra-se um histórico de alguns Estudos de Caso já realizados com sucesso no ITA. A seção 3 apresenta a metodologia de ensino adotada nesses Estudos de Caso. Na seção 4, descreve-se como esta metodologia foi aplicada em um projeto acadêmico envolvendo três disciplinas. Por fim, as seções 5 e 6 apresentam, respectivamente, os principais resultados obtidos e algumas considerações finais.

2 Histórico de Projetos Acadêmicos

A Tabela 4 ilustra alguns projetos reais de colaboração científica e tecnológica desenvolvidos entre o ITA e outras Instituições, que vêm motivando, ao longo dos anos, diferentes cenários para trabalhos acadêmicos e práticos realizados em disciplinas ministradas no PG/EEC-I do ITA.

A utilização desses Estudos de Caso vem proporcionando um universo de pesquisa e desenvolvimento que extrapola as fronteiras da sala de aula e têm originado diversas produções acadêmicas como dissertações de mestrado, teses de doutorado, assim como artigos publicados em eventos nacionais e internacionais, como por exemplo, [Silva e Cunha, 2004], [Martins et al., 2005], [Cunha et al., 2006], [Colonese et al., 2006], [Colonese et al., 2007] e [Loubach et al., 2008].

Assim, os conceitos transmitidos em aulas teóricas vêm sendo aplicados, praticados e investigados em situações abstraídas da realidade, o que torna as aulas muito mais produtivas e motivadoras aos alunos.

Em contrapartida, as produções acadêmicas resultantes, devidamente verificadas e validadas, vêm sendo incorporadas aos projetos reais em desenvolvimento.

Com o intuito de se obter um melhor aproveitamento desta prática de ensino, baseada no uso de Estudos de Caso, necessita-se de um planejamento detalhado da estratégia utilizada para que teoria e prática fiquem sincronizadas e balanceadas de acordo com a proposta acadêmica de cada disciplina.

Tabela 4: Síntese de Projetos Acadêmicos por Disciplina

Ano	Projeto	Descrição	Disciplinas
2001	SHOP-PORTAL	Portal de um Shopping Aeroportuário	Projeto de Sistemas de Banco de Dados / Tecnologias da Informação
2002	EBADMA	Empresa Brasileira de Administração Aeroportuária	
2003	GMR	Gestão de Mobilidade Rodoviária	
2004	ALC	Apoio à Logística de Cargas	
2005	ARP-J	Avaliação de Riscos de Pessoas Jurídicas	
2006	SIG-D	Sistema de Informações Georreferenciadas Dinâmicas	
2007	SIG-PA	Sistema de Informações Georreferenciadas para Processamento Analítico	
2003	VIGILANTE-TRIPHIBIUS	Protótipo de sistema de software embarcado e de tempo real para uma Estação de Controle e um Veículo Experimental do Tipo carro Anfíbio voador	Sistemas Embarcados de Tempo Real / Qualidade, Confiabilidade e Segurança de Software
2004	PVA-PVT	Protótipo de sistema de software embarcado e de tempo real para um protótipo de Veículo autônomo e um protótipo de veículo tripulado	
2005	VANT-EC-SAT	Protótipo de sistema de software embarcado e de tempo real para um veículo aéreo não tripulado, uma estação de controle e um satélite	
2006/ 2007	VANT-EC-SAME	Protótipo de sistema de software embarcado e de tempo real para um veículo aéreo não tripulado, uma estação de controle e um satélite de monitoramento ecológico.	

A próxima seção apresenta a metodologia adotada nos últimos anos para conduzir a parte prática das disciplinas mencionadas anteriormente.

3 Metodologia de Ensino

A adoção de um exemplo real permite aos alunos vivenciarem o processo de engenharia do sistema escolhido. O projeto acadêmico da disciplina, considerado como avaliação prática, torna-se um desafio para os alunos, que precisam construir um protótipo deste sistema de acordo com as melhores práticas da Engenharia de Software.

A metodologia aplicada ao desenvolvimento dos Estudos de Caso adota a divisão do projeto em aplicativos (subsistemas) que ficam sob responsabilidade de cada aluno. Devido ao tempo disponível em cada semestre letivo das disciplinas, a criação do projeto acadêmico toma por base uma redução de escopo do projeto real, limitando os aplicativos em 5 ± 2 requisitos, a fim de se evitar a explosão combinatorial de requisitos e buscar compatibilidade com a carga horária de aulas práticas. Atualmente, realiza-se o planejamento deste projeto considerando 18 semanas de aula, como tempo máximo de desenvolvimento.

No decorrer do semestre letivo, à medida que o processo de desenvolvimento evolui, os aplicativos de cada aluno vão sendo integrados com outros afins, consolidando-se em aplicativos e equipes maiores. Conforme os níveis de integrações ocorrem, os alunos são clusterizados em grupos formados por 5 ± 2 alunos e respectivos aplicativos. Usando uma abordagem *bottom-up*, são definidos 04 (quatro) níveis de atuação, partindo de pequenos aplicativos individuais até a consolidação em um protótipo de sistema computadorizado. A cada integração entre os níveis, os aplicativos recebem nomes particulares a saber: a) Aplicativos setoriais - do Nível 0

para o Nível 1; b) Aplicativos corporativos - do Nível 1 para o Nível 2; e c) Protótipo do Projeto Acadêmico - do Nível 2 para o Nível 3. Esta prática tem fornecido resultados satisfatórios quanto à distribuição dos trabalhos entre os alunos e ao gerenciamento de atividades de equipes heterogêneas.

Cada aluno responsabiliza-se pelo aplicativo de Nível 0 escolhido até o final das atividades da disciplina, ou seja, mesmo após as integrações, o aluno não perde o foco do seu aplicativo. Dessa forma, a identidade do aplicativo, assim como a atuação do aluno são preservadas durante a evolução dos trabalhos. Neste contexto, os alunos têm a oportunidade de trabalhar de forma individual, nos aplicativos de Nível 0, consolidando e aperfeiçoando os conhecimentos adquiridos, e posteriormente em equipes multidisciplinares, nos aplicativos dos demais níveis, compartilhando experiências e vivenciando situações que eles podem encontrar em suas vidas profissionais.

A partir da formação dos grupos de aplicativos e da definição do escopo de atuação de cada um, os alunos recebem orientações quanto ao processo de desenvolvimento e quanto às ferramentas de apoio aplicáveis. Neste caso, utiliza-se uma customização do Processo Unificado da Rational (*Rational Unified Process - RUP*), detalhando as fases e as atividades a serem praticadas, assim como as iterações envolvidas e os artefatos a serem produzidos, como mostra a

Figura 21. Os alunos assumem diferentes papéis dentro do processo, trabalhando como analistas, programadores, administradores de banco de dados, testadores, entre outros.

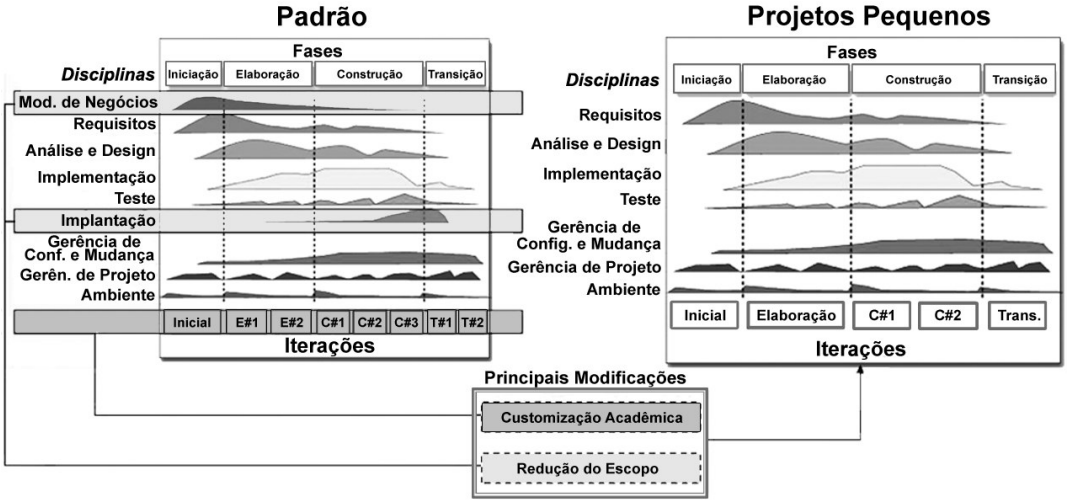


Figura 21: Customização do RUP para Projetos Pequenos [Cunha, 2008]

No que se refere às ferramentas de apoio, os alunos têm a oportunidade de utilizar Ambientes Integrados de Engenharia de Software Ajudada por Computador (*Integrated Computer Aided Software Engineering Environment - I-CASE-E*) de fabricantes como, por exemplo, *IBM Rational*, *ESRI*, *Computer Associates* e *Oracle* que mantêm parcerias acadêmicas com o ITA.

As transições entre as fases do RUP são determinadas pelas entregas das listas de exercícios, que são realizadas em média a cada 04 (quatro) semanas. Na transição da fase de iniciação para elaboração, definiu-se um conjunto mínimo, necessário e suficiente de artefatos customizados para atender às necessidades do Estudo de Caso, a saber: Documento de Visão, Modelo de Casos de Uso, Glossário e Casos de Teste.

Como marco da transição entre a fase de elaboração e de construção, os alunos entregam os modelos de caso de uso, classe, seqüência e atividade (se necessário). Pode-se solicitar também a entrega do artefato Documento de Arquitetura de Software, referente ao Projeto como um todo, que deve ser elaborado em conjunto pelos alunos da disciplina.

A quantidade de iterações na fase de construção depende do número de integrações entre os aplicativos (heurística de 5 ± 2). O final desta fase é marcado pela entrega do protótipo do sistema idealizado e do seu código-fonte devidamente documentado.

Como resultado final do projeto, geram-se relatórios a partir das ferramentas CASE de suporte, fornecendo a documentação completa do processo de desenvolvimento. Para finalizar as avaliações práticas, os alunos realizam apresentações sobre o trabalho executado, abordando as principais dificuldades encontradas e sugerindo melhorias para a metodologia aplicada.

Ao término da disciplina, os alunos são motivados a refletir sobre a importância do desenvolvimento de software segundo um processo bem definido, suportado por um conjunto de métodos, técnicas e ferramentas da Engenharia de Software.

Como prática de ensino, esta metodologia exige um grau de experiência, comprometimento e disponibilidade consideráveis por parte do professor da disciplina. Três pontos-chave devem ser destacados e observados na aplicação desta abordagem: 1) Conhecimento e experiência: visão geral do processo para definição e planejamento do Estudo de Caso; 2) Capacidade de liderança e de motivação: gestão de pessoas trabalhando em equipes e executando tarefas; e 3) Maturidade no processo de avaliação: percepção e cuidado na avaliação do aluno trabalhando individualmente e em grupo.

3.1 Interdisciplinaridade no Projeto Acadêmico

Além dos trabalhos multidisciplinares em cada disciplina, existe também uma relação interdisciplinar entre aquelas oferecidas em um mesmo semestre letivo. No primeiro semestre de 2008, os conceitos interdisciplinares puderam ser praticados com maior abrangência em três disciplinas encadeadas: Projeto de Sistemas de Banco de Dados (CE-240); Tecnologias da Informação (CE-245); e Teste de Software (CE-229), esta última oferecida pela primeira vez no PG/EEC-I, do ITA.

Ressalta-se que o planejamento para cada disciplina deve respeitar as suas especificidades, de forma que o enfoque da avaliação prática esteja em reforçar primeiramente o conteúdo teórico da disciplina, e em segundo plano as práticas recomendadas pela Engenharia de Software.

Esta prática requer ainda que o planejamento das disciplinas envolvidas seja realizado em conjunto entre os professores, com o intuito de garantir sincronismo no cronograma e na realização das atividades, em especial aquelas consideradas interdependentes.

O Estudo de Caso escolhido para o primeiro semestre de 2008 foi o Projeto MONITORAMA II, uma abstração acadêmica do Projeto de Integração e Cooperação Amazônica para a Modernização do Monitoramento Hidrológico (ICA-MMH), em desenvolvimento entre o ITA e a Agência Nacional de Águas (ANA), com apoio da Financiadora de Estudos e Projetos (FINEP).

A partir do Projeto MONITORAMA II, os alunos da disciplina CE-240 ficaram responsáveis pela modelagem lógica e física dos aplicativos de Banco de Dados que

serviram de base para os Protótipos de Sistemas de Informação, desenvolvidos na disciplina CE-245. Já na disciplina CE-229, os alunos ficaram responsáveis pela verificação e validação dos Protótipos de Banco de Dados e de Sistemas de Informação.

Esta dinâmica interdisciplinar pôde ser comprovada com sucesso, principalmente, por alunos que cursaram duas ou três disciplinas, podendo observar enfoques diferentes, de acordo com a fase do desenvolvimento de um sistema computadorizado, assim como as interdependências entre estas fases.

Na próxima seção, detalha-se o cenário do Projeto MONITORAMA II e sua atuação em cada uma das três disciplinas.

4 Cenário do Estudo de Caso - Projeto MONITORAMA II

O Projeto MONITORAMA II tem como objetivo principal o desenvolvimento de um protótipo de um Sistema Automático de Coleta e Armazenamento de Dados Hidrológicos da região Amazônica. Na adaptação para um projeto acadêmico, realizou-se a sua divisão em subsistemas organizados numa estrutura de 04 (quatro) níveis. A Figura 22 ilustra esta divisão, focando os aplicativos abordados nas três disciplinas, assim como as três integrações ocorridas para a construção do Protótipo do Projeto MONITORAMA II.

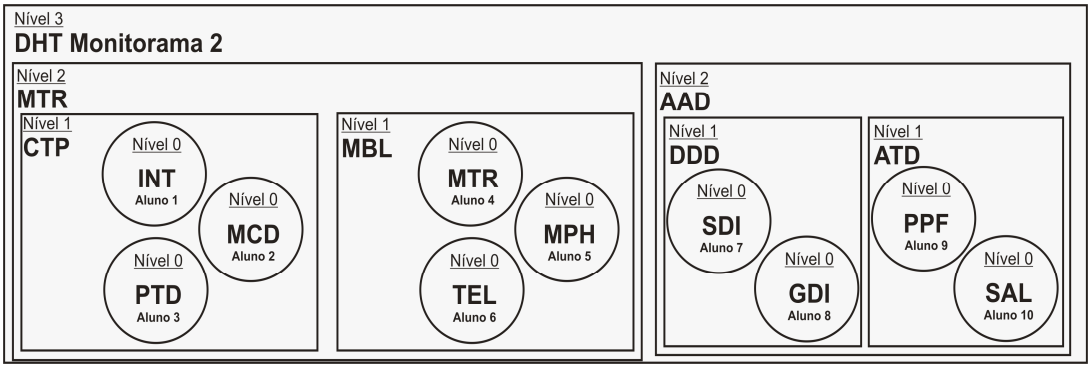


Figura 22: Projeto MONITORAMA II - Divisão em Subsistemas

Três alunos cursaram as três disciplinas no mesmo período letivo e dessa forma utilizaram o mesmo Estudo de Caso para o desenvolvimento de seus trabalhos acadêmicos. Os aplicativos de Nível 0 (INT, MCD e PTD), mostrados na Figura 22, foram escolhidos por estes alunos, que trabalharam de forma interdisciplinar especificamente em cada uma das disciplinas, até que o Nível 3 de integração fosse alcançado, conforme a Figura 23.

O grau de envolvimento dos alunos, bem como os papéis que eles assumiram, durante o desenvolvimento do projeto acadêmico, dependeram das matérias que eles cursaram naquele semestre. Neste Projeto Acadêmico, na disciplina CE-240, modelou-se os aplicativos de Banco de Dados que serviram de suporte para os módulos. Na CE-245, utilizou-se de tecnologias da informação para implementar os módulos, e por fim, na CE-229, aplicou-se técnicas de teste de software para verificar a codificação dos módulos.

Um aluno que se matriculou nas três disciplinas, por exemplo, assumiu o papel: de administrador de banco de dados, na CE-240; de analista e programador, na CE-245; e de testador, na CE-229. Ele teve que realizar atividades associadas a cada um destes papéis para cumprir os requisitos especificados nas listas de exercício de cada

disciplina, conforme as integrações ocorriam. Assim, nem todas as atividades foram executadas para todos os aplicativos, conforme ilustra a Tabela 5.

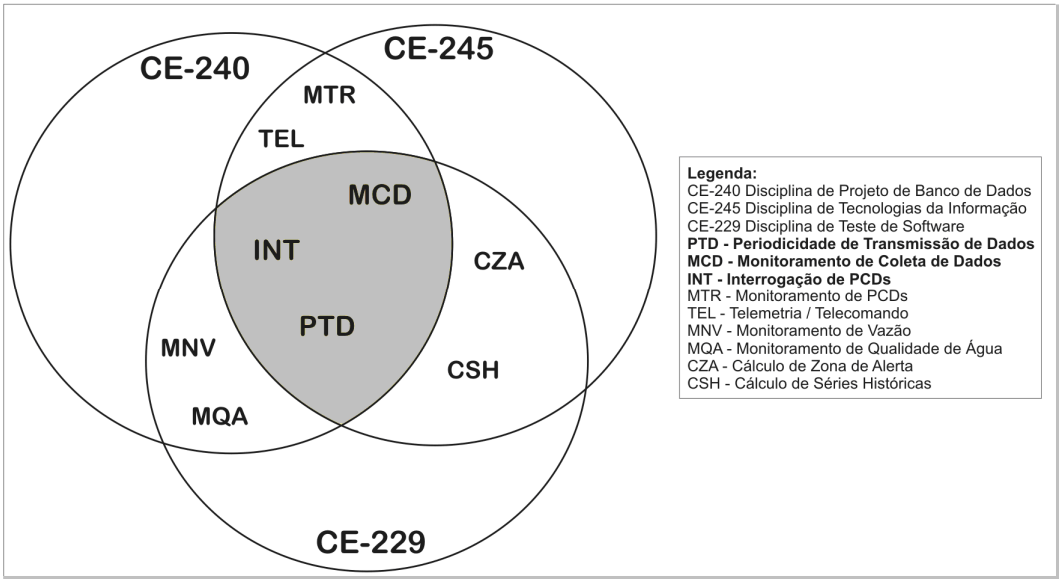


Figura 23: Relação entre Subsistemas e Disciplinas

Tabela 5: Trabalhos por Disciplina de acordo com o Nível de Integração

Disciplina	Lista de Exercício	Foco	Ferramentas de Suporte	Nível de Integração
CE-240	1	Especificação de requisitos do aplicativo de Banco de Dados (BD)	ERwin Data Modeler	0
	2	Criação do modelo lógico de dados do subsistema de BD setorial	ERwin Data Modeler	1
	3	Desenvolvimento do subsistema de software corporativo e do modelo físico de dados	ERwin Data Modeler e Oracle	2 e 3
	4	Documentação do sistema de banco de dados e controle de versão	ERwin Data Modeler; ERwin Data Model Validator e ClearCase	3
CE-245	1	Especificação dos requisitos do subsistema de software	Rational Software Architect (RSA); RequisitePro e ClearCase	0
	2	Criação do modelo lógico do subsistema setorial	ClearCase e RSA	1
	3	Desenvolvimento do sistema de software corporativo	ClearCase e RSA	2 e 3
	4	Documentação do sistema de software	ClearCase; RSA e SoDA	3
CE-229	1	Planejamento e especificação dos casos de uso de teste	RequisitePro e ClearCase	0
	2	Realização dos testes	Rational Manual Tester	1
	3	Documentação dos testes realizados	RSA; ClearCase e SoDA	1

5 Resultados do Projeto Acadêmico

Todas as etapas propostas foram realizadas de forma a comprovar que a metodologia utilizada aplica-se ao desenvolvimento de projetos de alta complexidade, com um número elevado de desenvolvedores. O sucesso destes Projetos Acadêmicos encontra-se diretamente relacionado à heurística dos 5 ± 2 elementos, pois ela impede uma explosão combinatorial que torne impraticável a conclusão das tarefas pelos alunos dentro das 18 semanas previstas para o semestre letivo.

A experiência tem mostrado que os resultados obtidos com a aplicação de problemas reais em sala de aula, através de Estudos de Caso, justificam o esforço necessário com o planejamento didático da(s) disciplina(s) envolvida(s). Comumente ex-alunos entram em contato com o professor da disciplina, principalmente por e-mail, informando que o aprendizado adquirido tem sido bastante útil nas empresas onde passaram a trabalhar. O *feedback* dos alunos também contribui para melhoria contínua da metodologia de ensino utilizada , a partir de críticas, sugestões e recomendações.

Esses alunos sentem-se contagiados pelas boas práticas de Engenharia de Software ensinadas e, por analogia, as utilizam em problemas com os quais se deparam no dia-a-dia. Eles percebem a importância da utilização de métodos, técnicas e ferramentas CASE e as vantagens de seguir um processo bem definido e documentado.

Como contrapartida, os alunos que mais se destacaram são convidados a participar ativamente dos projetos reais, passando a receber auxílio financeiro através de bolsa de estudo. Várias idéias discutidas academicamente também são incorporadas a estes projetos. Como resultado desta prática citam-se ainda produções acadêmicas em trabalhos de pesquisa de nível mestrado e doutorado.

6 Conclusão

Este trabalho mostrou, através de um Estudo de Caso, como a Engenharia de Software pode ser ensinada como conteúdo complementar em disciplinas de um programa de pós-graduação em Engenharia da Computação.

Constatou-se o sucesso da utilização desta metodologia de ensino em projetos acadêmicos envolvendo várias disciplinas. Para isso, fez-se necessário um planejamento adequado para as disciplinas, dentro do escopo do processo de desenvolvimento do Projeto MONITORAMA II. Os resultados obtidos tanto academicamente, quanto em relação ao desenvolvimento real do projeto, justificaram o esforço necessário neste planejamento.

A utilização de projetos reais como exemplos acadêmicos tornou-se viável neste contexto, permitindo aos alunos vivenciarem, antecipadamente, em um cenário fictício, situações que somente presenciariam quando inseridos no mercado de trabalho.

Considerando o histórico bem sucedido de aplicações da metodologia apresentada, os autores recomendam fortemente que práticas de ensino como esta sejam adotadas em outras áreas do conhecimento, pois enriquecem o conteúdo teórico das disciplinas, além de aumentar o grau de participação, motivação e envolvimento de alunos e professores no processo de ensino-aprendizagem.

Referências

COLONESE, E.; LOUBACH, D. S.; CUNHA, A. M. (2007). **Self-adaptive component-based interoperability framework for real-time systems**. In Proceedings of 46th IEEE Conference on Decision and Control, New Orleans.

COLONESE, E.; RAMOS, D. B.; SILVA, C. M. B.; MARIN, L. H. G.; CUNHA, A. M.; DIAS, L. A. V. (2006). **Informações georreferenciadas de sensoriamento remoto: um estudo de caso com coleta de dados utilizando veículos aéreos não tripulados vant**. In Anais do Sensoriamento Remoto nas Forças Armadas (SERFA), São José dos Campos.

CUNHA, A. M. (2008). **Notas de Aula da Disciplina Sistemas Embarcados de Tempo Real** (CE-235). ITA.

CUNHA, A. M.; SANTOS, W. A.; LOUBACH, D. S.; NASCIMENTO, M. R.; NOBRE, J. C. S. (2006). **Applying model driven development in aerospace software prototype with IBM Rational Rose Real-Time**. In Proceedings of the IBM Rational Software Development Conference, Orlando.

LOUBACH, D. S.; RAMOS, D. B.; SAOTOME, O.; CUNHA, A.M. (2008). **Comparing source codes generated by case tools with hand coded**. In Proceedings of 5th International Conference on Information Technology (ITNG): New Generations, Las Vegas.

MARTINS, O.; SANTOS,W. A., FERREIRA, A. S., ANJOS, L. S., CUNHA, A. M. (2005). **A strategy for teaching real time embedded systems at the Brazilian Aeronautical Institute of Technology - ITA**. In Proceedings of 3rd International Conference on Education and Information Systems (EISTA): Technologies and Applications, Orlando.

SILVA, C. C.; CUNHA, A. M. (2004). **Uma metodologia para realização de atividades práticas em cursos de eletrônica e computação: Um estudo de caso**. In Proceedings of World Congress on Engineering and Technology Education (WCETE), Santos.

Usando PBL na Qualificação de Profissionais em Engenharia de Software

Simone C. dos Santos, Maria da Conceição Moraes Batista, Ana Paula Cavalcanti, Jones O. Albuquerque, Silvio Meira

CESAR.EDU – Centro de Estudos e Sistemas Avançados do Recife (C.E.S.A.R)
simone.santos@cesar.org.br, ceca.moraes@cesar.org.br, ana.paula@cesar.org.br, jones@cesar.edu.br, silvio@cesar.org.br

Abstract. This article presents an educational methodology based-on PBL (Problem Based Learning) to improve the effectiveness of learning in software engineering, promoting the ability of students to solve real problems into software factories environments. This methodology has been applied in a professional master course in Software Engineering on three classes of this program, since 2007. The use of PBL showed a growth of the software factories performance and a reduction of the difference among the student´s grades of 3,58/10 for 0,83/10 to the end of the first semester. This course is run by C.E.S.A.R (http://www.cesar.org.br), a research institute with experience in development of innovative software.

Keywords: PBL, Software Engineering, Software Factories, Master Course

Resumo. Este artigo apresenta uma metodologia de ensino baseada em PBL (aprendizagem baseada em problemas) com o objetivo de aprimorar a efetividade do aprendizado em Engenharia de Software, promovendo a habilidade de estudantes para resolver problemas reais dentro de ambientes de fábricas de software. Esta metodologia tem sido aplicada no Mestrado Profissional em Engenharia de Software (MPES) em três turmas, desde 2007. O uso de PBL mostrou um crescimento no desempenho das fábricas e uma redução da diferença entre os rendimentos dos estudantes de 3,58/10 para 0,83/10 ao final do primeiro semestre. Este curso é executado pelo C.E.S.A.R, instituto de pesquisa com experiência em desenvolvimento de software com inovação.

Palavras-chave: PBL, Engenharia de Software, Fábricas de software, Mestrado Profissional

1 Introdução

O crescimento do mercado de Tecnologia da Informação (TI) e seu papel cada vez mais importante na criação de vantagem competitiva das organizações têm gerado uma forte demanda por novas e diferentes maneiras de se produzir software com alta qualidade, flexibilidade e ganhos de produtividade. Em particular, esta demanda se expressa através de requisitos como tempo de resposta ao mercado (time-to-market) de soluções de TI altamente reduzido e necessidade de eficiência e eficácia de soluções, exigindo inovação na aplicação de TI para melhores e mais rápidos retornos. Claramente estes requisitos não podem ser alcançados sem profissionais com competências (conhecimentos, habilidades e aptidões individuais) diferenciadas, capazes de ter uma visão ampla de problemas sob aspectos gerenciais, interpessoais e de negócios, além dos aspectos tecnológicos.

Estudos sobre os objetivos da educação superior em TI têm cada vez mais relacionado o programa de qualificação e sua metodologia de ensino às futuras carreiras profissionais dos estudantes. Em [TYNÄLÄ, 1999], o autor discute que a educação superior deveria ter como objetivos principais o treinamento de profissionais, o aprendizado contínuo e a preparação para a prática profissional. Esta abordagem pode ser aplicada integrando teoria e prática dentro de um currículo que promova a aquisição de conhecimento geral e específico para resolução de problemas reais de mercado.

Neste contexto, a proposta deste artigo é apresentar e discutir os resultados da aplicação do processo instrucional de aprendizagem baseado em problemas reais (PBL - Problem Based Learning) [NING, 1995] em um curso de Mestrado Profissional em Engenharia de Software (MPES) como uma forma eficaz de aprendizagem.

Para a implementação do método PBL, o curso provê um ambiente baseado em Fábricas de Software [MORAES, MEIRA e ALBUQUERQUE, 2003], no qual os estudantes estão imersos em projetos práticos de desenvolvimento de software de forma sistemática e controlada por processos de qualidade. A avaliação do aprendizado é conduzida de forma que o trabalho coletivo e as habilidades individuais sejam complementares e reflitam diretamente no desempenho do estudante. Resultados da aplicação desta metodologia na primeira turma do mestrado são apresentados ao final do artigo.

Este artigo esta organizado em cinco seções. A Seção 2 discute sobre a motivação do uso do método PBL no ensino de Engenharia de Software. A Seção 3 descreve a metodologia proposta para o ensino de Engenharia de Software. Alguns resultados da aplicação da metodologia na primeira turma do MPES são apresentados na Seção 4. Finalmente, a Seção 5 apresenta as conclusões e considerações finais.

2 Aprendizagem baseada em Problemas Reais

Desenvolvido na educação médica na década de 70, o método PBL [NING, 1995] tem sido adaptado em um número crescente de áreas de atuação, incluindo a Engenharia, e em diferentes níveis educacionais [RIBEIRO e MISUKAMI, 2005], [GÜZELIS,2006].

Sob uma ótica bem simples, PBL pode ser definido como um método instrucional que usa um problema para iniciar, direcionar e motivar o aprendizado. Como um método instrucional, PBL é consistente com os princípios da abordagem construtivista, que defende que "*o quê as pessoas entendem é uma função do conteúdo, contexto, atividades e objetivos do aprendiz*" [SAVERY and DUFFY, 1995]. Entretanto, o

desenvolvimento de um processo efetivo para resolução de problemas é só um dos objetivos do PBL. Em [RIBEIRO e MISUKAMI, 2005], os autores enfatizam:
"*Este método também pretende apoiar os estudantes na aquisição de uma base de conhecimento estruturada em torno de problemas da vida real e no desenvolvimento de competências e atitudes, incluindo trabalho em equipe e habilidades de auto-aprendizado, cooperação, ética e respeito aos pontos de vista de outras pessoas*."

Em [PETERSON, 1997], o autor ressalta três importantes critérios que promovem um aprendizado mais eficaz com o uso de PBL:

1. O aprendizado acontece em um ambiente onde os estudantes estão imersos na prática, em atividades em que recebem feedback de seus colegas estudantes e professores;
2. Os estudantes recebem guias e suporte de seus pares, de maneira a promover um ensino multi-direcional envolvendo outros estudantes, professores e monitores, diferentemente do ensino convencional, normalmente unidirecional (professor para estudante);
3. O aprendizado é funcional, a partir de problemas reais.

A adoção de PBL não é uma tarefa fácil, principalmente porque propõe a quebra do paradigma do ensino convencional já tão enraizado em nossa cultura. Exige mudança de postura tanto do professor como do estudante, trabalhando uma forma de aprendizagem altamente investigativa e questionadora, além de uma certa maturidade profissional. Em [WATERS and MCCRACKEN, 1996], os autores descrevem uma experiência do uso de PBL em Engenharia de Software, em um curso de graduação em Sistemas de Gerenciamento da Informação. A classe foi dividida em 5 grupos de 6 pessoas e um projeto de sistema de leilão interativo foi eleito como estudo de caso. Nesta experiência, uma das principais dificuldades já se apresentou na etapa de levantamento de requisitos, quando os estudantes se preocupavam em rapidamente desenvolver o software a partir de requisitos mal levantados e pobremente documentados, ao invés de entenderem bem os problemas a serem resolvidos e suas implicações nos requisitos de software.

Com base nos princípios de Peterson e nas experiências de Waters, este artigo propõe uma metodologia para o ensino de profissionais da indústria de TI, em particular, para os profissionais envolvidos no ensino de pós-graduação, uma vez que estudantes neste nível possuem alguma experiência anterior com desenvolvimento de software e, portanto, com um nível de maturidade diferente dos estudantes que ainda se encontram no estágio de graduação, sedimentando conceitos e fundamentos que serão pré-requisitos em um curso de pós-graduação.

3 Implementação de PBL com Fábricas de Software

A metodologia tem como principal fundamento o uso de fábricas de software para criar o ambiente onde os estudantes estarão imersos na prática e em contínua interatividade com professores e colegas estudantes, como destaca o Princípio 1 de Peterson.

Com o objetivo de sistematizar a criação das fábricas de software, alguns elementos-chave foram identificados com foco nos outros dois princípios de Peterson, como ilustra a Figura 1: (1) Capital Humano e (2) Conteúdo, no que diz respeito ao ensino multidirecional e guias e suporte ao capital humano envolvido e; (3) Projetos Reais, com foco na necessidade de um aprendizado funcional. A partir destes elementos, um processo de construção de fábricas foi definido tendo como principal

referência o trabalho de [MORAES, MEIRA e ALBUQUERQUE, 2003]. As seções subseqüentes discutem brevemente estes aspectos, considerando o contexto do Mestrado Profissional em Engenharia de Software (MPES).

3.1 Elementos-chave na Criação e Execução das Fábricas de Software

A metodologia identifica quatro grupos de capital humano envolvidos no processo e criação das fábricas de software: o *time* que compõe uma fábrica, responsável pela definição do negócio da fábrica e desenvolvimento de software; os *tutores/professores*, responsáveis por ministrar as aulas, executando o papel de consultores especialistas em áreas de sua disciplina e facilitadores no processo de aprendizado; os *monitores de fábrica*, responsáveis pelo acompanhamento de todas as fábricas quanto ao atendimento de objetivos aos quais elas se comprometeram, sob os pontos de vista de processos e produtos; os *clientes*, responsáveis pela demanda do software, validação de requisitos e dos resultados acordados com as fábricas de software.

Figura 1: Elementos-chave na criação e execução das fábricas de software.

Quanto ao conteúdo, as disciplinas constituem a base conceitual da metodologia, subsidiando o time, professores, monitores e clientes em conhecimentos para a resolução dos problemas. No contexto de Engenharia de Software, as disciplinas e suas ementas usam como referência o guia SWEBOK (Software Engineering Body of Knowledge – http://www.swebok.org/), uma iniciativa do IEEE Computer Society. O SWEBOK está subdividido em dez áreas de conhecimento em Engenharia de Software: Gerência de Configuração de Software; Gerência de Projetos de Software; Processo de Engenharia de Software; Ferramentas e Métodos; Qualidade de Software; Requisitos de Software; Design de Software; Construção de Software; Teste de Software; Manutenção de Software. Tendo como base as disciplinas do SWEBOK, o MPES trabalha com sete disciplinas durante a etapa da vivência das fábricas de software, associadas de acordo com a Tabela 1.

O programa foi projetado para executar as disciplinas em módulos semanais de 27 horas/aula cada, sendo 15 horas de aulas conceituais, 12 horas de práticas guiadas e uma avaliação individual ao final da semana. As horas de práticas são utilizadas em atividades para ajudar os alunos a aplicar a teoria vista em sala de aula nos projetos das fábricas de software, e essas tarefas são apoiadas por docentes que atuam como consultores especialistas nas atividades. Um pool de projetos é criado para cada turma que se inicia.

Um projeto, para participar do pool, deve cumprir alguns requisitos: deve ter características de inovação; relevância para a indústria; modelo de negócios aplicado e; ter um processo de desenvolvimento de software que permeie todo o ciclo de desenvolvimento. Vale salientar que, os projetos do pool de projetos são realmente projetos reais, portanto, possuem características diferentes um dos outros, mesmo que concentrados em uma mesma área de conhecimento.

Tabela 1. Relação entre disciplinas do MPES e SWEBOK

MPES	SWEBOK
1. Fábrica de Software	Processos de ES; Ferramentas e Métodos
2. Requisitos e de Interface	Requisitos de Software
3. Gestão de Projetos	Gerência de Projetos de Software
4. Arquitetura	Design de Software
5. Engenharia de Reuso	Design de Software; Construção de Software
6. Interoperabilidade	Design de Software; Construção de Software
7. Verificação e Validação	Testes; Manutenção; Qualidade de Software

Os projetos precisam ainda ter um cliente real envolvido. Os clientes motivam os estudantes a se envolverem no projeto para a solução dos problemas. É importante ressaltar que, se o estudante se engaja em uma autêntica resolução de problemas, então ele é o "dono" do problema. Os aprendizes percebem o problema como real quando existe relevância pessoal de cada indivíduo envolvido [SAVERY and DUFFY, 1995].

3.2 Processo de Construção das Fábricas de Software

O processo de construção das fábricas de software se dá como atividades de avaliação da disciplina Fábrica de Software, na qual os estudantes serão orientados a compor seus grupos e assim estruturar suas equipes para definirem e formalizarem suas respectivas fábricas de software. Este processo é exercitado desde 2003 [MORAES, MEIRA e ALBUQUERQUE, 2003] e tem-se mostrado eficiente como um mecanismo de organização de equipes, definição de um processo de desenvolvimento de software e gestão de projetos associados a estas equipes. O processo de construção é composto por sete macro-atividades:

1. *Divisão dos Estudantes em Equipes*: A metodologia recomenda equipes de 5 a 7 integrantes, cada uma representando uma fábrica de software. A divisão de equipes considera afinidades, competências e habilidades de cada estudante para que se evite concentração e/ou escassez de competências/habilidades em uma mesma equipe. Para cada novo projeto, é necessário alocar os papéis entre os membros da fábrica, tais como: gerente de projeto, arquiteto de software, engenheiro de software, analista de requisitos, engenheiro de testes e de processos, entre outros. Considerando o tamanho da equipe, normalmente cada integrante precisará desempenhar mais de um papel neste processo. É importante ainda salientar que, apesar do seu papel, cada estudante se envolve em todas as etapas do processo de desenvolvimento de software conduzidas pelas disciplinas.
2. *Estudo e Apropriação dos Modelos de Fábricas de Software Disponíveis*: Este estudo se dá em modelos práticos apresentados pelos professores no decorrer do curso. Uma vez entendido os processos de desenvolvimento de software destas fábricas, estas precisarão definir quais características dos processos estudados serão incorporadas a sua realidade.
3. *Definição do Ciclo de Vida do Software na Fábrica*: Esta etapa inclui a definição do processo de desenvolvimento de software, fluxo de atividades, papéis, artefatos e templates. Uma fábrica de software precisa de uma plataforma de processo de desenvolvimento de software configurável baseada em conceitos sólidos assim como as melhores práticas encontradas no mercado.
4. *Organização das Fábricas em sua Infra-estrutura*: Este passo inclui se apropriar da infra-estrutura necessária para o funcionamento da fábrica: site da fábrica,

repositório de templates de artefatos, contratos de nível de serviço, métricas de desenvolvimento e contratação, ambientes de desenvolvimento e de produção da fábrica, garantia da qualidade dos produtos e processos da fábrica.

5. *Definição do Modelo de Negócio da Fábrica*: Aqui são definidos os tipos de licença de software utilizados pela fábrica em seus contratos, a política de propriedade intelectual dos produtos gerados pela fábrica, o modelo de comunicação com os clientes, o modelo de contratação e os templates de contrato comercial e técnico da fábrica.
6. *Definição da Política de Avaliação Interna da Fábrica*: Esta etapa contempla a avaliação dos processos e do capital humano como executor deste processo, bem como definição da política de contratação e renovação/capacitação do capital humano da fábrica.
7. *Política de avaliação do Cliente*: Nesta etapa cada fábrica precisará definir a política de avaliação de satisfação do cliente, definindo critérios de aceitação de produtos e processos, além de estratégias de melhoria para os itens avaliados.

Observa-se, do processo de construção de fábricas de software, que suas atividades apenas guiam o estudante para que ele próprio efetive suas definições e organização de equipes necessárias às suas respectivas fábricas. Assim, tem-se como componente pedagógico, além do aspecto prático-experimental, a inovação, uma vez que o processo não limita nem impõe restrições à criatividade das equipes em suas definições.

Uma vez construídas as fábricas de software, com seus times, projetos e clientes reais, planejamento de disciplinas e professores alocados, as fábricas seguem a execução de seus projetos, acompanhadas pelos professores das disciplinas e pelos monitores das fábricas. Estes últimos avaliam o processo conduzido e produtos gerados, por meio de reuniões periódicas de acompanhamento (status report), análise de artefatos gerados e avaliações individuais de desempenho. O número de reuniões de acompanhamento é definido com base no número de iterações dos projetos de software desenvolvidos nas fábricas, planejados antes do início das aulas pelo monitor de fábrica. Mais detalhes sobre o processo de avaliação de disciplinas e fábricas podem ser encontrados em [CAVALCANTI, A. P. et al., 2008].

A etapa de vivência das fábricas dura seis meses e é concluída com uma apresentação dos projetos para uma banca, simulando uma situação de apresentação para investidores onde as equipes terão que apresentar os resultados de seus projetos sob quatro aspectos: vendas, tecnologia, inovação e capacidade de gerar resultados positivos. Concluída esta etapa, os estudantes passam para etapa de Pesquisa, para elaboração da sua dissertação.

4 Estudo de Caso: Resultados da Aplicação da Metodologia no MPES

O MPES foi implantando em 2007, iniciando sua primeira turma em agosto do mesmo ano. A primeira turma foi composta por 18 alunos com faixa etária média de 26,84 anos e perfis profissionais focados nas áreas de engenharia de testes e engenharia de qualidade. Embora duas novas turmas estejam em andamento, no relato de experiência apresentado nesta seção, só serão consideradas as notas da primeira turma, visto que as fábricas de softwares montadas na segunda turma ainda não concluíram seu processo de avaliação e a terceira turma iniciou suas atividades no mestrado há pouco tempo.

Com o objetivo de validar a aplicação de PBL através da implantação de fábricas de software, algumas métricas foram definidas para verificar a efetividade da metodologia no contexto de educação de disciplinas relacionadas à Engenharia de Software. Além disso, verificou-se a necessidade de identificar as possíveis ações de melhoria no processo de educação definido, de forma que o mesmo possa ser evoluído de maneira incremental, aplicando os princípios de melhoria contínua. De uma forma mais abrangente, busca-se validar o como a aplicação de PBL está contribuindo para a formação dos alunos em uma pós-graduação.

Nesse contexto, foi necessário definir um método para validar os resultados obtidos, de forma que os valores alcançados possam ser analisados para geração de insumos e oportunidades de melhoria para o programa do MPES. Dessa forma, a avaliação da efetividade do processo e resultados obtidos pelos alunos são realizados com base nos resultados coletivos das fábricas, como mostra a Tabela 3, representando as notas aplicadas às fábricas de software no decorrer do semestre, com base no processo de avaliação descrito em [CAVALCANTI, A. P. et al., 2008].

A análise dos resultados coletados é feita considerando a escala descrita na Tabela 2, onde *x* representa o percentual de aumento da primeira à última nota.

Tabela 2: Escala de resultados.

VARIAÇÃO	RENDIMENTO
x>= 70%	Excelente
40%<=x<70%	Satisfatório
x<40%	Ruim

A Tabela 3 representa a evolução das notas coletivas, em uma linha de tempo cronológica, através de 3 formas de avaliação coletiva, realizada tomando como base 4 fábricas de software, onde foram possíveis executar:

- 3 status report, uma avaliação baseada no acompanhamento do processo das fábricas e cumprimento de marcos;
- 1 avaliação independente, focada em avaliação de artefatos produzidos;
- 1 avaliação de alinhamento, foco em desempenho e sintonia entre os membros de um time.

Tabela 3: Análise do rendimento com base em percentual.

FSW	Status Report	Avaliação Independente	Status Report	Avaliação Alinhamento	Status Report	Média	Desempenho 1ª.e última
1	8,58	8,37	9,33	9,28	9,17	**8,95**	**7%**
2	6,17	7,90	8,92	8,20	9,00	**8,04**	**46%**
3	5,00	6,90	7,00	7,90	8,67	**7,09**	**73%**
4	7,08	7,90	8,50	8,24	9,50	**8,24**	**34%**

Os dados obtidos a partir das avaliações deram origem ao gráfico da Figura 2, onde é possível observar que 3 das 4 fábricas obtiveram um constante crescimento ao longo de suas avaliações. A única fábrica que apresentou a nota do último status report inferior a um status report anterior teve um desempenho melhor do que as outras fábricas durante a maior parte do processo de avaliação. Isto mostra que as fábricas evoluem ao longo das avaliações realizadas sobre elas.

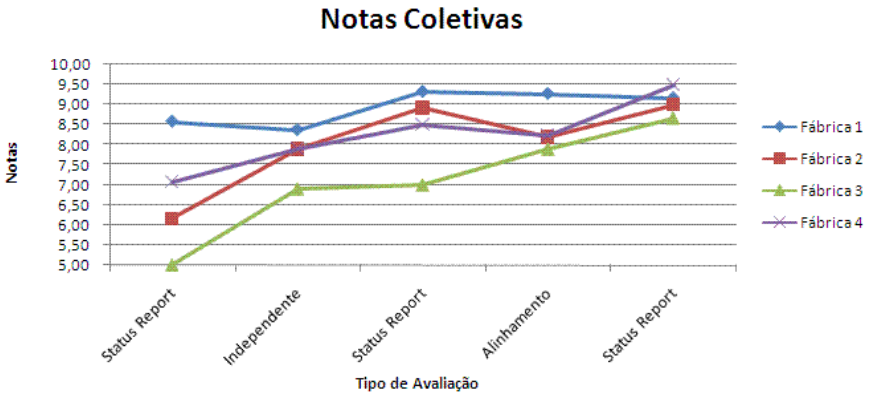


Figura 2: Gráfico de desempenho das fábricas de software da 1ª. turma do MPES.

Pode-se observar também que, no primeiro status report, havia uma diferença relevante entre os desempenhos das fábricas, chegando a 3,58 pontos. No último status report, a maior diferença entre os desempenhos das fábricas ficou em apenas 0,83 pontos (a maior nota foi 9,50 e a menor foi 8,63), o que mostra uma convergência das notas para um padrão de qualidade das fábricas observado ao final das disciplinas do mestrado. Outro dado importante foi o maior progresso observado em uma fábrica, que evoluiu em 3,67 pontos da primeira à última nota, representando um progresso de 73%, um desempenho considerado excelente de acordo com os critérios definidos na Tabela 2. Desta forma, os resultados da primeira turma mostraram que a metodologia está no caminho certo.

5 Conclusões e Considerações Finais

As demandas de mercado por profissionais com qualificação voltadas para a prática profissional e desenvolvimento de competências multidisciplinares têm estimulado o uso de metodologias de ensino diferenciadas. Embora, o método PBL tenha construído sua reputação na área médica, seus princípios mostram-se completamente alinhados às necessidades do ensino de Engenharia de Software, em particular, quando trata-se do desenvolvimento de software, uma vez que esta é uma atividade que exige trabalho em equipe, alta interatividade entre todos envolvidos e vivência prática de projetos em condições reais de mercado. Neste contexto, este artigo apresentou os resultados da metodologia adotada no MPES do C.E.S.A.R, tendo como base a criação de fábricas de software para ensinar a desenvolver software. Neste cenário, os estudantes aprendem fazendo, princípio fundamental da metodologia PBL. A metodologia é inovadora quando define a estratégia de criação e acompanhamento destas fábricas, estabelecendo processos, ferramentas e papéis específicos, permitindo que a prática aconteça de forma sistematizada e estruturada. Já na primeira turma, resultados positivos foram alcançados, como os apresentados na Seção 4.

Atualmente duas turmas estão em andamento. Algumas melhorias foram implantadas a partir da execução da metodologia na primeira turma, em particular quanto à preparação de professores, à participação mais efetiva do cliente nos projetos e ao processo de acompanhamento das fábricas. Neste último, as etapas de avaliações foram ampliadas, novos status reports foram incluídos, tornando o processo mais controlado e interativo.

Finalmente, por não ter encontrado métricas publicadas sobre a aplicação de PBL no contexto deste artigo, não foi possível a comparação com propostas externas. Para se obter uma análise comparativa, os resultados de cada turma serão comparados entre si, a partir de métricas coletadas durante o processo. Com estas informações

poderemos identificar pontos de melhoria na metodologia, que serão objetos de futuras publicações.

7 Referências

CAVALCANTI, A. P.; Santos S. C.; M. C. Moraes; Albuquerque J. O.; and Meira S. R. L. An Evaluation Approach Based on the Problem-Based Learning in a Software Engineering Master Course. Journal of Technology Management & Innovation, Special Issue on Entrepreneurship Education, 2008.

GÜZELIS, C. An Experience on Problem Based Learning in an Engineering Faculty. Turk J Elec Engin, Vol.14, No., p. 67-76, 2006.

MORAES, A. K.O.; and MEIRA, S. R. L. and ALBUQUERQUE, J. O. Open Source Software Factory – Step by Step: A Case Report. In: FIRST INTERNATIONAL CONFERENCE ON OPEN SOURCE COLLABORATIVE DEVELOPMENT PLATFORMS (Héphaïstos), 2006.

NING, C. Undergraduate academic programme: planning, development, implementation and evaluation. J. Engng. Educ., v.11, n.3, p.175-84, 1995.

PETERSON, M. Skills to Enhance Problem-based Learning. Med Educ Online [serial online] 2,3, 1997.

RIBEIRO, L. R. C. and MIZUKAMI, M.G. An experiment with PBL in higher education as appraised by the teacher and students. Interface - Comunic., Saúde, Educ., v.9, n.17, p.357-68, 2005.

SAVERY, JR and DUFFY, TM. Problem based learning: An instructional model and its constructivist framework. Educ Technology,;35(5):31-7, 1995.

TYNÄLÄ, P. Towards expert knowledge? A comparison between a constructivist and a traditional learning environment in the university. J. Educ. Res., v.31, p.357-442, 1999.

WATERS, R. and MCCRACKEN, M. Problem-Based Learning in Computer Science. In: 5TH ANNUAL CONFERENCE ON PROBLEM-BASED LEARNING, 1996.

Utilizando uma ferramenta de gerência de projetos para auxiliar no ensino de Engenharia de Software

Valéria Lelli Leitão[7] Rossana Maria de Castro Andrade

Departamento de Computação – Universidade Federal do Ceará (UFC)

{rossana, valerialelli}@lia.ufc.br

Abstract. Software Engineering (SE) is concerned with all aspects involved in software development. Project management, in the scope of SE, is a practice that aims organization, productivity and software quality. Accordingly, the project manager, in addition to having skills and knowledge related to that, should also use project management tools to help the project development. Hence, it is important to focus also on project management when teaching SE undergraduate and graduate courses. The goal of this paper is to present the experience of using dotProject, a multiproject web manager, as a tool to monitor the students activities in the software development of a practical assignment of a SE course and familiarize these students in a project management environment.

Keywords: Software Engineering, Project Management and Tools

Resumo. A Engenharia de Software (ES) se preocupa com todos os aspectos envolvidos no desenvolvimento de software. A gerência de projetos, no escopo de ES, é uma prática que objetiva organização, produtividade e qualidade do software. Para gerenciar um projeto, o gerente além de ter habilidades e conhecimentos deve utilizar ferramentas que facilitem o andamento do mesmo. Nesse contexto, é importante adicionar a prática de gerência de projeto na disciplina de ES, incluindo o uso de uma ferramenta de gerência de projetos. O objetivo deste artigo é apresentar a experiência do uso do *dotProject*, um gerenciador de *multiprojetos web*, como ferramenta para acompanhar os alunos nas atividades de desenvolvimento de software da disciplina e familiarizá-los em um ambiente de gerência de projetos.

Palavras-chave: Engenharia de Software, Gerência de Projeto e Ferramentas.

[7] Aluna do Curso de Mestrado do MDCC. Bolsista FUNCAP desde 2007.1.

1 Introdução

A Engenharia de Software (ES) deve adotar uma abordagem sistemática e organizada para o desenvolvimento de software e usar ferramentas e técnicas apropriadas dependendo do problema a ser resolvido, das restrições de desenvolvimento e dos recursos disponíveis para um projeto [Sommerville, 2007].

Segundo o PMBOK (Project Management Body of Knowledge) [PMBOK, 2000, p.4], um projeto é um empreendimento temporário com o objetivo de criar um produto ou serviço único. Temporário significa que cada projeto tem um começo e um fim bem definidos. Único significa que o produto ou serviço produzido é de alguma forma diferente de todos os outros produtos ou serviços semelhantes. O PMBOK é um guia na área de conhecimento de gerenciamento de projetos desenvolvido pelo PMI (Project Management Institute) [PMI, 2008] que abrange as seguintes áreas de conhecimento: escopo, tempo, custo, qualidade, recursos humanos, comunicações, riscos e aquisições. Assim, podemos dizer que durante o desenvolvimento de software todos esses conhecimentos e práticas de gerenciamento devem ser atendidos.

Para gerenciar um projeto, o gerente, além de possuir habilidades e conhecimentos, também deve utilizar ferramentas, conhecidas como software de gerência de projeto, que facilitem o controle do mesmo. A utilização dessas ferramentas auxilia o gerente e sua equipe a acompanhar as atividades do projeto, a controlar de forma eficiente o cronograma, colaborando para uma boa integração dentro do projeto e também complementando outras áreas de gerenciamento. Segundo Pressman [Pressman, 2006], o cronograma do projeto e o plano de projeto devem ser acompanhados e monitorados em uma base contínua e isso é feito através das ferramentas de gerência de projetos. Além disso, tais ferramentas fornecem *links* para outras ferramentas que dão suporte para outros processos de desenvolvimento de software.

Verifica-se na literatura o grande uso de ferramentas de gerência de projetos, por exemplo, atualmente, cerca de quatrocentas e duas ferramentas desse tipo podem ser encontradas [Web-Based Software, 2008], dentre as quais, ferramentas livres e comerciais.

Para o contexto de sala de aula a ferramenta de gerência de projetos a ser selecionada deve atender aos seguintes requisitos básicos: software livre com interface *web* e gerenciador de multiprojetos. Com essas características, a ferramenta pode ser instalada sem custos no servidor do laboratório de computação do Departamento da Universidade Federal do Ceará (UFC) e os alunos podem acessar de qualquer lugar através da Internet, possibilitando também ao professor e monitor acompanhar todos os projetos na mesma ferramenta.

Em [Lelli et al., 2006], são avaliadas cinco ferramentas de gerência de projetos de acordo com as áreas de conhecimento do PMBOK mencionadas anteriormente. Dessas cinco, quatro são ferramentas livres. A análise realizada mostrou que existem dois tipos de ferramentas, as do tipo *desktop* que são instaladas localmente no computador e as do tipo *web* que são instaladas em um servidor e podem ser acessadas via Internet. Dentre todas as ferramentas analisadas nesse artigo, a que mais se adequou ao contexto de sala de aula foi o *dotProject* que é livre, *web* e gerenciador multiprojetos. Esse gerenciador permite controlar vários projetos num mesmo ambiente, além de possuir funcionalidades essenciais para o gerenciamento de projetos, tais como gráfico de *Gantt* e relatórios. A versão escolhida da ferramenta para ser utilizada, durante a disciplina de Engenharia de Software, é o *dotProject* 2.0.4 [DotProject, 2008].

Com as características citadas acima, ao utilizarmos o *dotProject* possibilitamos aos alunos de Engenharia de Software praticar o uso de uma ferramenta que propicia o desenvolvimento *online* das atividades necessárias para gerenciar os seus projetos de software.

O objetivo deste artigo é apresentar a experiência prática do uso do *dotProject* como ferramenta para auxiliar no acompanhamento das atividades dos alunos no desenvolvimento de um software na disciplina de Engenharia de Software bem como familiarizá-los em um ambiente de gerência de projetos. A disciplina em questão foi ministrada no semestre 2006.2 para os alunos do curso de Ciência da Computação da Universidade Federal do Ceará (UFC).

Este artigo está organizado da seguinte forma: na Seção 2 são apresentados os trabalhos relacionados com este artigo. Na Seção 3 é descrita a metodologia utilizada. Na Seção 4 os resultados alcançados com o uso da ferramenta *dotProject* na disciplina de ES e, por fim, na Seção 5 é apresentada a conclusão e os trabalhos futuros.

2 Trabalhos Relacionados

Em [Hood, 2006] é descrito uma experiência de ensino no gerenciamento de projetos através de uma ferramenta de simulação de projetos baseada na *web*, *SimProject* [SimProject, 2008]. Durante a disciplina os alunos são organizados em equipes de três a cinco alunos que recebem vários projetos baseados em construção modelada por LEGOs®. A ferramenta *SimProject* cria uma seqüência de tarefas que são divididas em doze marcos e fornece *feedback* quantitativo e qualitativo a cada marco a fim de alcançar os objetivos do projeto. Dessa forma, ela permite aos alunos monitorar as tarefas e ajustá-las de acordo com as lições aprendidas.

Embora esse trabalho aplique conceitos de gerenciamento de projetos através da simulação de vários projetos distribuídos por equipes, ele se difere do proposto neste artigo em vários pontos. Primeiro, a ferramenta utilizada é de gerência de projeto, *dotProject*, e não de simulação. Dessa forma, as equipes criam suas tarefas e seus marcos na ferramenta, levando em consideração o plano da disciplina que contém os artefatos que serão recebidos pelo professor e as suas datas de entrega. Segundo, cada equipe escolherá apenas um projeto que será de software para desenvolver ao longo do semestre e, além disso, o *feedback* das atividades das equipes será fornecido pelo professor e/ou monitor e não pela ferramenta.

3 Experiência com o uso do *dotProject*

A disciplina de ES é obrigatória e é ofertada anualmente para o Curso de Computação. Ela possui quatro créditos, totalizando sessenta horas-aula por semestre e as turmas têm em média quarenta alunos. A metodologia de ensino utilizada na disciplina de ES do Departamento de Computação (DC) da UFC prevê, em primeiro lugar, a disponibilização e uso da ferramenta *dotProject* em um servidor, para que os alunos possam acessá-la a qualquer momento através de um *link* via Internet.

Em paralelo, os alunos são alocados em equipes de quatro integrantes em média e cada equipe escolhe livremente uma aplicação de software. A cada equipe é indicado, pelos membros da mesma, um líder que tem o papel de gerente de projeto. As equipes são cadastradas na ferramenta e cada líder cadastra a sua aplicação como sendo um

projeto. A cada projeto são adicionadas tarefas que devem ser executadas durante a disciplina.

Embora o *dotProject* tenha sido adotado durante a disciplina, os alunos também tiveram a oportunidade de conhecer outras ferramentas de gerência de projetos assim como as suas principais funcionalidades através de aulas teóricas ministradas pelo monitor ou professor. Com isso, os alunos ficaram aptos a perceber funcionalidades importantes que uma ferramenta deve conter para o gerenciamento eficiente de um projeto.

O *dotProject* tem funcionalidades importantes para o gerenciamento de projetos através dos módulos: Companhias, Departamento, Projeto, Tarefa, Calendário, Arquivos, Contato, Fórum, Tickets e Ajuda. A forma como essa ferramenta é utilizada como apoio ao desenvolvimento da aplicação durante o ensino de ES é descrita a seguir:

- No módulo "Companhias" é criada uma companhia para cada equipe, e assim cada uma pode cadastrar o seu projeto. O acesso das equipes é restrito, elas só podem visualizar os próprios projetos (e.g., acessar, ver, adicionar, editar e deletar as tarefas). O professor e seu monitor fazem parte da companhia Engenharia de Software que tem por objetivo avaliar as equipes através de seus projetos e tarefas. As permissões são gerenciadas por eles através dos módulos "User Admin" e "System Admin" que são desabilitados para os alunos. Dessa forma, o professor e seu monitor têm acesso como administradores da ferramenta (adicionar, excluir e gerenciar usuários) e também permissão para acessar todos os projetos e companhias.
- No módulo "Projeto", as equipes adicionam seu projeto e selecionam a companhia da qual fazem parte. No cadastro do projeto existem os campos: descrição do projeto, criador do projeto, datas de início e fim, progresso, status, custo, preferência de cor, dentre outros. O campo progresso se refere ao valor, em porcentagem, do que vem sendo concluído do projeto.
- No módulo "Tarefas", os alunos cadastram as tarefas do seu projeto, colocando a descrição, os responsáveis, as dependências, as datas de início e fim, o status, o progresso, entre outros. Após cadastrar as tarefas os usuários podem gerar os relatórios e o gráfico de *Gantt* que é um gráfico de barras que foca o cronograma do projeto. Essas tarefas já podem ser visualizadas pelos alunos no plano da disciplina de ES que fica disponível na página da disciplina [LIA, 2008].
- No módulo "Calendário", as equipes podem ver as tarefas e os eventos agendados, tendo a opção de visualizarem as atividades a serem desenvolvidas durante todo o mês ou apenas a do dia atual.
- No módulo "Arquivos", as equipes podem anexar qualquer tipo de documento à ferramenta e ainda podem fazer uma breve descrição sobre o mesmo.
- O módulo "Tickets" pode ser comparado a um mural de bilhetes que pode funcionar como um *bug report* da ferramenta, por exemplo, o usuário pode notificar o administrador quando ele não conseguir fazer o *upload* de algum arquivo ou gerar o gráfico de *Gantt*.
- No módulo "Fórum", as equipes podem criar discussões entre os seus integrantes para esclarecer dúvidas sobre alguns assuntos relacionados com o desenvolvimento da aplicação.

- No módulo "Contatos", cada equipe tem acesso aos dados dos seus integrantes, tais como: nome, telefone, endereço, e-mail, dentre outros.
- Finalmente, no módulo "Ajuda", existe apenas uma indicação para o site da ferramenta, pois não existe uma documentação *offline*.

No caso de 2006.2, o foco consistiu no desenvolvimento de uma aplicação para dispositivos móveis ao longo da disciplina. Essa aplicação envolveu um cliente no dispositivo móvel e um servidor, além de tarefas de definição da aplicação, planejamento do projeto, estudo de viabilidade, especificação de requisitos, projeto da aplicação, relatório técnico sobre padrões de software, implementação e testes da aplicação, e apresentação final do projeto. Vale ressaltar que neste caso, a nota do projeto correspondia a dez pontos (cada tarefa tem uma nota atribuída) e equivalia a um terço da nota final dos alunos.

Com base na metodologia descrita, apresentamos na próxima seção os resultados obtidos com sua aplicação em sala de aula e demais considerações a respeito dessa experiência.

4 Resultados Alcançados e Discussões

Através das funcionalidades do *dotProject* descritas anteriormente é feita a captura dos resultados obtidos com o uso da ferramenta na disciplina de ES. Esses resultados são avaliados de acordo com algumas áreas de conhecimento do PMBOK, os quais são listados a seguir:

Gerenciamento dos recursos humanos:

- Controle dos alunos na ferramenta: é feito através da simulação de um ambiente real de multiprojetos, que inclui a gerência de permissões de acesso de usuários. Esse gerenciamento contribui para o aprendizado, além disso, para o exercício da ética em ES, pois não é permitido aos alunos visualizarem os projetos das equipes que não pertencem.
- Controle de atribuições de tarefas: todos os alunos são cadastrados na ferramenta para que o gerente de projeto distribua as tarefas aos membros da sua equipe. A princípio, apenas o líder de cada equipe seria cadastrado, porém no momento de delegar as tarefas para os responsáveis, ele constatou que os demais integrantes não aparecem na lista e comunicou ao professor e monitor. Dessa forma, surge a necessidade de cadastrar todos os envolvidos no projeto, ou seja, todos os alunos da disciplina. Nesse contexto, os líderes perceberam a necessidade de atribuir tarefas aos demais membros da equipe. Aqui estimulamos o trabalho em conjunto para que aprendam a dividir as tarefas e responsabilidades durante a execução do projeto.

Gerenciamento do escopo e prazo:

- Controle de requisitos e prazos: A companhia Engenharia de Software (descrita na Seção 2), representada pelo professor e seu monitor, funciona como um cliente que solicita vários projetos e verifica se o projeto está atendendo aos seus requisitos. Como em um ambiente real, o cliente negocia mudanças nos requisitos e os alunos nos prazos de entrega. Portanto, as equipes percebem que

nem sempre o cronograma é definitivo, e com isso aprendem a gerenciá-los durante o projeto fazendo as alterações necessárias.

- Controle de Execução/Prazos: devido à atualização do progresso das tarefas ser feita de modo manual, a equipe tem que controlar essa atualização à medida que vai executando as mesmas. Caso não faça esse controle, a ferramenta mostrará o projeto e essas tarefas em atraso. Essa situação induz os alunos a ficarem atentos ao prazo de cada tarefa e conseqüentemente do projeto. Eles aprendem a gerenciar o cronograma do projeto e percebem a importância de atualizar as informações na ferramenta. Por exemplo, pode existir uma tarefa que para ser executada dependa do término de alguma outra tarefa, o responsável por essa tarefa só poderá iniciá-la quando souber que a anterior foi finalizada, portanto, o uso do *dotProject* facilita a comunicação entre os integrantes das equipes durante o projeto.
- Acompanhamento de desempenho das equipes: O gráfico de *Gantt* é muito útil para acompanhar o progresso da equipe em relação às atividades desenvolvidas ao longo do projeto assim como o prazo estabelecido para cada tarefa. Os tipos de relatório gerados pela ferramenta, tais como: "alocação das horas dos usuários", "tarefas completadas", "tarefas atrasadas", "tarefas alocadas por usuários" também auxiliam no acompanhamento do desempenho das equipes. Nesse contexto, o professor e o monitor podem entrar em contato com as equipes que estão com as tarefas em atraso para verificar o motivo da pendência e ajudá-las nas suas dificuldades.

Gerenciamento de comunicação:

- Organização de atividades: o módulo "Calendário" exibe o período que as tarefas serão realizadas. Esse recurso auxilia os alunos a organizarem suas atividades para que o projeto seja entregue no prazo.
- Revisão de documentos: a funcionalidade do módulo "Arquivos" facilita muito a correção de alguns trabalhos da disciplina, por exemplo, a rede de atividades que a ferramenta não gera. Os alunos podem utilizar outro tipo de software para fazê-la e depois anexam ao *dotProject*. Essa funcionalidade permite que a documentação fique disponível na ferramenta. Isso facilita a revisão da documentação pelo professor ou pelo monitor ou pela própria equipe.
- Controle de comunicação: o módulo "Contatos" contribui para a comunicação durante a execução das atividades, pois contém as informações importantes dos alunos como *e-mail* e telefone. Como a ferramenta possui notificação automática via *e-mail*, assim que um líder atribui uma tarefa a um membro da equipe ou até mesmo se a tarefa é alterada por outro integrante, os responsáveis pela tarefa recebem um *e-mail* de aviso.
- Notificação de problemas: a característica do módulo "Tickets" permite aos alunos notificarem o professor e seu monitor quando houver algum problema técnico na ferramenta. Por exemplo, caso um aluno não consiga fazer o *upload* de um arquivo na ferramenta, ele pode enviar a mensagem que ficará salva no *dotProject* e os administrados da ferramenta serão avisados.

Contudo, algumas deficiências foram identificadas durante o uso da ferramenta. Por exemplo, ela não possui documentação *offline*. Devido a isso, as equipes tiveram dificuldades de usar a ferramenta. Para suprir essa necessidade, aulas práticas e teóricas sobre o *dotProject* devem ser ministradas para os alunos da disciplina.

Como é feito o gerenciamento das permissões de acesso, as equipes não podem interagir entre si. Na ferramenta essa funcionalidade só é permitida entre os membros da mesma equipe. A deficiência é suprimida através de outros métodos adotados (e.g., lista de discussão) que colaboram para essa interação.

Para o uso da ferramenta tivemos um trabalho inicial que demandou muito tempo porque tivemos que cadastrar quarenta e sete alunos distribuídos em catorze equipes na ferramenta, criando *login* e senha temporária e atribuindo cada aluno a sua companhia e projeto. Contudo, depois dessa fase inicial ficou mais fácil acompanhar o progresso dos alunos.

Quanto ao uso da ferramenta pelos estudantes, das catorze equipes formadas, quatro não atualizavam com freqüência o cronograma. Nenhuma equipe utilizou o módulo "Fórum" da própria ferramenta, isso se deve ao fato de preferirem a lista de discussão da disciplina. Todas as equipes utilizaram o módulo "Arquivos", embora algumas também enviassem os trabalhos para o e-mail do professor e/ou monitor. Nenhuma equipe utilizou o módulo "Tickets", quando acontecia algum problema, os alunos sempre preferiam se reportar à lista de discussão.

Um problema técnico ocorrido durante a disciplina foi que o servidor no qual a ferramenta estava instalada deu problema e alguns trabalhos não apareciam mais no módulo "Arquivos". Como algumas equipes entregaram seus trabalhos somente através desse módulo tivemos que recuperar os arquivos através de uma cópia de segurança que mantínhamos no servidor.

Quanto ao aproveitamento dos alunos, 55% deles obtiveram nota superior ou igual a 7.0 (sete) no projeto. A principal causa identificada para esse rendimento foi que a utilização de uma ferramenta de gerência de projetos centralizada e acompanhada pelo professor e monitor acarretou em um monitoramento mais rigoroso nos projetos.

Outro motivo identificado foi a grande dificuldade dos alunos na entrega das tarefas do projeto, pois ele é composto de muitas tarefas e algumas equipes não entregaram todas. Um fato ocorrido foi que em duas equipes, um dos integrantes que era responsável por fazer o *upload* da tarefa no *dotProject*, não fez. Dessa forma, toda a equipe foi prejudicada, baixando a nota da equipe no projeto. Os alunos alegaram que não entregaram por esquecimento e isso mostrou que algumas equipes não estavam gerenciando o cronograma de suas atividades na ferramenta.

Com intuito de avaliar os resultados apresentados, foi feita uma comparação do desempenho das equipes na implementação do projeto com o semestre em que não foi utilizado o *dotProject* como ferramenta para acompanhar as atividades do projeto, no caso 2008.1. Embora em 2007.1, o *dotProject* não tenha sido adotado, não enquadramos esse semestre porque a tecnologia utilizada para o desenvolvimento das aplicações não foi a mesma, os alunos desenvolveram aplicações *web* em vez de aplicações para dispositivos móveis (cliente e servidor).

No semestre de 2006.2, das catorze equipes formadas, nove conseguiram implementar a aplicação com sucesso (64,30%), apenas três (21,42%) não conseguiram implementar a aplicação e duas equipes (14,28%) implementaram apenas a parte do servidor.

Já no semestre 2008.1 ficou a critério de cada equipe escolher livremente uma ferramenta de gerência de projetos para fazer o acompanhamento das suas atividades. Nesse caso, verificou-se que das doze equipes formadas, apenas cinco (41,66%)

conseguiram implementar a aplicação com sucesso e três (25%) implementaram somente a parte do servidor.

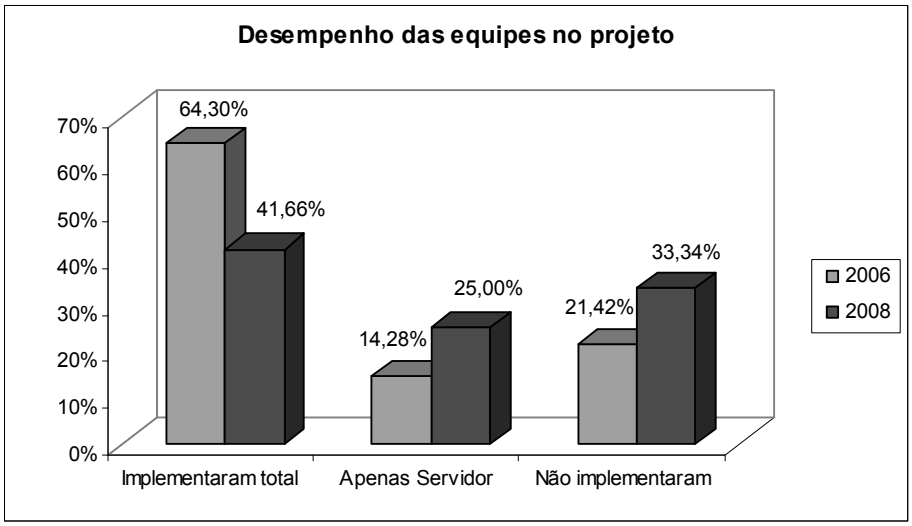


Figura 1 – Desempenho das equipes na implementação da aplicação

Conforme os dados apresentados no gráfico da Figura 1, percebemos que o desempenho das equipes em relação à implementação da aplicação em 2008.1 diminuiu. A causa identificada para esse rendimento foi o monitoramento descentralizado das atividades dos alunos proporcionado pelo uso de várias ferramentas de gerência de projetos. Assim, observa-se que usar uma ferramenta de gerência de projetos que não possibilite ao professor e monitor acompanhar a atividades das equipes via *web* e gerenciar todos os projetos ao mesmo tempo leva a um controle deficiente que não detectará possíveis problemas que podem ocorrer durante o projeto, causando atraso no cronograma e, conseqüentemente a não implementação da aplicação.

5 Conclusão

A utilização do *dotProject*, uma ferramenta de gerência de projetos, durante a disciplina de ES, para acompanhar as atividades dos alunos à medida que eles desenvolvem uma aplicação, mostrou-se positiva, proporcionando aos alunos um conhecimento prático na área de gerência de projetos, abrangendo as principais áreas de conhecimento do PMBOK, e experiência no uso de uma ferramenta automática para gerenciar um projeto de software.

Já para o professor e monitor, o uso da ferramenta favoreceu a identificação das dificuldades e sucessos das equipes na execução das tarefas, além de ajudar a estabelecer a comunicação com os alunos e facilitar a avaliação final das equipes, garantindo assim, uma melhoria no aprendizado da disciplina.

Percebe-se que um monitor para auxiliar muitas equipes dificulta esta atividade. Portanto, para suprir essa deficiência, pretende-se acrescentar mais monitores no processo de acompanhamento dos alunos no projeto, os quais podem ser alunos de mestrado ou doutorado cursando Estágio a Docência.

6 Agradecimentos

Agradecemos a colaboração das alunas de Doutorado Maria de Fátima Costa Souza e Fabiana Gomes Marinho pela revisão textual.

7 Referências

DOTPROJECT. The home of dotProject – the Open Source Project Management tool. Disponível em: <http://www.dotproject.net>. Acesso em: 05 ago. 2008.

HOOD, J. D.; HOOD, C. S. "Teaching software project management using simulations", In proceedings of ACM's Eleventh Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE), New York, v. 38, n. 3, p. 289-293, June 2006.

LELLI, V. L.; DINIZ, E.; ANDRADE, R. M. C. Ferramentas de Gerenciamento de Projetos. In: Encontro de Iniciação à Pesquisa, 12., 2006, Fortaleza. Resumos... Fortaleza: UNIFOR, 2006. p. 257.

LIA. Laboratório de Computação do Departamento de Computação da Universidade Federal do Ceará. Disponível em: <http://disciplinas.lia.ufc.br/es062/>. Acesso em: 13 ago. 2008

PMBOK. Um Guia do Conjunto de Conhecimentos em Gerenciamento de Projetos – Guia PMBOK (Project Management Body of Knowledge). Tradução por PMI. 2ed. EUA: 2000.

PMI. Project Management Institute – Making project management indispensable for business results. Disponível em: <http://www.pmi.org>. Acesso em: 05 ago. 2008.

SIMPROJECT. SimProject - A project management simulation for classroom instruction. Disponível em: <http://84.40.31.176/SimProject2/Index.aspx>. Acesso em: 22 set. 2008.

SOMMERVILLE, Ian. **Engenharia de Software**. Pearson Education, 2007. 568 p.

PRESSMAN, Roger. S. **Software Engineering - A practitioner's Approach**. McGraw-Hill, 2006. 720 p.

Uma Proposta de Metodologia para o Ensino de Engenharia de Software

Rossana Maria de Castro Andrade Fabiana Gomes Marinho[8] Valéria Lelli Leitão[9] Lincoln Souza Rocha[10]

Departamento de Computação – Universidade Federal do Ceará (UFC)

{rossana,fgmarinho,valerialelli,lincoln}@lia.ufc.br.br

Abstract. In some Computer Science undergraduate courses, Software Engineering (SE) has been taught with insufficient hours to cover all theoretical concepts. Thus, the amount of concepts taught in class increases, leaving few time for discussions and practical work. This paper presents a practical-theoretical teaching methodology that has been adopted in the SE course of the Computer Science (CS) undergraduate course of the Federal University of Ceará (UFC). This methodology consists of including, in parallel to the theoretical part, the development of a software project. The proposed methodology facilitates the understanding of the concepts presented, in class and can also improves the student performance.

Keywords: Software Engineering, teaching methodology.

Resumo. Em alguns currículos de Ciência da Computação, a disciplina de Engenharia de Software (ES) é ministrada com créditos insuficientes para cobrir todos os conceitos da área. Isso causa um excesso de conteúdo abordado em sala de aula, não deixando espaço para discussões e trabalhos práticos. Este artigo apresenta uma metodologia de ensino teórico-prática adotada na disciplina de ES do Departamento de Computação (DC) da Universidade Federal do Ceará (UFC) que consiste em incluir, em paralelo ao ensino teórico, o desenvolvimento de um projeto de software. Com o uso dessa metodologia, espera-se facilitar o entendimento dos conceitos teóricos apresentados em sala de aula bem como melhorar o desempenho dos alunos.

Palavras-chave: Engenharia de Software, metodologia de ensino.

8 Aluna do Curso de Doutorado do Mestrado e Doutorado em Ciência da Computação (MDCC). Bolsista CNPQ desde 2008.1 – Processo Nº 552924/2008.3.

9 Aluna do Curso de Mestrado do MDCC. Bolsista FUNCAP desde 2007.1.

10 Aluno do Curso de Doutorado do MDCC. Bolsista CNPQ desde 2008.1 – Processo Nº 550924/2007-8.

1 Introdução

Segundo Piaget (PIAGET,1975), a educação permanente e a renovação incessante do conhecimento são extremamente necessárias, no entanto, o essencial não está somente em aprender, mas sim em aprender a aprender. A idéia proposta por Piaget, afirma que o que caracteriza a aprendizagem é o movimento de um saber fazer a um saber. Neste contexto, é enfatizada a necessidade de que a relação pedagógica entre professor e aluno seja elaborada com base metodológica e planejamento adequado. O professor deve ser responsável por um esforço construtivo, agrupando teorias modernas de aprendizagem.

De acordo com Demo (DEMO,1997), o aluno constrói o conhecimento através da interação com o que está sendo apresentado, portanto, cabe ao professor conduzir uma metodologia de ensino que enfatize a prática do conteúdo ensinado e estimule as idéias dos alunos. Seguindo essa mesma linha de pensamento, Fiorentini (FIORENTINI, 2002) e Pinheiro e Gonçalves (PINHEIRO e GONÇALVES, 2001), afirmam que o professor precisa utilizar técnicas de ensino dinâmicas e adequadas aos interesses dos alunos, com o objetivo de conquistar sua participação durante as aulas, ou seja, o professor precisa desafiar seus alunos de forma que eles busquem soluções para os problemas propostos.

A Engenharia de Software (ES) é uma disciplina relacionada com todos os aspectos da produção de software, desde os estágios iniciais até a sua manutenção, envolvendo não apenas os processos técnicos do desenvolvimento, como também as atividades de gerenciamento de projeto e a utilização de ferramentas, métodos e teorias que apóiem a sua produção (SOMMERVILLE, 2007). Portanto, ensinar todo o conteúdo da disciplina de ES por meio de aulas expositivas tradicionais pode dificultar o aprendizado dos alunos, devido à grande quantidade de informações na teoria envolvida.

A partir do enfoque educacional mencionado anteriormente é apresentada neste artigo uma metodologia de ensino proposta para a disciplina de ES do Departamento de Computação (DC) da Universidade Federal do Ceará (UFC). Este artigo apresenta ainda a avaliação dos resultados obtidos com a utilização da metodologia de ensino proposta desde 2005.

O restante deste artigo está organizado em cinco seções. Na seção 2 são apresentados os trabalhos relacionados e são destacadas as principais diferenças identificadas. Na seção 3 a metodologia de ensino proposta para a disciplina de ES é descrita. Na seção 4 os resultados obtidos com o uso da metodologia são avaliados. Na seção 5 as conclusões e os trabalhos futuros do trabalho são apresentados.

2 Trabalhos Relacionados

Em Aguiar (AGUIAR, 2004) é apresentado um modelo de processo de desenvolvimento de software denominado *easYProcess*. Este processo tem como objetivo auxiliar na gerência do desenvolvimento de aplicações em disciplinas da Engenharia de Software na graduação do Departamento de Sistemas e Computação da Universidade Federal de Campina Grande (UFCG). Nessa metodologia os conceitos teóricos são explorados por meio do desenvolvimento de um projeto de software. Porém, a principal diferença identificada com relação à metodologia proposta neste artigo está no fato dos alunos terem de utilizar um único modelo de processo de desenvolvimento. Além disso, os temas das aplicações são previamente selecionados pelos professores do próprio departamento.

Em Alves (ALVES, 2006) uma metodologia de ensino é proposta para as disciplinas de Engenharia de Software da Universidade Federal do Pampa (UNIPAMPA). Nesta metodologia os alunos também praticam os conceitos apresentados através do desenvolvimento de um projeto de software. Entretanto esta metodologia difere da metodologia proposta neste artigo em dois pontos: o modelo de processo de software é definido previamente pelo professor, além disso, o projeto é iniciado na disciplina de Análise e Projeto de Sistemas I e concluído na disciplina Análise e Projeto de Sistemas II, tendo, portanto, uma duração de dois semestres letivos. O objetivo desta estratégia é integrar as duas disciplinas e seus professores.

3 A Metodologia de Ensino

Na metodologia proposta, as atividades são divididas em duas categorias: intra e extra-classe. As atividades intra-classe são realizadas em sala de aula e englobam aulas expositivas, incluindo também atividades em classe como exercícios e discussão em grupo. As atividades extra-classe, por sua vez, são aquelas desenvolvidas fora do espaço físico da sala de aula, como forma complementar de ensino. Para realizarem as atividades extra-classe os alunos são organizados em equipes de no máximo quatro alunos e suas execuções contam com o auxílio de monitores. A cada monitor são alocadas três ou quatro equipes.

As atividades realizadas extra-classe têm por objetivo o desenvolvimento de um projeto de software, obtendo como produto final uma aplicação para dispositivos móveis (*e.g.*, celulares, *palms*, PDAs). O domínio de dispositivos móveis foi selecionado como alvo para o desenvolvimento por tratar-se de uma área da computação que engloba tecnologias recentes, estimulando os alunos para a aplicação dos conceitos apresentados no decorrer da disciplina.

Cada equipe é livre para escolher o modelo de processo de desenvolvimento, as ferramentas de desenvolvimento, a ferramenta de gerência de projeto e as linguagens de programação que mais se adequam às características do projeto e dos membros da equipe.

Os projetos para dispositivos móveis, em geral, envolvem um cliente e um servidor. O cliente pode ser desenvolvido em Java Micro Edition (JME), *Binary Runtime Environment for Wireless* (Brew), SuperWaba, C++, dentre outras, ficando a cargo da equipe selecionar a linguagem que considera mais apropriada. Da mesma forma, a linguagem utilizada no desenvolvimento do servidor é selecionada pela própria equipe (e.g., PHP, JSP/Servlet). A comunicação entre cliente e servidor, por sua vez, pode ser feita por meio das tecnologias *Bluetooth*, *General Packet Radio Service* (GPRS), dentre outras. Além disso, a utilização de padrões de projeto (GAMMA, 1994), bem como o uso de *frameworks* (GAMMA, 1994) é exigida.

Para facilitar o entendimento do domínio e das tecnologias envolvidas, o professor disponibiliza um calendário de aulas extras a serem ministradas pelos monitores e/ou especialistas nas linguagens ou ferramentas utilizadas. Em geral, aulas sobre UML, JME, padrões de projeto, ferramentas de gerência de projeto e *frameworks* são ministradas. Entretanto, de acordo com as necessidades específicas de cada grupo, um acompanhamento mais direcionado é empregado pelos monitores como forma de favorecer o desempenho dos alunos e o desenvolvimento das aplicações em tempo hábil, uma vez que as mesmas devem ser conduzidas no período de um semestre letivo.

Neste contexto a metodologia de ensino proposta está baseada em cinco características básicas que são descritas a seguir:

1. **Atividades em grupo:** o trabalho em grupo é um dos principais alicerces da metodologia. Os grupos são montados de acordo com o tema selecionado para as aplicações. Cada grupo é composto por quatro integrantes e constitui uma equipe de projeto, que é composto por um líder, que representa o gerente do projeto, e pelos demais integrantes, que representam os demais papéis necessários para o desenvolvimento da aplicação. As atividades de desenvolvimento e gerenciamento do projeto são realizadas pelos alunos, exercitando habilidades como liderança, cooperação e articulação.
2. **Interação com o monitor e com o professor:** a interação dos integrantes do grupo com os monitores e com o professor acontece por meio de reuniões presenciais. Além disso, a disciplina possui uma lista de discussão e um endereço na *web* onde podem ser encontrados: o plano da disciplina com cronograma, uma lista das atividades solicitadas, as notas das avaliações e todas as notas de aula e material didático extra como fonte de pesquisa. Todas as atividades realizadas pelos alunos são acompanhadas pelo professor e pelos monitores, com o objetivo de colaborar com a aprendizagem do grupo e esclarecer dúvidas sobre o conteúdo ou sobre as etapas do processo de desenvolvimento. Além disso, os alunos são orientados com relação às atividades a serem realizadas e sobre a utilização das ferramentas necessárias. Os monitores selecionados são alunos capacitados no conteúdo da disciplina.
3. **Uso de ferramentas:** os alunos são estimulados a utilizarem as ferramentas de gerência de projeto e as ferramentas de desenvolvimento adequadas para a elaboração dos artefatos pertencentes a cada uma das fases do processo de desenvolvimento. As principais ferramentas disponíveis para uso são apresentadas para os alunos em horário complementar (extra-classe) por meio de palestras e seminários realizados por especialistas na área que podem ser monitores e/ou alunos de mestrado e doutorado em Estágio a Docência.
4. **Prática de um modelo de processo de software:** um modelo de processo de software é selecionado por cada equipe de projeto de acordo com a aplicação a ser desenvolvida. No entanto, independente do modelo de processo selecionado, a disciplina possui um conjunto de artefatos básicos que devem ser entregues por todas as equipes nos marcos pré-estabelecidos pelo professor. Esse conjunto de marcos é composto pelos principais artefatos resultantes de cada etapa dos modelos de processo de desenvolvimento existentes na literatura e tem como objetivo controlar o tempo de desenvolvimento das aplicações das equipes e auxiliar nas atividades de gerência da disciplina.
5. **Avaliação equilibrada:** a avaliação é feita por meio de uma média aritmética entre as notas de duas provas e de um projeto de desenvolvimento. As provas buscam avaliar os aspectos teóricos da disciplina, enquanto o projeto tem por objetivo focar na aprendizagem colaborativa entre os membros do grupo, bem como, avaliar a aplicação prática dos conceitos apresentados em sala de aula. Nas provas, os alunos recebem notas que variam de zero (0) a dez (10), enquanto no projeto, os

alunos são avaliados na medida em que os artefatos de cada etapa do modelo de processo de desenvolvimento do projeto são produzidos. A nota para cada artefato é avaliada em dois estágios. No primeiro estágio, uma nota é aplicada pelo monitor a partir da análise dos seguintes elementos: respeito à data de entrega do artefato, uso de ferramenta para geração do artefato, uso de ferramenta para gerência dos recursos e participação da equipe nas reuniões presenciais com os monitores. Em seguida, reuniões são realizadas entre os monitores e o professor com objetivo de repassar a avaliação e o desempenho das equipes. No segundo estágio, a nota atribuída pelo monitor é revisada pelo professor da disciplina. Ao final da disciplina, o projeto final e os artefatos gerados são apresentados pelos alunos em sala de aula. Além dos resultados alcançados em cada uma das etapas, a apresentação inclui a demonstração da aplicação final. Tanto a apresentação quanto a demonstração são avaliadas seguindo o mesmo procedimento dos demais artefatos.

As Tabelas 1, 2, 3, 4, 5 e 6 apresentam o conjunto mínimo de artefatos exigidos e o objetivo de cada um deles, classificados de acordo com as etapas do processo de desenvolvimento.

Tabela 1. Conjunto de artefatos de Gerência de Projetos

Gerência de projeto	
Artefato	**Objetivo**
Plano de Projeto	Estabelece os recursos disponíveis, a estrutura analítica do projeto, os riscos a serem monitorados e os mecanismos de acompanhamento do modelo de processo escolhido
Cronograma	Estabelece as dependências entre as atividades, o prazo estimado necessário para atingir cada marco e a alocação dos recursos nas atividades

Tabela 2. Conjunto de artefatos de Elicitação e Análise de Requisitos

Elicitação e Análise de Requisitos	
Artefato	**Objetivo**
Estudo de Viabilidade	Define se a aplicação contribui para os objetivos gerais da organização e a viabilidade de implementação considerando restrições de tecnologia, custo, prazo e integração com outros sistemas existentes
Especificação de Requisitos	Estabelece o que a aplicação deve implementar. Inclui os requisitos de usuário e uma especificação detalhada dos requisitos do sistema
Especificação de Casos de Uso	Detalha os requisitos da aplicação por meio da descrição de fluxos básicos e fluxos alternativos que descrevem seqüências de eventos de um ator para completar um processo.

Tabela 3. Conjunto de artefatos de Design

Design	
Artefato	**Objetivo**
Especificação de Arquitetura	Apresenta uma visão geral de alto nível da arquitetura prevista do sistema, mostrando a distribuição das funções e componentes nos módulos do sistema.
Projeto da Aplicação	Detalha como as funcionalidades serão fornecidas pelos componentes da aplicação por meio do diagrama de classes, diagramas de seqüência e diagrama de estados

Relatório de Padrões de Software	Identifica os padrões de projeto utilizados durante o desenvolvimento da aplicação

Tabela 4. Conjunto de artefatos de Implementação

Implementação	
Artefato	**Objetivo**
Código	Apresenta o código fonte resultante da implementação da aplicação

Tabela 5. Conjunto de artefatos de Testes

Testes	
Artefato	**Objetivo**
Plano de Testes	Define o nível de cobertura segundo o qual os elementos mais críticos do software serão testados com prioridade e com um nível de cobertura mais amplo (BASTOS *et al.*,2007)
Relatório de Testes	Registra os defeitos encontrados durante a realização das atividades de testes (BASTOS *et al.*,2007)

Tabela 6. Conjunto de artefatos de Encerramento

Encerramento	
Artefato	**Objetivo**
Apresentação	Resume em uma apresentação aberta para s demais alunos os resultados obtidos com o desenvolvimento do projeto

4 Avaliação do Desempenho com a Metodologia Proposta

Para validar o uso e avaliar os resultados obtidos, a metodologia de ensino apresentada neste artigo vem sendo aplicada na disciplina de ES do Departamento de Computação da UFC desde 2005. A disciplina de ES é obrigatória e é ofertada anualmente para o Curso de Computação. Ela possui quatro créditos, totalizando sessenta horas-aula por semestre e as turmas têm em média quarenta alunos.

Nos quatro períodos letivos em que a metodologia de ensino foi aplicada, aproximadamente, quarenta projetos foram realizados, abrangendo um total de cento e sessenta alunos. No ano de 2005, somente alunos da graduação cursavam a disciplina de ES, no entanto, a partir de 2006, alunos dos cursos de mestrado e doutorado também passaram a cursar da disciplina.

A metodologia adotada no ano de 2005 contou com um único monitor para dar suporte a todas as equipes e com várias ferramentas de gerência de projeto, pois ficou a critério de cada equipe escolher uma ferramenta de gerência de projeto para acompanhar as suas atividades. Esta estratégia dificultou um acompanhamento criterioso das equipes por parte do monitor e do professor, aumentando, entretanto as notas atribuídas aos projetos, devido à falta de um maior rigor no controle das atividades desempenhadas pelas equipes. Com base no gráfico apresentado na Figura 1, observa-se que no ano de 2005 o desempenho nos projetos foi 35% superior ao desempenho nas provas.

Visando um acompanhamento mais rigoroso dos trabalhos das equipes por parte do monitor e do professor, no ano de 2006 a metodologia foi refinada. O diferencial foi a adoção de uma única ferramenta de gerência de projetos para todos as equipes denominada *dotProject* (DOTPROJECT, 2008). No entanto, as equipes continuavam livres para escolher o modelo de processo de desenvolvimento, as ferramentas de

desenvolvimento e tecnologias que julgassem mais adequadas para as características do projeto e dos membros da equipe.

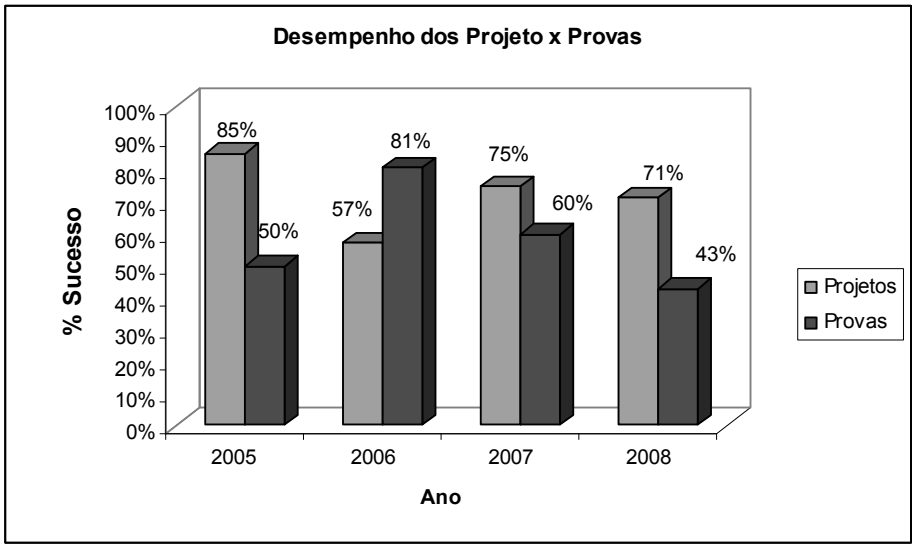


Figura 1. Análise do desempenho dos alunos nos projetos e provas

A ferramenta *dotProject* foi selecionada por tratar-se de um gerenciador de projetos web que pode ser acessada via Internet, propiciando o acompanhamento on-line das atividades, reduzindo o esforço necessário para acompanhar o andamento dos vários projetos. Nesse cenário, conforme pode ser verificado na Figura 1 houve uma inversão dos dados em relação ao ano anterior. O desempenho dos projetos foi 24% inferior ao das provas. Em compensação, os estudantes foram mais bem assessorados pelo monitor e professor extra-classe, o que resultou em um maior aprendizado.

No ano de 2007 a estratégia foi mantida. Analisando os dados apresentados no gráfico da Figura 1, pode-se perceber que o desempenho geral dos alunos, envolvendo provas e projetos, ficou mais equilibrado, tendendo ao nivelamento entre o desempenho individual nas provas e o desempenho coletivo no projeto. Entretanto, neste período percebeu-se que o uso de um único monitor para acompanhar todas as equipes não era suficiente para atender às necessidades específicas de aprendizado de todos os alunos, portanto, um atendimento extra-classe mais direcionado tornava-se necessário.

Com o objetivo de suprir não só as necessidades de execução dos projetos, como também refletir uma melhoria no desempenho dos alunos, em 2008 a metodologia foi refinada novamente. Nesse cenário, juntamente ao monitor, foi acrescentada uma equipe de assistentes de ensino composta por três alunos de pós-graduação em Estágio a Docência.

A cada integrante da nova equipe da monitoria foram alocadas apenas três equipes, possibilitando um suporte mais próximo aos alunos durante as atividades práticas. Com isso, um maior rigor na condução e controle de execução dos projetos foi adicionado. Em uma análise do gráfico mostrado na Figura 1, pode-se verificar que o desempenho nos projetos se manteve satisfatório, porém, o desempenho nas provas diminuiu. A causa identificada para a diminuição do rendimento dos alunos nas provas foi o aumento do grau de dificuldade das mesmas, necessário devido a qualidade do suporte extra-classe dado aos mesmos por parte da monitoria.

5 Conclusões e Trabalhos Futuros

De acordo com os resultados obtidos após a implantação da metodologia de ensino proposta podemos concluir que os alunos da disciplina de Engenharia de Software do Departamento de Computação da Universidade Federal do Ceará que utilizaram a metodologia de ensino demonstraram, ao final da disciplina, um preparo e conhecimento maior das práticas de Engenharia de Software.

Com um enfoque maior na gerência dos projetos e utilizando os conceitos na prática, a metodologia permite um aprendizado mais consistente do uso correto das técnicas teóricas e da importância de praticar um modelo de processo de software com todas as etapas incluídas no mesmo durante o desenvolvimento das aplicações.

O sentimento positivo demonstrado pelos alunos, ao final dos projetos, foi fator decisivo para que a metodologia continue sendo utilizada nos semestres seguintes de modo que possa ser continuamente melhorada e refinada. Além disso, como os resultados apresentados neste trabalho não foram comparados com as turmas anteriores a 2005, em que a metodologia ainda não era utilizada, pretende-se fazer essa avaliação como forma de validar a metodologia apresentada neste trabalho.

6 Agradecimentos

Agradecemos a colaboração dos alunos de graduação Bruno Sabóia Aragão e Luciana Paiva na elaboração dos modelos dos artefatos utilizados na disciplina. Agradecemos ainda a colaboração da aluna de Doutorado Maria de Fátima Costa Souza pelo auxílio na análise pedagógica dos dados obtidos com a metodologia e também à professora Dra. Bernadette Farias Lóscio, pelas informações dos resultados obtidos com o uso da metodologia no ano de 2007.

Referências

AGUIAR, Y. P. C., Lima, A. H. G., Leite, F. L. J., et. al. **easYProcess: Um Processo de Desenvolvimento para Uso no Ambiente Acadêmico**. In: XII Workshop de Educação em Informática - XXIV Congresso da Sociedade Brasileira de Computação, Salvador. 2004.

ALVES, A. G.,BENITTI, F.B.V. **Processo de Desenvolvimento Integrando Disciplinas de Engenharia de Software** In: XIV Workshop sobre Educação em Computação - XXVI Congresso da Sociedade Brasileira de Computação, Campo Grande. 2006.

BASTOS, A., RIOS, E., CRISTALLI, R., MOREIRA, T. **Base de conhecimento em teste de software**. 2ª. Edição, São Paulo. Editora Martins Fontes, 2007.

DEMO, P. A nova **LDB: Ranços e avanços**. Campinas: Papirus, 1997. 9ª ed.

DOTPROJECT. **The home of dotProject – the Open Source Project Management tool**. Disponível em: <http://www.dotproject.net>. Acesso em: 05 ago. 2008.

FIORENTINI, L. M. R. **Materiais didáticos escritos nos processos formativos a distância**. In: CONGRESSO DE ENSINO SUPERIOR A DISTÂNCIA, I, 2002. Petrópolis. Anais. Petrópolis: EsuD, 2002.

GAMMA, E., HELM, R., JOHNSON, R., and VLISSIDES, J. **Design Patterns, Elements of Object Oriented Software**, Addison Wesley, Reading, MA, 1994.

PIAGET, J. **Psicologia e Pedagogia**. Rio de Janeiro: Forense-Universitária, 1975. 3ª edição.

PINHEIRO, B. M. e GONÇALVES, M. H. **O Processo Ensino-Aprendizagem**. Rio de Janeiro: Editora SENAC Nacional, 2001.

SOMMERVILLE, I., **Engenharia de Software**, 8ª ed.SP: Pearson, 2007.