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[1] 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];

[1] 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[2] para fazer parte da modelagem e a IDE (*Integrated Development Environment*) Netbeans[3] 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 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.

---

[2] http://jude.change-vision.com/jude-web/index.html.
[3] http://www.netbeans.org/.

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 1 apresenta relação dos artefatos de software a serem gerados nos trabalhos.

Tabela 1 – 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 1 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 2 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 3 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).

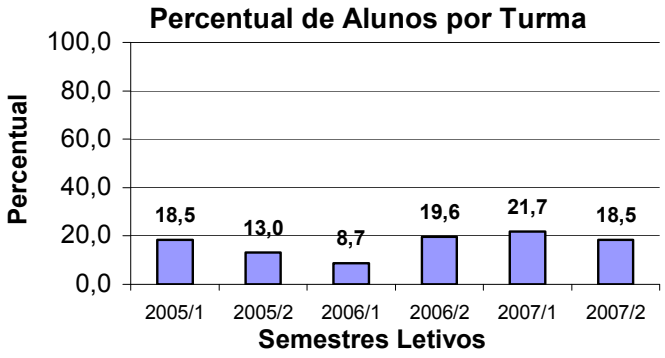

Figura 1 – Percentual de Alunos por Turma

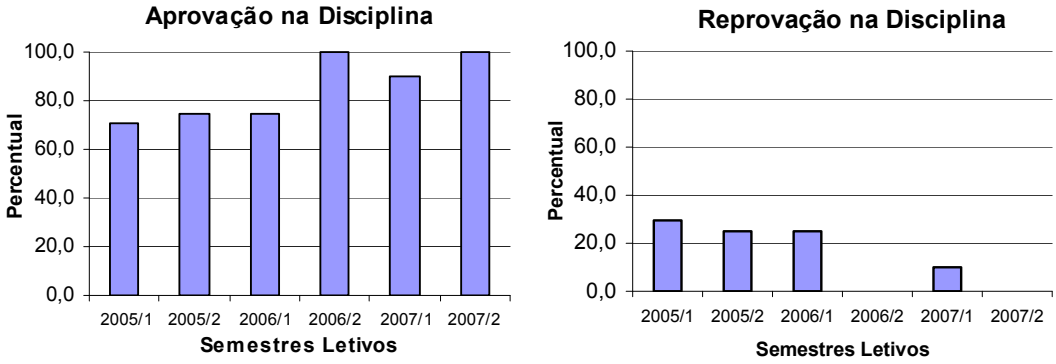

Figura 2 – Percentual de Aprovação e Reprovação dos Alunos por Semestre

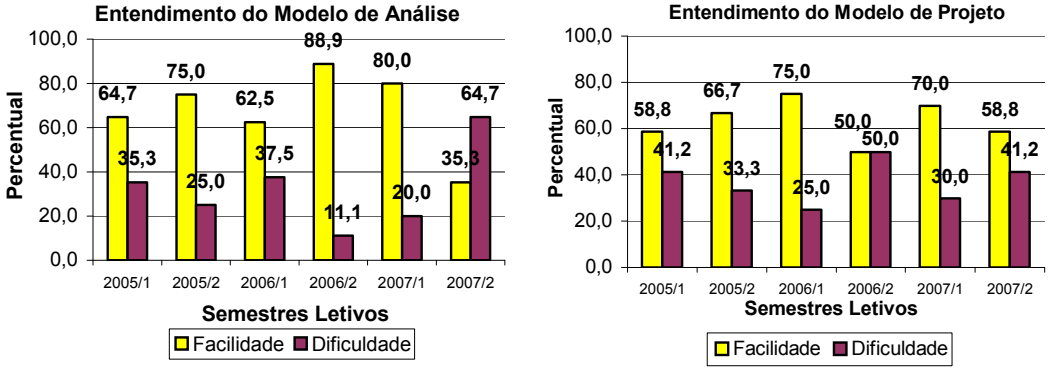

Figura 3 – Entendimento do Modelo de Análise e do Modelo de Projeto

A Figura 4 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 5 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%.

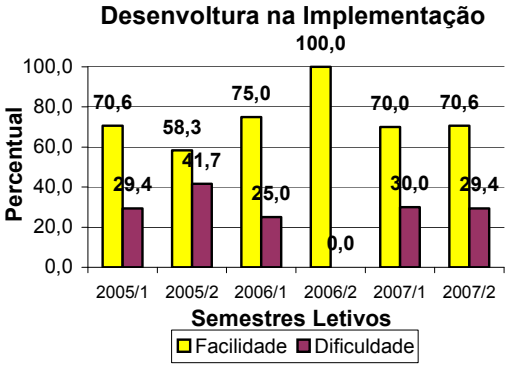

Figura 4 – Desenvoltura para Realizar a Implementação do Software

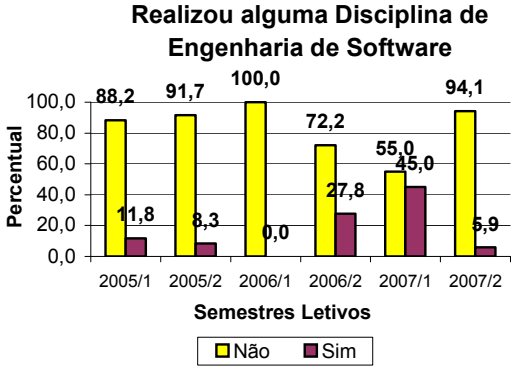

Figura 5 – Alunos com Conhecimento de Engenharia de Software

A Figura 6 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.

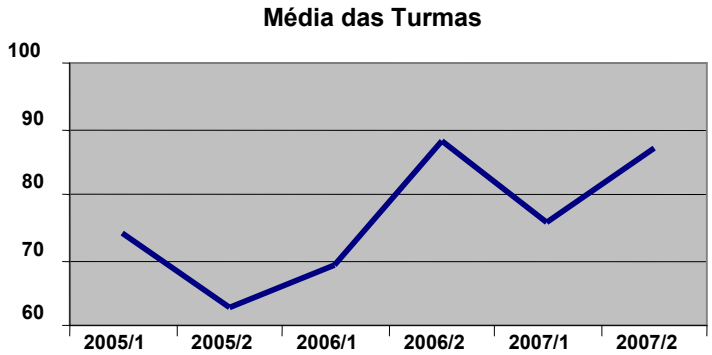

Figura 6 – Média Final das Turmas

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.