EXPERIMENTOS PARA O ENSINO DE ARQUITETURA DE SOFTWARE BASEADO NO METAMODELO EML

Renato Manzan de Andrade – renato.manzan@poli.usp.br
Escola Politécnica da Universidade de São Paulo – Departamento de Engenharia de Computação e Sistemas Digitais
Av. Prof. Luciano Gualberto, 158 Tr. 3 – Cidade Universitária
01065-970 - São Paulo - SP
Reginaldo Arakaki – reginaldo.arakaki@poli.usp.br

Resumo: *A Arquitetura de Software tem sido reconhecida como um artefato fundamental no processo de desenvolvimento de software. Nos últimos anos, o interesse em tópicos arquiteturais cresceu vertiginosamente, tanto na área de pesquisa, de educação e na prática de Engenharia de Software. Apesar de seu papel crucial para o sucesso de um projeto, a maioria dos currículos dos cursos de graduação em engenharia da computação não aborda arquiteturas de software formalmente. Este artigo apresenta uma abordagem prática para o ensino de arquitetura de software baseada no metamodelo da EML.*

Palavras-chave: Arquitetura de software, Metamodelos educacionais, Engenharia de Software

1. INTRODUÇÃO

Aplicações de software modernas envolvem adaptabilidade, escalabilidade, interoperabilidade, escalabilidade, distribuição e portabilidade. Conforme o tamanho e a complexidade das aplicações crescem, o problema do design ultrapassa o nível de complexidade dos algoritmos e das estruturas de dados da computação: o projeto e a especificação da estrutura e da dinâmica do sistema como um todo emerge como um novo tipo de problema que envolve toda a organização e o controle do sistema, protocolos de comunicação, sincronização, acesso a dados, distribuição das funcionalidades entre os diversos componentes e suas interfaces, relacionamentos, localização física, reuso de componentes e políticas de segurança.

A escolha de estilos arquiteturais e a implementação entre diversas alternativas que satisfaçam os requisitos funcionais e não funcionais constituem o nível arquitetural de design (SHAW; GARLAN, 1996).

Embora a arquitetura de software seja largamente reconhecida como um dos mais importantes artefatos produzidos durante o processo de desenvolvimento de software, em muitos casos o conceito de arquitetura ensinado nas academias não são efetivamente mapeados em aplicações reais de arquitetura de software (SHAW, 2000).

Estudantes possuem uma sólida base em programação, porém raramente conhecem conceitos arquiteturais e suas influências na qualidade, produtividade, custo e manutenção do software (ROSCA et al., 2003).

A maioria dos currículos existentes aborda fortemente algoritmos e estruturas de dados, tópicos extremamente importantes, mas que sem uma abordagem arquitetural, não

possibilitam o nível de abstração adequado para a construção de sistemas complexos com qualidade.

Durante a graduação os estudantes são expostos a poucos exemplos arquiteturais, geralmente em disciplinas como Sistemas Operacionais e Compiladores, onde a arquitetura é apresentada de forma extremamente teórica, o que prejudica o aprendizado (SHAW, 2000) e impossibilita a percepção da importância de uma arquitetura, pois geralmente os graduandos não conseguem "enxergar" uma abstração relativamente elevada a partir de um ponto de vista puramente teórico. Esta abordagem faz com que os alunos se tornem centrados apenas em código-fonte.

Com um aprendizado centrado em código, os estudantes tendem a criar um modelo mental muito simplista da natureza do software, no qual o maior nível de abstração aprendido resume somente a código fonte onde um problema é resolvido com um programa.

Um modelo mental é a representação da organização do conhecimento na mente humana. Mesmo sem uma orientação de como um modelo mental deva ser criado, o aluno cria um "automaticamente" para lidar com os conhecimentos aprendidos (GRUNDY et al., 1998). Um estudante ordinário está acostumado a codificar pequenos programas sem uma orientação arquitetural, o que contribui para a construção de um modelo mental simplista (BUCCI et al., 1998).

Estudantes com este modelo mental tendem a iniciar a codificação imediatamente após receber a especificação do problema a ser resolvido resultando em uma séria lacuna na formação acadêmica: há uma expectativa que recém-formados saibam construir sistemas complexos, porém os mesmos não dispõem dos pré-requisitos conceituais para fazê-lo de forma efetiva (BLAKE, 2003).

Este problema também ocorre nas empresas, onde muitos profissionais, mesmo após anos de experiência, mantêm um modelo mental simplista, o que dificulta a aceitação e o aprendizado de visões mais elaboradas, envolvendo componentes e conceitos arquiteturais.

Para evitar a formação de um modelo mental demasiado simplista na mente dos alunos, é necessário aumentar o nível de abstração do ensino a um patamar no qual a arquitetura de software possua um papel fundamental no desenvolvimento de sistemas computacionais.

O desenvolvimento de uma visão de mais alto nível de software significa esclarecer que um sistema não é um conjunto de arquivos descritos em uma linguagem de programação, mas um conjunto de elementos a partir dos quais o sistema é construído, considerando as interações entre eles, os padrões que orientam a composição e possíveis restrições.

O foco do ensino deve ser alterado do paradigma segundo o qual um programa resolve um problema para uma visão mais sofisticada que interpreta uma aplicação como uma coleção de componentes que determinam uma arquitetura. Desta forma, os estudantes poderão aprender conceitos como componentes distribuídos, organização estrutural e dinâmica de um sistema e requisitos não funcionais.

Um dos desafios à educação em Engenharia de Software é preparar os estudantes com uma forte base de Arquitetura de Software em seus currículos, capacitando-os para desenvolver e distribuir aplicações cada vez mais complexas de uma forma eficiente. (SHAW, 2000). Por esta razão, os conceitos de arquitetura de software estão se tornando cada vez mais importantes, por sua enorme influência na qualidade do software produzido e no processo de desenvolvimento.

Estudantes também devem aprender novas plataformas de desenvolvimento e suas respectivas ferramentas. Currículos deveriam ser revisados para incorporar conceitos teóricos associados com tópicos arquiteturais e os cursos poderiam incluir abordagens práticas. É imperativo que os programas educativos de engenharia de software tenham uma atitude mais proativa e considerem que a exposição dos estudantes a projetos focados em arquitetura pode potencialmente aumentar a qualidade da formação dos alunos (MURRAY, 2004).

2. O METAMODELO EML

A OUNL-EML (*Open Universiteit Nederland - Educational Modeling Language*) (KOPER, 2001) é uma iniciativa da Universidade Aberta da Holanda para modelagem de processos de aprendizagem e usa a UML para a representação dos modelos. Esta iniciativa, ainda em desenvolvimento, foi iniciada em 1998 e concorre pela padronização de um formato europeu aberto para interoperabilidade entre tecnologias de aprendizado (*Learning Technology Interoperability Standards*). Esta padronização tem sido feita por várias organizações e empresas, entre as quais se pode destacar o *Comité Européen de Normalization/Information Society Standardisation System - Learning Technologies Workshop* (CEN/ISSS LT-WS), *Institute for Electronic and Electrical Engineers Learning Technology Standards Committee* (IEEE LTSC) e *International Standards Organisation - Joint Technical Committee 1- Subcommittee 36: Standards for Learning Educational and Training* (ISO/JTC1/SC36).

O meta-modelo pedagógico que sustenta a EML permite representar integralmente o processo de ensino, isto é, não apenas seu conteúdo, mas também os objetivos, regras, ambientes de aprendizagem, interações e atividades dos estudantes e professores, permitindo o uso de várias abordagens pedagógicas.

Este meta-modelo baseia-se em um processo de ensino no qual o elemento principal é representado pela unidade de estudo, que pode ser definida como a menor unidade de ensino que satisfaz um ou mais objetivos de aprendizagem inter-relacionados, ou seja, elementos educacionais atômicos cuja aplicação visa atingir os objetivos educacionais estabelecidos (KOPER, 2001).

Como decorrência da definição, uma unidade de ensino, vista que é minimal, não pode ser particionada sem que haja uma perda semântica no processo de ensino/aprendizagem, porém engloba um conteúdo educacional que possibilita a existência de um processo de aprendizagem realizado por professores e alunos, permitindo que uma unidade de ensino seja reutilizada em vários cursos.

No contexto deste artigo, a unidade de estudo engloba conceitos de arquitetura de software como modelos, visões, ferramentas e mecanismos que possibilitam a realização de um processo de desenvolvimento de software centrado em arquitetura.

O meta-modelo pedagógico é constituído de quatro pacotes principais, representados na figura abaixo:

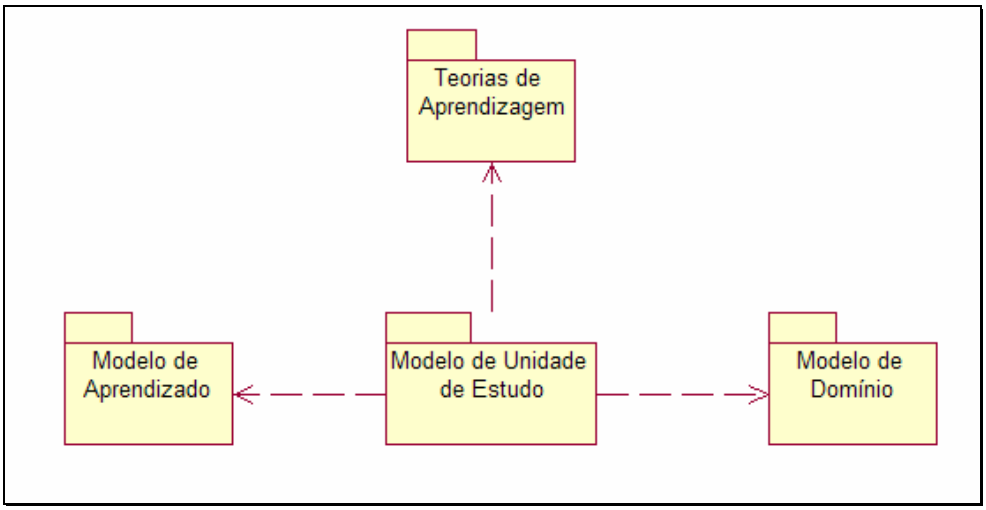

Figura 1 – Pacotes do meta-modelo pedagógico da EML (Adaptada de KOPER, 2001)

Segue uma descrição dos pacotes:

• **Modelo de aprendizado:** descreve o processo de aprendizado dos estudantes baseando-se nas teorias de aprendizagem;

• **Modelo de domínio:** representa o conteúdo e a organização dos tópicos a serem lecionados, como por exemplo, arquitetura de software. Importante esclarecer que cada domínio do conhecimento possui suas próprias estruturas de conhecimento, habilidades e competências, havendo diversas culturas e estilos de prática, o que exige que os diversos modelos pedagógicos sejam adaptados a determinados contextos;

• **Modelo de unidade de estudo:** descreve como as unidades de estudo aplicáveis em determinado contexto são modeladas, considerando-se um modelo de aprendizado e o modelo de domínio;

• **Teorias de aprendizagem:** repositório de teorias pedagógicas, princípios e modelos de aprendizagem. Na instanciação deste meta-modelo é necessária a especialização para uma determinada teoria.

O metamodelo da EML está representado na figura abaixo:

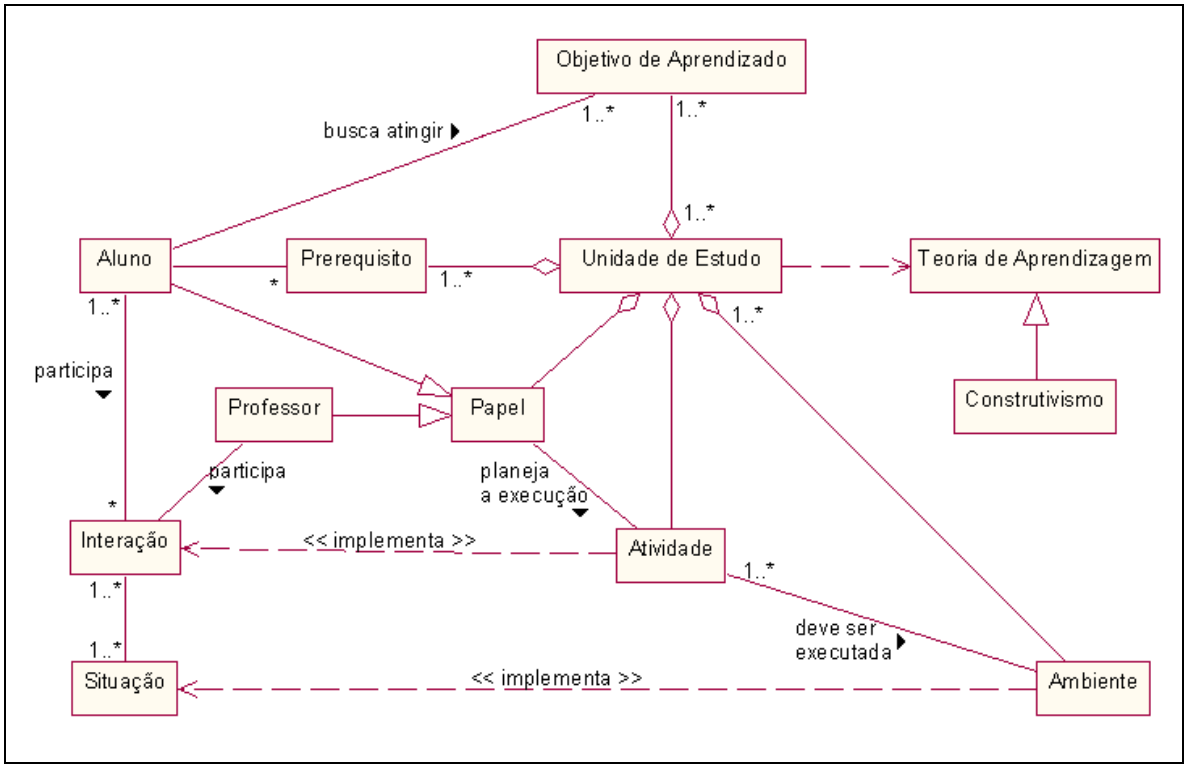

Figura 2 – Meta-modelo pedagógico definido pela EML (Adaptada de KOPER, 2001)

A partir da meta-modelo da EML e utilizando o Construtivismo como referência pedagógica, foram criados experimentos para o ensino de arquitetura de software, descritos no próximo item.

3. EXPERIMENTAÇÕES PARA O ENSINO DE ARQUITETURA DE SOFTWARE

De acordo com o Construtivismo, a aprendizagem é definida como um processo de reestruturação de conceitos prévios com a assimilação de conceitos novos, permitindo a

elevação do nível de abstração do conhecimento através da reorganização dos modelos mentais, que são gerados através da interação contínua entre o indivíduo e o meio (POZO, 1998); (VICHIDO et al., 2003).

A teoria construtivista estimula a criação de simulações de ambientes para que o estudante possa resolver "problemas reais" e criar seu próprio modelo mental que acomode o novo conhecimento adquirido. Neste ambiente, o professor deixa de atuar como transmissor do conhecimento e atua como orientador e facilitador do aprendizado, preocupando-se em prover ambientes e ferramentas que ajudem os alunos a interpretar as múltiplas perspectivas de análise do problema, auxiliando na consolidação do conhecimento (JONASSEN, 1998).

O uso de experimentações no processo de ensino/aprendizagem é incentivado pela teoria construtivista, uma vez que permite que o aluno eleve o nível de abstração de seu modelo mental objetivando a assimilação de novos conhecimentos, os quais devem ser ministrados em um nível crescente de abstração, o que permite a consolidação do aprendizado. Neste processo de constante reformulação dos modelos mentais, o professor deve motivar os alunos a analisar, compreender, interpretar e aplicar o novo conhecimento em uma simulação de um problema real (DICK, 1991).

Segundo CRONJÉ (1997), o conhecimento é construído pela execução de atividades baseadas em problemas do mundo real. Tais atividades, ministradas em um nível crescente de complexidade, possibilitam a realização de experiências por parte dos alunos que auxiliam no aprendizado.

O conhecimento deve ser construído na mente do aluno, que participa ativamente do processo de aprendizagem, o que exige uma atitude pró-ativa por parte do estudante, o qual deve participar mais ativamente do processo de ensino/aprendizagem, exercitando a criatividade na resolução de problemas. Este ambiente, no qual o aluno participa ativamente em sua aprendizagem, cria uma motivação natural para aprender, o que certamente refletirá em sua vida profissional (MULDER et al., 1997).

3.1. Kits de estudo

Visando auxiliar o ensino de arquitetura de software, os autores criaram kits de estudo, elementos de aprendizagem práticos aderentes ao metamodelo da EML, incluindo praticamente todos os elementos-chave para a construção da solução de um problema envolvendo conceitos arquiteturais.

Os kits são utilizados como ferramentas práticas de apoio ao ensino de arquitetura de software, possibilitando a construção de ambientes de ensino mais ricos, ou seja, com alto nível de aproveitamento por parte dos alunos e aderentes à teoria construtivista, colocando o aluno no controle do processo de aprendizagem.

Unindo o aprendizado centrado no aluno com uma forte iniciativa prática imersa em um contexto que parte do concreto para o abstrato, pode-se explorar a habilidade inata do ser humano de explorar para descobrir (OHLSSON; JOHANSSON, 1995).

Conforme VOCKELL e SCHWARTZ (1992), o conhecimento é na maioria das vezes organizado de forma hierárquica e por esta razão uma das formas de aumentar a eficácia do aprendizado é pela identificação dos pré-requisitos, ou seja, os conhecimentos necessários para o aprendizado de uma determinada abstração arquitetural, primando pela assimilação dos conceitos principais da arquitetura em questão.

Baseando-se nas considerações de VOCKELL e SCHWARTZ e considerando a riqueza, a complexidade e as múltiplas dimensões dos conceitos de arquitetura, os kits de estudo foram desenvolvidos utilizam abordagem iterativa e incremental, na qual os tópicos são apresentados em um nível de abstração crescente.

Esta abordagem permite que os alunos assimilem os conceitos de um determinado estilo arquitetural, reconhecendo suas vantagens e desvantagens em determinados contextos, antes de aprenderem um estilo ainda mais abstrato.

Para o ensino prático de arquitetura de software, o kit de estudo é composto pela descrição de um estilo arquitetural, considerando aspectos teóricos, motivações para o uso do estilo em questão, cenário de utilização em um determinado contexto, diagramas UML e um exemplo completo de implementação utilizando a plataforma J2EE (*Java 2 Enterprise Edition*).

Os kits de estudo representam as unidades de estudo do meta-modelo da EML permitindo um alto grau de customização do processo de ensino, e portanto a utilização do em vários cenários.

Além da flexibilidade, os kits possibilitam o uso de provas de conceito (*proof of concept*), uma técnica que permite demonstrar que uma determinada idéia é tecnicamente possível. No contexto do trabalho, provas de conceito são utilizadas para demonstrar a viabilidade de construção de um sistema de software utilizando um determinado estilo arquitetural, o que aumenta as chances de que a solução adotada satisfaça o nível de serviço esperado pelo usuário, potencialmente reduzindo o risco de fracasso. Provas de conceito exercitam a arquitetura cobrindo muitos aspectos importantes no desenvolvimento de uma aplicação, como por exemplo: testes de componentes, funcionalidade e performance.

Até o momento da escrita deste artigo, os seguintes kits de estudo já foram desenvolvidos na plataforma J2EE:

• **Kit de estudo – Instalação e configuração da plataforma de desenvolvimento e produção:** kit de auxílio para a instalação e configuração da plataforma tecnológica utilizada.

• **Kit de estudo – Arquitetura 2 camadas:** arquitetura simples com apenas 2 camadas (browser acessando base de dados diretamente). Traz a introdução do conceito de arquitetura de software e provê discussões em laboratório sobre vantagens e desvantagens de determinadas arquiteturas. Tópicos como coesão e acoplamento entre componentes de software também são considerados.

• **Kit de estudo – Acesso aos dados:** considerações sobre componentes de acesso às bases de dados. Mostra o uso de componentes configuráveis e reutilizáveis para conexão com os mais diversos tipos de bancos de dados. Neste kit é discutido o conceito de *stored procedures*, que permite desacoplar a lógica de negócio da lógica de dados.

• **Kit de estudo – Arquitetura 3 camadas:** evolução do kit de arquitetura 2 camadas, apresentando conceitos de componentização, como por exemplo: componentes de IHC (Interface Humano-Computador), controle e negócio. Salienta-se a importância de arquitetura baseada em componentes e do uso da engenharia simultânea no processo de desenvolvimento de software.

• **Kit de estudo – Modelos de Componentes:** introdução ao modelo de objetos distribuídos na plataforma J2EE. Uso de componentes EJB e servidores de aplicação que demonstram algumas funcionalidades gerenciadas pelo container J2EE como, por exemplo: ciclo de vida dos componentes, serviço de nomes e registro, gerenciamento transacional, segurança e controle de acesso e persistência de objetos.

• **Kit de estudo – Arquitetura n camadas:** arquitetura composta pelas camadas de IHC, controle, lógica de negócio, componentes de infra-estrutura e de acesso aos dados e

stored procedures. Aspectos de integração e implantação (*deployment*) do sistema em produção também são considerados neste kit.

O conjunto dos kits desenvolvidos abrange os seguintes aspectos de arquitetura de software: utilização prática de estilos arquiteturais em projetos de desenvolvimento de software, desenvolvimento baseado em componentes, uso de técnicas de engenharia simultânea e de provas de conceito, desenvolvimento centrado em arquitetura, utilização de ferramentas integradas para a construção de sistemas distribuídos, uso de servidores web, de aplicação e de base de dados.

4. ESTUDO DE CASO

A abordagem experimental para o ensino de arquitetura de software foi utilizada em uma disciplina de laboratório do curso de Engenharia da Computação da Escola Politécnica da USP.

Nesta disciplina, todos os alunos, estudantes de Engenharia Elétrica com ênfase em Computação, agrupados em equipes de 2 a 3 participantes, deviam desenvolver um sistema com o uso das técnicas de orientação a objetos.

O projeto desenvolvido pelas equipes era um sistema web que permitia que empresas cadastradas oferecessem empregos a profissionais ou estudantes, que por sua vez podiam cadastrar seus currículos no sistema. O sistema permitia:

- Inclusão de currículos pelo interessado;
- Busca inteligente de um perfil profissional;
- Solicitação de contatos por empresas interessadas;
- Monitoração *on-line* pelos administradores;
- Gerenciamento das informações sobre a formação dos engenheiros, estagiários, pesquisadores e profissionais pelos administradores;
- Registros de logs para rastreio de acesso.

Durante o decorrer da disciplina, os kits de estudo foram ministrados em ordem crescente de abstração e complexidade, o que permitiu a assimilação dos principais conceitos de cada estilo arquitetural. A ordem de aplicação dos kits é representada na tabela abaixo:

Tabela 1 – Ordem de aplicação dos kits de estudo

Ordem de Aplicação	**Nome do Kit de estudo**
1	Instalação e configuração da plataforma de desenvolvimento e produção
2	Arquitetura 2 camadas
3	Acesso aos dados
4	Arquitetura 3 camadas
5	Modelos de Componentes
6	Arquitetura n camadas

O Kit de estudo – Arquitetura n camadas (vide item 3.1) serviu como base para a implementação do projeto feito pelos alunos. A figura seguinte ilustra a divisão das camadas deste estilo arquitetural, implementada com a tecnologia J2EE. As setas indicam a comunicação entre camadas ou entre componentes:

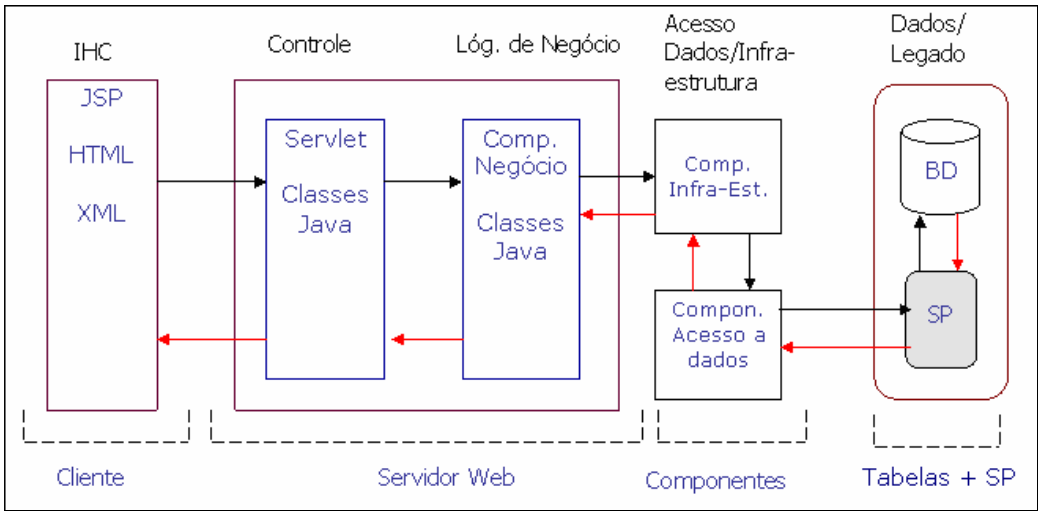

Figura 3 – Arquitetura utilizada no projeto

O fato de a turma inteira trabalhar no mesmo projeto tornou a disciplina muito próxima de um projeto real, no qual o professor desempenhava o papel de cliente. Esta aplicação permitiu explorar conceitos como gerência de projeto, trabalho colaborativo, gerência dos artefatos produzidos e engenharia simultânea. Aspectos humanos também tiveram que ser considerados, pois naturalmente surgiram conflitos entre as equipes afetando a motivação dos alunos e o desenvolvimento do projeto.

6. CONSIDERAÇÕES FINAIS

O aprendizado de conceitos arquiteturais baseado em experimentações pode auxiliar os estudantes na aquisição de importantes conhecimentos essenciais ao engenheiro de software como trabalho em equipe, processos de desenvolvimento de software, habilidades de comunicação e gerência de projeto, em um ambiente realista de desenvolvimento centrado em arquitetura de software, o que permite que os alunos adaptem e utilizem em suas vidas profissionais o que foi ensinado na Academia, contribuindo para suprir a alta demanda da sociedade por engenheiros de software altamente qualificados.

Este artigo representa uma parcela de uma pesquisa que vem sendo realizada pelos autores no desenvolvimento de *frameworks* para o ensino de arquitetura de software.

REFERÊNCIAS BIBLIOGRÁFICAS

BLAKE, M. B., Student-Enacted Simulation Approach to Software Engineering Education, **IEEE Transactions on Education**, Vol. 46, No. 1, p. 124-133, 2003.

BUCCI, P.; LONG T.; WEIDE B., Teaching software architecture principles in CS1/CS2, **Foundations of Software Engineering - Proceedings of the third international workshop on Software architecture**, p. 9-12, 1998.

CEN/ISSS LT-WS WEB SITE, **Comité Européen de Normalization/Information Society Standardisation System - Learning Technologies Workshop**. Disponível em <http://www.cenorm.be/isss>. Acesso em: 15 de maio de 2005.

CRONJÉ, J., Education for technology, technology for education, **Guidelines for technology-enhanced education at the University of Pretoria**, Volume I, ITI Working Paper, Institute for Technological Innovation, 1997.

DICK, W., An instructional designer's view of constructivism. **Educational Technology**, Volume 31, p. 41-44, 1991.

GRUNDY, J.; HOSKING, J.; MUGRIDGE, B., Inconsistency Management for Multiple-View Software Development Environments, **IEEE Transactions on Software Engineering**, VOL. 24, No. 11, p. 960-981, 1998.

IEEE LTSC WEB SITE, **Institute for Electronic and Electrical Engineers, Learning Technology Standards Committee**. Disponível em <http://ltsc.ieee.org>. Acesso em: 10 de maio de 2005.

ISO/JTC1/SC36 WEB SITE, International Standards Organisation - **Joint Technical Committee 1- Subcommittee 36: Standards for Learning Educational and Training**. Disponível em <http://jtc1sc36.org>. Acesso em: 24 de abril de 2005.

JONASSEN, D. **Designing constructivist learning environments**. Instructional theories and models. 2. ed. Mahwah, 1998.

KOPER, R., **Modeling Units of Study from a Pedagogical Perspective: the pedagogical meta-model behind EML**. Open Universiteit Nederland, 2001. Disponível em <http://eml.ou.nl/introduction/docs/ped-metamodel.pdf>. Acesso em: 25 de maio de 2005.

MULDER, M.; LIDTKE D.; STOKES E., Enterprise enhanced education: an information technology enabled extension of traditional learning environments, **Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education**, p.355-359, 1997.

MURRAY M., Move to component based architectures: introducing Microsoft's .NET platform into the college classroom, **The Journal of Computing in Small Colleges - ACM: The Consortium for Computing in Small Colleges**, Volume 19 , Issue 3, p. 301 – 310, 2004.

OHLSSON, L.; JOHANSSON, C., A practice driven approach to software engineering education, **IEEE Transactions on Education**, Volume: 38, p. 291-295, 1995.

POZO, J., **Teorias cognitivas da aprendizagem**. 3. ed. Porto Alegre: Artes Médicas, 1998.

ROSCA, D.; TEPFENHART, W.; MCDONALD, J., Software Engineering Education: Following a Moving Target, **Proceedings of the 16th Conference on Software Engineering Education and Training**, p. 129 – 139, 2003.

SHAW, M., Software engineering education: a roadmap, **International Conference on Software Engineering - Proceedings of the conference on the future of software engineering**, p. 371-380, 2000.

SHAW, M.; GARLAN D., **Software Architecture: Perspectives on an Emerging Discipline**, Prentice-Hall, Inc., Upper Saddle River, 1996.

VOCKELL, L.; SCHWARTZ, M., **The computer in classroom**. 2 ed.., McGraw-Hill, 1992.

VICHIDO, C.; ESTRADA, M.; SANCHEZ, A., A constructivist educational tool: Software architecture for web-based video games. **Proceedings of IEEE on the Fourth Mexican International Conference on Computer Science**, p. 144-150, 2003.

EXPERIMENTS FOR SOFTWARE ARQUITECTURE TEACHING BASED ON THE EML METAMODEL

Abstract: *Software architecture has been recognized as an important artifact in the software development process. In the last few years, the interest in software architecture as an area of research, education, and practice within software engineering has been growing steadily. In despite of its pivotal role in the software development process, most existing undergraduate software engineering curricula do not include software architecture disciplines formally. This paper presents a practical approach to teach software architecture based on the EML metamodel.*

Key-words: *Software Architecture, Educational metamodels, Software Engineering*