

Ensino de Engenharia de Software: Desafios, Estratégias de Ensino e Lições Aprendidas

Rafael Prikladnicki¹, Adriano Bessa Albuquerque²,
Christiane G. von Wangenheim³, Reinaldo Cabral⁴

¹Faculdade de Informática (FACIN) – PUCRS – Porto Alegre – RS

²Universidade de Fortaleza - UNIFOR – Fortaleza – CE

³PPGCC/INE/CTC – UFSC – Florianópolis – SC*

⁴PESC/COPPE – UFRJ – Rio de Janeiro – RJ

rafaelp@pucrs.br, adrianoba@unifor.br, gresse@gmail.com,
cabral@cos.ufrj.br

Abstract. *This paper presents a set of teaching experiences in Software Engineering from four Universities in Brazil. These experiences were planned and executed based on the identification of important challenges faced during the learning processes in the last years. Lessons learned and plans for future activities are also presented.*

Resumo. *Neste artigo apresenta-se um conjunto de experiências de ensino em Engenharia de Software vivenciadas por quatro Instituições de Ensino Superior no Brasil. Estas experiências foram planejadas e executadas a partir da identificação de desafios importantes durante o processo de ensino nos últimos anos. Junto com as experiências são apresentadas algumas lições aprendidas e passos futuros.*

1. Introdução

A formação qualificada e a capacitação de profissionais são cada vez mais necessárias na sociedade em que vivemos. Seja em cursos de curta duração, graduação ou pós-graduação, formar bons profissionais faz parte do compromisso das Instituições de Ensino Superior (IES) com a sociedade (Enricone, 2002). Especificamente no ensino de Engenharia de Software (ES), a qualidade dos profissionais está diretamente relacionada à qualidade da educação, embora também existam outros fatores que contribuem para isto (Beckman et al., 1997). A qualidade da educação em ES pode contribuir significativamente à melhoria do estado da arte do desenvolvimento de software e auxiliar a solução de alguns problemas tradicionais e crises relacionadas com as práticas da indústria de software (Gibbs, 1994). Hoje, a educação e o treinamento para formar profissionais de software, devem incluir não apenas conhecimentos básicos na área de computação, mas também o ensino de conceitos, processos e técnicas para definição, desenvolvimento e manutenção de software (Saiedian, 1999; ACM/IEEE, 2008).

* Anteriormente afiliada à UNIVALI

Mas ao mesmo tempo em que esta importância em relação ao conteúdo é reconhecida, também se demonstra cada vez mais significativo tratar aspectos didáticos e pedagógicos no ensino (Saiedian, 1999). Aulas tradicionais, estratégia de ensino predominante, raramente satisfazem a necessidade de aprendizagem (Grillo, 2002).

Segundo Junior (2008) e Grillo (2002), os métodos de ensino podem estar focados no professor ou focados no aluno. Cada um tem suas vantagens e desvantagens, em função do conteúdo da aula, das características dos alunos, da turma, entre outros fatores. Os métodos focados no aluno podem gerar uma maior motivação por parte de quem aprende, além de uma participação mais ativa no processo e uma melhor aprendizagem no nível de aplicação dos conceitos (Junior, 2008), o que hoje parece ser um dos principais pontos fracos no ensino de ES. E é exatamente este o tema deste artigo. A partir de experiências reais de quatro diferentes Instituições, apresentam-se estratégias de ensino de ES usando uma abordagem participativa com foco no aluno. A próxima seção apresenta os conceitos relacionados com o processo de ensino, enquanto que na seção 3 apresentam-se as estratégias de ensino. Lições aprendidas e recomendações são apresentadas na seção 4.

2. O Processo de Ensino

O professor é um dos principais responsáveis pelo ensino, pesquisa e gerenciamento do processo educacional. É ele quem escolhe os métodos utilizados para captar a atenção do seu público em um determinado período. O professor deve utilizar instrumentos didáticos que priorizem a participação do aluno, gerenciando suas expectativas e habilidades. Existem duas categorias principais de métodos de ensino: as focadas no professor e as focadas no aluno. A tabela 1 apresenta uma comparação entre os métodos (Junior, 2008; Grillo, 2002).

Tabela 1. Comparação entre as duas categorias de métodos de ensino

	Focado no professor	Focado no aluno
Papel do professor	<ul style="list-style-type: none"> •Principal fornecedor da informação •Especialista •Avaliador do rendimento 	<ul style="list-style-type: none"> •Facilitador •Fornece informação para ajudar na compreensão da informação
Clima de aprendizagem	<ul style="list-style-type: none"> •Individualista 	<ul style="list-style-type: none"> •Coletiva •Foco na coesão de grupo
Orientação	<ul style="list-style-type: none"> •Baseada na experiência e nos conhecimentos do professor 	<ul style="list-style-type: none"> •Baseada na experiência e conhecimento dos alunos
Programa de estudos	<ul style="list-style-type: none"> •Definido pelo professor 	<ul style="list-style-type: none"> •Negociado entre professor e alunos
Objetivo de ensino	<ul style="list-style-type: none"> •Definido pelo professor •Resultado padrão 	<ul style="list-style-type: none"> •Definido pelos alunos •Resultados diferentes para cada aluno
Aquisição de conhecimento	<ul style="list-style-type: none"> •Enfoque na aquisição •Foco na memorização 	<ul style="list-style-type: none"> •Enfoque na utilização e absorção de conhecimento com foco em problemas reais
Métodos de ensino	<ul style="list-style-type: none"> •Didático •Grande participação do professor 	<ul style="list-style-type: none"> •Métodos que envolvem a participação dos alunos (técnicas dinâmicas)
Foco na educação	<ul style="list-style-type: none"> •Educação individual 	<ul style="list-style-type: none"> •Educação coletiva
Avaliação	<ul style="list-style-type: none"> •Executada pelo professor •Uso tradicional de provas e notas 	<ul style="list-style-type: none"> •Os alunos também são responsáveis pela avaliação

Quanto mais expositiva for a aula, mais ela será centrada no professor. Por outro lado, quanto mais prática e dinâmica for a aula, mais ela será centrada no aluno. Apesar de a aula expositiva ser o principal e mais antigo instrumento de ensino no Brasil em

todos os níveis, a pedagogia centrada no professor tende a valorizar relações hierárquicas (Junior, 2008). Uma aula expositiva acaba sendo pouco eficiente, pois ativa apenas o sentido da audição. Eventos simulados e atividades vivenciais permitem assimilar situações diferentes na prática. Sendo assim, a aprendizagem mais prática acaba sendo caracterizada pelo envolvimento de uma pessoa em uma determinada atividade, com resultados mais positivos, visto que os alunos acabam sendo responsáveis pela definição de rumos de uma situação proposta.

2.1. Ensino de Engenharia de Software

A ES é uma disciplina preocupada com a aplicação de teoria, conhecimento e prática para o desenvolvimento efetivo e eficiente de sistemas de software que satisfaçam os requisitos dos usuários (ACM/IEEE, 2008). Geralmente, ES é ensinada no ensino formal no nível de graduação e/ou pós-graduação ou por meio de treinamentos profissionais de curta duração.

Em relação à estratégia de ensino, o currículo na área de ES (ACM/IEEE, 2004; ACM/IEEE, 2008) ressalta a necessidade de mover-se para além do formato de aula expositiva. Neste sentido, é importante considerar a variação de técnicas de ensino e aprendizado. As abordagens mais comuns para ensinar ES incluem aulas expositivas, aulas de laboratório, entre outros. Entretanto, abordagens alternativas podem ajudar os alunos a aprender de maneira mais efetiva, como por exemplo: a substituição de aulas expositivas por discussão de casos práticos (Gnatz et al, 2003), dinâmicas de grupo, o uso de jogos (Wangenheim e Shull, 2009) e *Capstone projects* (um esforço em grupo em que alunos executam um projeto do início ao fim) (Goold e Horan, 2002). Neste sentido, são apresentadas a seguir algumas experiências vivenciadas pelos autores com foco no aumento da participação do aluno no processo de ensino-aprendizagem em ES.

3. Estratégias participativas para o ensino de Engenharia de Software

3.1. O uso de dinâmicas de grupo, educação à distância e atividades práticas

Na PUCRS, no curso de graduação em Sistemas de Informação (SI), a área de Engenharia de Software é abordada principalmente em seis disciplinas (PUCRS, 2009). Neste artigo o foco é no relato da experiência com a disciplina de Gerência de Projetos de Software, que tem fomentado o aumento da participação do aluno desde 2004. Esta disciplina, de 60 horas-aula, envolve o ensino de conceitos básicos de gerência de projetos de software, relacionando o ciclo de gerenciamento com as etapas do ciclo de desenvolvimento.

Objetivo de aprendizagem: apresentar os conceitos básicos de gerência de projetos de software, mesclando teoria e prática.

Estratégia de ensino: A estratégia de ensino se propõe a tratar dos conceitos de gerência de projetos de forma diferenciada. A interação com os alunos é enfatizada através de dinâmicas que exploram assuntos específicos. São características chave da estratégia: (i) diversificação nas técnicas de dinâmicas de grupo: uso de dinâmicas envolvendo raciocínio, mini-fábrica de aviões e jogo de memória. A dinâmica com aviões são utilizadas para introduzir conceitos de processo de gerência de projetos. O jogo de memória é utilizado para trabalhar os conceitos do PMBOK (PMI, 2008) e raciocínio é utilizado em dinâmicas sobre gerência de comunicação; (ii) aula prática em laboratório: apesar de gerência de projetos ter um conteúdo bastante teórico, a disciplina

é organizada de forma que os alunos explorem conceitos de forma prática, envolvendo a definição de escopo de projeto, criação de EAP (estrutura analítica de projeto) e desenvolvimento de cronogramas; (iii) planejamento dos trabalhos pelos alunos: as datas de entregas (parcial e final) dos trabalhos são planejadas pelos alunos, respeitando datas limites definidas pelo professor. Isto faz com que os alunos tenham uma experiência real de planejamento e monitoramento de um projeto, que no caso é o trabalho da própria disciplina. A entrega no prazo vale nota, e qualquer alteração deve ser justificada; e (iv) aulas semipresenciais: 20% da disciplina (em média seis aulas) são realizadas utilizando-se de recursos EAD (educação à distância). Nestas aulas, os alunos vivenciam situações reais de projetos e precisam resolver estando fora do ambiente de aula. Uma contribuição relevante da aula EAD é o desenvolvimento de habilidades de trabalho em equipe e comunicação à distância.

Avaliações: A avaliação da disciplina é realizada através de provas individuais e a entrega de dois trabalhos, todos com pesos iguais. Parte das atividades desenvolvidas nas aulas em formato EAD compõe a nota do trabalho 1, e parte compõe a nota de uma das provas. Além disso, existe uma atividade para avaliar a percepção de evolução do aprendizado, onde os alunos preenchem questionários no primeiro dia de aula e recebem o mesmo questionário no último dia de aula para avaliar a evolução do aprendizado e comparar com as respostas fornecidas no início do semestre;

Experiências: Esta disciplina começou a incentivar a participação dos alunos em 2004, e desde então vem evoluindo de formato. Os questionários foram introduzidos no ano de 2004. Em 2005 foram introduzidas duas dinâmicas (para ensino da importância de processo de software e para ensino do processo de gerenciamento das comunicações), complementadas por uma terceira dinâmica em 2008 (ensino de gerenciamento de projetos iterativos e incrementais). As aulas em laboratório foram introduzidas em 2006, e o planejamento do trabalho por parte dos alunos foi introduzido em 2008. As aulas EAD foram introduzidas em 2009. Apesar de haver o estímulo da avaliação, houve uma participação acima de 90% dos alunos nas atividades EAD. No futuro planeja-se a inclusão de jogos de computador.

3.2 O uso de *capstone projects* e atividades práticas

Na UNIVALI no curso de graduação em Ciência da Computação no campus São José a área de Engenharia de Software é principalmente abordada por quatro disciplinas. Dentro deste contexto, o objetivo das primeiras disciplinas é ensinar os conceitos básicos. No contexto deste artigo, o foco está na última das quatro disciplinas, chamada de Análise e Projeto de Sistemas 2. Esta disciplina representa um *capstone project* conforme sugerido nas recomendações da ACM/IEEE (2004), em que grupos de alunos planejam e executam um projeto de software do início até o fim durante um semestre inteiro.

Objetivos de aprendizagem: Os objetivos de aprendizagem são de reforçar a compreensão de conceitos, processo e técnicas de ES e de que o aluno tenha a competência para aplicá-los com assistência na prática. Além disso, visa à evolução de habilidades, como: comunicação, trabalho em equipe e resolução de problemas.

Estratégia de ensino: O enfoque principal desta disciplina é a internalização de conhecimento e habilidades pelo "*learning by doing*". Durante a disciplina é simulado um projeto de software, incluindo o planejamento, a execução (cobrindo as fases de análise de requisitos, *design*, implementação e testes) e a monitoração final. As equipes

de projetos são formadas por aproximadamente seis alunos. Cada aluno assume uma responsabilidade principal, porém, participa ativamente em todas as fases do projeto. O cliente é representado pelo professor, que também fornece uma descrição de alto-nível das necessidades do cliente.

O projeto se inicia com a fase de planejamento, onde é elaborado um plano de projeto. Em seguida, se iniciam as atividades técnicas conforme o plano de projeto e seguindo um modelo de processo de desenvolvimento pré-definido com base no modelo deciclo de vida cascata. Durante a execução do projeto, os alunos realizam as fases de análise de requisitos, *design*, implementação e testes. Em paralelo, os alunos coletam dados referentes aos prazos e esforço gastos nas atividades. Estes dados são avaliados no final do projeto representando um resumo de monitoração de projeto e são comparados com as estimativas no plano de projeto inicial. No final de cada fase, cada grupo entrega e apresenta os resultados parciais do projeto.

Avaliações: As avaliações da disciplina são realizadas por meio das entregas e apresentações dos resultados parciais do projeto no decorrer do semestre. No final do semestre é realizada uma avaliação individual por meio de uma prova.

Experiências: Esta disciplina está sendo oferecida neste formato (com pequenas variações) desde 2000. O principal ponto forte observado pelos alunos é o fato que a disciplina oferece a oportunidade de acompanhar a execução de um projeto de software do início ao fim com o objetivo de reunir todos os conhecimentos e habilidades aprendidos anteriormente durante o curso. Observa-se, que a necessidade de efetivamente aplicar os conceitos da ES também ajuda a reforçar o conhecimento teórico, além de começar a criar a competência de aplicá-los. Isto fornece aos alunos a oportunidade de alcançar competências e desenvolver habilidades, que não são facilmente adquiridas com métodos de ensino tradicionais.

Para possibilitar que a disciplina reflita cada vez mais a realidade dos projetos de desenvolvimento de software, planejou-se também a troca de projetos entre os grupos em cada fase (por exemplo, o Grupo A desenvolve os requisitos referente ao projeto X, faz o *design* referente ao projeto Y para o qual grupo B desenvolveu os requisitos). Porém, esta prática requer que todos os trabalhos sejam realizados com um nível de qualidade mínima e dentro dos prazos dos marcos, para não prejudicar nenhum grupo de alunos. Outro ponto crítico da organização da disciplina é a criação de grupos maiores, pois o grau da contribuição de cada aluno pode variar significativamente causando injustiças na avaliação, mas, por outro lado, permite que os alunos treinem também habilidades, como trabalho em grupo e comunicação.

3.3. O uso de atividades lúdicas e jogos

A disciplina de ES do curso de graduação em Ciências da Computação da Universidade de Fortaleza – UNIFOR é ofertada no penúltimo semestre do curso e tem como pré-requisitos as disciplinas de Análise e Projeto de Sistemas 1 e Análise e Projeto de Sistemas 2. A disciplina é ministrada com foco nos modelos de maturidade de software, principalmente no MR MPS (SOFTEX, 2007).

Objetivos de aprendizagem: Aumentar a compreensão de conceitos envolvidos na execução dos processos relacionados aos principais modelos de maturidade, bem como aumentar a percepção do aluno sobre como se dá a execução de cada um destes processos e sobre a importância de executá-los de forma integrada.

Estratégia de ensino: Primeiramente são apresentados conteúdos relacionados à qualidade de produtos de software para que posteriormente sejam apresentados conteúdos relacionados a processos de software, abrangendo a Normas ISO, processos ágeis e de forma geral o CMMI (SEI, 2006) e MR MPS (SOFTEX, 2007). Após uma ampla contextualização, os conteúdos relevantes de cada processo do MR MPS são apresentados e trabalhos em equipe. No final da disciplina, objetivando consolidar a compreensão dos processos Gerência de Projetos, Gerência de Requisitos, Garantia da Qualidade, Medição e Gerência de Configuração do MR MPS, é realizada uma atividade lúdica em grupo, utilizando LEGO.

Uma das motivações de se utilizar LEGO é a relevância de atividades lúdicas para a retenção de conteúdos. Além disso, este brinquedo possibilita projetar, a partir de requisitos definidos, um produto a ser construído, havendo um bom nível de similaridade com a construção de software. Outra motivação para este tipo de abordagem em sala de aula é o professor poder realizar mais intervenções junto aos alunos e observar mais de perto as dificuldades encontradas durante a execução dos processos, percebendo assim, que conteúdos foram menos compreendidos. O uso de LEGO é planejado para ser executado em três aulas, seguindo um roteiro pré-definido. Os alunos devem planejar e construir um projeto real, fazendo o monitoramento e controle do projeto, a partir dos processos que foram definidos.

Avaliações: As avaliações dos alunos na disciplina são realizadas por meio de provas individuais e trabalhos em equipe. A atividade lúdica premia com 1 ponto na média final os membros da melhor equipe. Além disso, os alunos também avaliam a abordagem utilizada respondendo um questionário contendo 19 questões.

Experiências: De acordo com a última avaliação da disciplina, a grande maioria dos alunos (95%) considerou que a abordagem despertou neles a compreensão da relevância de se ter bons mecanismos de comunicação para a eficiência de processos de software. Além disso, 96% dos alunos consideraram que passaram a conhecer melhor as relações existentes entre os processos. Este resultado foi importante, pois em apenas três aulas a interação entre os processos foi mais compreendida do que nas aulas teóricas, onde é sempre muito mais difícil perceber os pontos de interligação entre os processos.

A grande maioria dos alunos considerou como principal ponto fraco da dinâmica o tempo estabelecido para sua execução, além de alguns alunos não encararem a experiência com seriedade. Também foram destacados os seguintes pontos fortes: a possibilidade de aumentar o nível de interação entre os alunos e aluno/professor e o auxílio para consolidar conceitos, a partir da vivência da teoria. A principal oportunidade de melhoria identificada foi aumentar o tempo de execução da abordagem.

3.4. Atividades lúdicas no aprendizado por analogia

Em cursos de graduação ou pós-graduação, é possível desenvolver projetos que envolvem todo o ciclo de desenvolvimento de software, a exemplo da estratégia apresentada na seção 3.2. Porém, em cursos de curta duração (de 4 a 16 horas) não é possível realizar este exercício e este é um grande desafio: como potencializar a aprendizagem em cursos de curta duração? Esta estratégia, desenvolvida pelo Laboratório de Engenharia de Software (LENS) da COPPE/UFRJ, trata esta questão.

Objetivos de aprendizagem: Prover o conhecimento e as habilidades necessárias à definição, à avaliação e à melhoria de processos de software que permeiam o ciclo de vida dos produtos de software.

Estratégia de ensino: A estratégia está baseada no uso de um instrumento didático que propicia o entendimento dos objetivos, da importância e da dinâmica dos processos de software pelo uso de analogias. O instrumento didático é constituído por um processo de aplicação, instrumentação (quebra-cabeças, um processo definido conforme o propósito, formulários de coleta de dados, *check-lists* e outros) e diretrizes de adaptação (permitem a configuração do instrumento para diferentes contextos).

Para aplicar o instrumento em um cenário de melhoria de processos de software, por exemplo, os alunos são divididos em grupos e assumem os papéis de garantia da qualidade, medição, gerente de projetos e desenvolvedores. Um processo definido (de montagem do quebra-cabeça) é entregue com o quebra-cabeça desmontado. Um objetivo de melhoria é definido para todos os grupos: “diminuir o tempo de montagem”. A partir daí são executados três ciclos de montagem com quebra-cabeças diferentes: o primeiro para que o processo seja aprendido; o segundo para que o processo seja mensurado e o terceiro para testar as melhorias feitas no processo. Cada aluno exerce o seu papel, mensurando o processo, avaliando a qualidade dos produtos ao longo da execução, gerenciando o projeto e desenvolvendo o produto (montando o quebra-cabeça) e todos atuam na melhoria de forma conjunta. Após a aplicação do instrumento todos os aspectos exercitados são discutidos e tratados com enfoque teórico.

Avaliações: A avaliação do alcance dos objetivos de aprendizagem é realizada pelo uso de critérios qualitativos que incluem a participação e envolvimento durante a utilização do instrumento didático e durante a discussão a *posteriori*, que considera os aspectos teóricos apresentados versus as situações vivenciadas durante a dinâmica. Todos os participantes são incitados a emitir sua opinião sobre como a teoria poderia ser utilizada para potencializar os resultados da dinâmica. Neste ponto, é comum observar o surgimento de discussões polêmicas que podem ser exploradas de diversas formas.

Experiências: O instrumento foi submetido a diferentes cenários de cursos de curta duração em que o LENS atuou: em cursos fechados para empresa pública, em cursos abertos para empresas privadas, em disciplinas de pós-graduação e em diferentes regiões do País (sudeste, norte e nordeste), variando de quatro a oito horas de duração.

O instrumento didático foi desenvolvido em outubro de 2005 e desde então vem evoluindo e sofrendo adaptações que permitem a exploração de diversos cenários. A abordagem descrita na seção 3.3, por exemplo, é um fruto da evolução desta estratégia definida para cursos de curta duração aplicada ao contexto de cursos de graduação.

4. Lições Aprendidas

A partir das experiências relatadas neste artigo foram identificadas importantes recomendações, traduzidas em lições aprendidas: **Lição 1:** O uso de *capstone projects* propicia aos alunos vivência em projetos de software ao mesmo tempo que permite mesclar teoria e prática em disciplinas onde a experiência desempenha um papel fundamental; **Lição 2:** jogos e dinâmicas auxiliam na retenção de um conjunto de conceitos e significados que são mais facilmente compreendidos pelos participantes; **Lição 3:** As atividades lúdicas bem planejadas proporcionam maior abertura para lidar com o novo, o inesperado e as dificuldades comuns em projetos de software; **Lição 4:** O uso de abordagens que incentivam os alunos a testar hipóteses, refletirem sob suas práticas e se posicionarem de forma crítica fomentam o aprendizado e a participação.

5. Considerações finais

Neste artigo foram apresentadas experiências de ensino de ES em cursos de graduação, pós-graduação e curta-duração de quatro IESs brasileiras. O foco do artigo foi na descrição de diversas experiências de ensino evitando as abordagens tradicionais. Devido ao limite de páginas, não foi possível aprofundar-se em cada estratégia de ensino com o nível de detalhe desejado. A partir desta limitação, uma das idéias futuras é conceber um local comum e integrado, de acesso público, para compartilhar diferentes estratégias de ensino para a comunidade de educadores de ES. Recomenda-se utilizar, sempre que possível, estratégias de ensino em que o aluno possa vivenciar os conteúdos apresentados em sala de aula, tirando-os da condição de meros observadores para manipuladores de instrumentos da realidade.

Agradecimentos

Este trabalho recebeu apoio do Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq.

Referências Bibliográficas

- ACM/IEEE. Computer Science Curriculum, 2008.
- ACM/IEEE. Software Engineering Curriculum. Guidelines for Undergraduate Degree Programs in Software Engineering, 2004.
- Beckman, K., Coulter, N., Khajenouri, S., Mead, N., Collaborations: Closing the industry-academia gap. *IEEE Software* **14** (6), pp. 49-57, 1997.
- Enricone, D. (Org.) *Ser Professor*, 3a edição, EDIPUCRS, 2002.
- Gibbs, W., Software's chronic crisis. *Scientific American* **271** 3 (1994), pp. 86-95.
- Gnatz, M., Kof. L., Prilmeier, F., Seifert, T. *A Practical Approach of Teaching Software Engineering*, Proc. 16th Conf. Software Eng. Education and Training, pp. 120-128, 2003.
- Goold, A. e Horan, P. *Foundation software engineering practices for capstone projects and beyond*. Proc. 15th Conference on Software Engineering Education and Training (CSEE&T 2002), IEEE CS Press, pp 140-146, 2002.
- Grillo, M. C., Práticas docentes e referenciais norteadores. *Caderno Marista de Educação*, Porto Alegre, v. 2, n. 2, p. 41-52, 2003.
- Junior, W. H., Sauaia, A. C. A. Aprendizagem Centrada no Participante ou no Professor? Um Estudo Comparativo em Administração de Materiais. *RAC*, **12** (3), pp. 631-658, 2008.
- PMI – Project Management Institute. *A guide to the project management body of knowledge*, 4th edition, 2008.
- PUCRS, Bacharelado em Sistemas de Informação, Grade curricular, disponível online em www.pucrs.br/inf/estruturacurricular/sistemasdeinformacao_4621.htm, acesso em Julho/09.
- SEI, CMMI® for Development, Version 1.2. CMU/SEI-2006-TR-008 ESC-TR-2006-008. Pittsburgh, PA Software Engineering Institute-SEI, Carnegie Mellon University: 561, 2006.
- SOFTEX, MR-MPS - Modelo de Referência para Melhoria de Processos de Software - Guia Geral v1.2, Campinas, SP, SOFTEX: 53, 2007.
- Saiedian, H., Software engineering education and training for the next millennium, *Journal of Systems and Software*, v. 49, i. 2-3, p. 113-115, 1999.
- Wangenheim, C. G. v., Shull, F. To Game or Not to Game?, *IEEE Software*, **26** (2), pp. 92-94, 2009.