Utilizando Resolução de Problemas para aproximar Teoria e Prática na Engenharia de Software

Cleo Z. Billa[1], **Márcia C. Cera**[1]

[1]Universidade Federal do Pampa (UNIPAMPA) - Campus Alegrete
Av. Tiarajú, 810 - Bairro Ibirapuitã – 97.546-550 – Alegrete – RS – Brazil

{cleobilla,marciacera}@unipampa.edu.br

Resumo. *Este artigo objetiva descrever o curso de Engenharia de Software da UNIPAMPA, onde almeja-se uma maior proximidade entre teoria e prática. Para tal, o curso foi concebido com base na abordagem de ensino-aprendizagem ABP (Aprendizagem Baseada em Problemas), a qual é implementada em componentes curriculares chamados de Resolução de Problemas. Neles, equipes de alunos são desafiados a proporem um sistema computacional que solucione um problema real. Na concepção da solução são praticados os conceitos de ES e a organização do trabalho em equipe. O foco deste relato são as disciplinas de RP-I e RP-III, ofertadas aos primeiro e terceiro semestres respectivamente.*

Abstract. *This paper describes the Software Engineering Undergraduate Program at UNIPAMPA, where our major goal is to approximate theory and practice. Aiming this goal, the program proposes to use the Problem Based Learning (PBL) methodology. This methodology is implemented on Problem Solving (PS) courses, where students are challenged to develop a computational system to real problem. In order to provide such solution, it is required software engineering concepts and team work. This paper reports our experiences in SP-I and SP-III.*

1. Introdução

No Brasil, a demanda pela formação de novos desenvolvedores de software é uma realidade, havendo hoje no setor de tecnologia um déficit de 115 mil trabalhadores conforme a [Associação Brasileira de Empresas de Tecnologia da Informação e Comunicação 2012]. Porém, o desenvolvimento e a manutenção de software demandam profissionais cada vez mais qualificados, capazes de compreender esses processos e de atuar explicitamente em sua definição e melhoria, com vistas a produzir software para os mais diferentes domínios e propósitos. Exige, portanto, um perfil específico de profissional da área de Computação, o Engenheiro de Software.

Buscando por alternativas, os professores da área de computação do Campus Alegrete da UNIPAMPA, desenvolveram o projeto do curso de Engenharia de Software (ES). Nele, procura-se estimular a realização de práticas coletivas para complementar o ensino dos conceitos, métodos e tecnologias da área de engenharia de software [Billa 2012, Forno et al. 2012]. Com isto, espera-se proporcionar meios para estimular um aprendizado com forte embasamento teórico-prático, pautado na autoaprendizagem e na aproximação com as práticas profissionais.

Os cursos de Engenharia de Software são recentes no Brasil, embora sejam comuns no exterior, inclusive com currículos definidos pela IEEE Computer Society e Association for Computing Machinery (ACM). Como o foco das atividades profissionais desta área estão voltados ao projeto, desenvolvimento e avaliação de sistemas computacionais, é necessário estimular os alunos a vivenciar este tipo de prática, onde a teoria ensinada no curso deve ser aplicada. A alternativa proposta para tentar aproximar a teoria e a prática na ES da UNIPAMPA foi através do uso de uma estratégia pedagógico/didática centrada no aluno, denominada Aprendizagem Baseada em Problemas - ABP (*Problem Based Learning* - PBL) [Araújo and Sastre 2009].

2. APB e Engenharia de Software

Estudos recentes sugerem que uma aprendizagem eficaz pode depender da adoção de estratégias de ensino–aprendizagem e está diretamente ligada a estratégias cognitivas e orientações motivacionais. Esses estudos demonstram que existem estratégias facilitadoras da aprendizagem que são susceptíveis de serem ensinadas [Martins 2002]. E de acordo com o [The Join Task Force on Computing Curricula - IEEE and ACM 2004], além de se discutir *quais* os tópicos de engenharia de software devem ser ensinados, deve-se discutir também *como* esses tópicos devem ser ensinados. Segundo [Schots et al. 2009], é importante discutir as estratégias para minimizar as dificuldades de ensino-aprendizagem e as adaptações no ensino para atender as demandas da indústria de software. Em [Prikladnicki et al. 2009] há um estudo para avaliar as principais formas de ensino de engenharia de software. Segundo eles, as abordagens mais comuns incluem aulas expositivas, aulas de laboratório, entre outros.

Porém existem diversos trabalhos que sugerem a utilização de projetos para o ensino de engenharia de software, entre eles [de Oliveira Barros and de Araujo 2008, da Cunha et al. 2008, dos Santos et al. 2008, Santos and Saba 2010]. Neles, têm-se a APB como estratégia de ensino–aprendizagem, com a exploração de problemas de diversos tipos, permitindo o desenvolvimento do raciocínio lógico, da criatividade, o aumento da motivação e da interpretação de textos pelo aluno quando da resolução do problema [Martins 2002].

Uma das considerações mais interessantes ao se aplicar a metodologia ABP para o ensino de engenharia de software é a proximidade com as situações vivenciadas no mercado de trabalho. 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 e as habilidades relacionadas a seres humanos 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 [de Oliveira Barros and de Araujo 2008, da Cunha et al. 2008].

Santos et al. [dos Santos et al. 2008] também apresenta uma metodologia de ensino baseada em APB 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.

Adicionalmente, o projeto, independentemente de qual seja, acaba sempre tomando um contexto multidisciplinar, já que os problemas normalmente não apresentam a divisão acadêmica de matérias e disciplinas [Araújo and Sastre 2009]. A experiência do uso de projetos para desenvolver o caráter multidisciplinar da engenharia de software é relato por [da Cunha et al. 2008]. Assim como [Braga 2009] também sugere a utilização de projetos para se ter um contexto interdisciplinar entre a disciplina de engenharia de software e as outras disciplinas em cursos de Ciência da Computação.

3. Estrutura do Curso de Engenharia de Software

A proposta do curso de Engenharia de Software da UNIPAMPA, foi idealizada tomando como base os currículos de referência disponibilizados pela IEEE [IEEE Computer Society Professional Practices Committee 2004] e pela ACM [The Join Task Force on Computing Curricula - IEEE and ACM 2004]. Durante a fase de planejamento do curso, percebeu-se que era importante encontrar meios para aproximar as vivências acadêmicas do cotidiano dos estudantes, com o ambiente de trabalho que eles encontrarão na vida profissional. Para que fosse possível atingir tal objetivo, foram buscadas metodologias alternativas de ensino-aprendizagem, onde chegou-se a ABP.

Essa intencionalidade é refletida no currículo pela proposição de disciplinas denominadas Resolução de Problemas (I a VI), que objetivam abordar, de modo interdisciplinar e em grupos, a resolução de um problema dentro de um eixo temático: Construção de Software (1º e 2º semestres); Modelagem e Projeto de Software (3º semestre); Análise e Validação de Software (4º semestre); Processo, Evolução, Qualidade e Gerenciamento (5º e 6º semestre) [The Join Task Force on Computing Curricula - IEEE and ACM 2004, IEEE Computer Society Professional Practices Committee 2004] . Finalizando, nos 7º e 8º semestres, com o desenvolvimento do Trabalho de Conclusão de Curso e do estágio supervisionado.

Nas disciplinas de Resolução de Problemas, os grupos de trabalho nos quais os alunos são distribuídos ficam sob a orientação de professores tutores, responsáveis pelo acompanhamento da disciplina. Nos Planos de Ensino dessas disciplinas, elaborados em conjunto pelos professores tutores, devem estar explicitadas as estratégias a serem adotadas com a turma, incluindo avaliações individuais e em grupo.

4. Estudos de caso

Nesta seção apresentaremos dois estudos de casos: as disciplinas de RRP-I (Seção 4.1) e RP-III (Seção 4.2). Por uma questão de espaço, optamos por descrever apenas as disciplinas de Resolução de Problemas que foram ofertadas concomitantemente no primeiro semestre de 2011. Elas representam a abordagem de dois eixos temáticos distintos: a Construção de Software (1º semestre) e Modelagem e Projeto de Software (3º semestre).

4.1. Resolução de Problemas I

O primeiro contato dos alunos com a iniciativa de aproximação entre teoria e prática é na disciplina de RP-I no primeiro semestre do curso. Há um cuidado especial em sua estruturação para introduzir tanto práticas elementares da área de engenharia de software, quanto práticas de uma abordagem não convencional de ensino-aprendizagem como a ABP. Além de acompanhar os alunos no restante do curso, o trabalho em equipe será o ponto chave para o aprendizado efetivo, logo é importante cativá-los em RP-I.

Dentro do eixo temático de Construção de Software, o primeiro semestre do curso visa prover princípios de lógica e programação; introdução aos conceitos básicos de projeto de sistemas (introdução ao levantamento de requisitos, estudos de caso, etc.) e o uso de ambientes computacionais. Neste sentido, a solução dos problemas propostos em RP-I devem envolver estes aspectos. Adicionalmente, os problemas precisam ser complexos o suficiente para requerer o trabalho em equipe na sua solução [Santos and Angelo 2009]. Desta forma será possível estimular iterações entre os alunos em prol da construção do conhecimento coletivo.

Visando estimular os alunos, os problemas sugeridos para a turma de 2011 representavam demandas de sistemas computacionais da própria UNIPAMPA: geradores de senhas seguras; verificadores de senhas; histórico de senhas; sugestão de disciplinas a serem cursadas pelos alunos e gerenciador de reservas de salas. Estes 5 temas foram trabalhados por 10 equipes de 5 ou 6 membros. Haviam 5 tutores sendo que cada tutor observou 2 grupos sob um dos 5 temas.

Para guiar as atividades vinculadas a concepção das soluções, foram produzidos alguns artefatos tanto para ajudar na organização do trabalho em equipe quanto na aplicação de conceitos embasadores da ES. Estes artefatos foram produzidos por professores e alunos na turma de 2010, dentro do contexto de um projeto de ensino com fomento da instituição. Foi uma experiência interessante poder contar com o ponto de vista dos alunos que já haviam cursado a disciplina, pois eles permitiram a identificação dos pontos onde, geralmente, estão a maior parte das dificuldades no processo de concepção da solução. Eles também assumiram o papel de monitores de RP-I, oferecendo atendimento extraclasse e trabalhando como facilitadores no processo de ensino-aprendizagem.

Os artefatos podem ser classificados em 3 aspectos: *(i)* apoio aos alunos; *(ii)* organização da equipe de trabalho e *(iii)* introdução as práticas de ES. No primeiro grupo incluem-se um blog (*http://esunipampa.blogspot.com.br/*), mantido pelos alunos, que serve como um meio de rápido acesso a informações sobre RP. Adicionalmente, em virtude de um levantamento de dificuldades referente a manipulação de arquivos em Java também foi elaborado um tutorial sobre o tema.

O segundo grupo de artefatos visa estimular a divisão de tarefas entre os membros da equipe. Nele, consta um Modelo de Ata de Reunião com Tutor onde ficam registradas as tomadas de decisão em função dos problemas e encaminhamentos necessários para a concepção da solução. É importante perceber que as decisões são tomadas pelas equipes, sendo que o tutor figura apenas como um mediador. Outro artefato de organização das equipes é o Modelo de Relatório Semanal, o qual mantém um registro da delegação de responsabilidades entre os membros da equipe. Nele constam: *as atividades programadas para a semana*, *as atividades realizadas na semana*, *as dificuldades enfrentadas durante a realização das atividades* e *as atividades previstas para a próxima semana*. Este último torna-se o primeiro item do relatório da semana seguinte. Este modelo estimula práticas de desenvolvimento iterativo e incremental do trabalho como as propostas no SCRUM [Schwaber and Beedle 2004].

O terceiro grupo de artefatos visa estimular a aplicação da ES na concepção da solução. Por se tratar do primeiro semestre do curso, os alunos ainda não estão familiarizados com práticas elementares da ES, então buscou-se estruturar um Documento de

Requisitos simples que as contemplasse. Ele documenta a solução incluindo uma visão geral do sistema (principais funcionalidades, escopo do produto, benefícios, limitações, etc); seus requisitos funcionais, não funcionais associados e suplementares; casos de uso; o projeto externo com a interface de entrada e saída de dados com os usuários e o projeto interno com a implementação do sistema propriamente dita. Este modelo inclui uma explicação item a item do que é esperado em seu preenchimento. Para reduzir as dúvidas no preenchimento do Documento de Requisitos, também foi produzido e disponibilizado um exemplo contendo uma solução proposta por uma equipe da turma de 2010. Como os problemas diferem de ano para ano, a disponibilização do exemplo estimula a reflexão sobre o que deve ser elaborado para o problema que está sendo solucionado.

A avaliação das atividades de RP-I ocorreram em várias frentes. A solução proposta foi avaliada incrementalmente através de 3 seminários (chamados *checkpoints*) onde os grupos apresentavam: *Checkpoint I* - escopo, principais funcionalidades e requisitos do sistema; *Checkpoint II* - planejamento da interface de entrada e saída do sistema; *Checkpoint III* - a implementação e demonstração do sistema. O Documento de Requisitos foi preenchido de forma incremental e a cada *checkpoint* ele era avaliado pelos tutores, que destacavam deficiências e apontavam possíveis melhorias. A cada novo *checkpoint*, o documento deveria refletir o amadurecimento da proposta oriundo das reflexões estimuladas, tanto na apresentação quanto nas observações feitas pelo tutor, e o andamento das atividades remanescentes. Logo, a cada *checkpoint*, tanto a apresentação do grupo quanto o Documento de Requisitos foi avaliado, possibilitando um acompanhamento da evolução do trabalho da equipe. Adicionalmente, os alunos foram avaliados individualmente pelo tutor a fim de verificar o comprometimento com as atividades dentro da equipe. Eles também fizeram uma auto avaliação com relação a sua postura dentro da equipe, assim como avaliaram os colegas. Todos estas frentes de avaliação visam despertar a reflexão sobre o desenvolvimento coletivo das atividades e de seu próprio engajamento na equipe.

4.2. Resolução de Problemas III

Durante a disciplina de RP-III, em 2011, foi proposto aos alunos o desenvolvimento de um sistema para auxiliar no ensino do Processo de Enfermagem. Assim, em um ambiente de desenvolvimento de software, os professores da enfermagem seriam os clientes. O processo de enfermagem é um instrumento metodológico e sistemático de prestação de cuidados. Serve para organizar a atividade intelectual do enfermeiro e provê guia para um determinado estilo de julgamento [Atkinson and Murray 1989]. O processo de enfermagem pode ser dividido nas seguintes fases: *(i)* histórico de enfermagem – anamnese e exame físico; *(ii)* diagnóstico de enfermagem; *(iii)* plano de cuidados; *(IV)* implementação dos cuidados planejados; *(V)* avaliação dos resultados; e *(VI)* evolução do paciente no prontuário.

Em RP-III, os alunos foram separados em 6 grupos com 4 integrantes. Como a disciplina tinha dois professores, cada professor foi tutor de 3 grupos. A disciplina foi dividida em 5 etapas:

1) Apresentação do Problema: Nessa primeira etapa as professoras de enfermagem apresentaram o processo de enfermagem e explicaram no que se esperavam que o software auxiliasse. A principal preocupação dos professores não era em ter um software que automatizasse, por exemplo, o diagnóstico a partir do histórico, mas

simplesmente que o ele fizesse o registro do diagnóstico que deveria ser dado pelo enfermeiro usuário do sistema. Dessa forma, o sistema basicamente deveria registrar as informações coletadas e inferidas pelo enfermeiro. O mais importante era que o sistema permitisse ao enfermeiro seguir o fluxograma do processo de enfermagem, sem pular ou repetir uma das fases.

2) Levantamento de Requisitos: Duas semanas após a primeira etapa, os alunos tiveram outro encontro com os professores de enfermagem para a fase de levantamento de requisitos, uma das principais etapas do processo de desenvolvimento de software. Essa etapa foi realizada duas semanas depois do primeiro encontro para que os alunos pudessem conversar com seus tutores para entender, de forma geral, o que é o processo de enfermagem e também como o software deveria se comportar. Durante a reunião de levantamento de requisitos os alunos fizeram diversas perguntas aos professores de enfermagem. Essas perguntas foram desde um nível mais amplo, como o que era esperado do software, até questões mais específicas, como quais informações sobre o paciente deveriam ser recolhidas, ou quais valores válidos para um determinado campo.

3) *Checkpoint I*: Dois meses após a apresentação do problema, os alunos fizeram uma apresentação para mostrar o andamento dos seus projetos. Para esse *checkpoint*, os tutores estabeleceram que os grupos deveriam ter cumprido as seguintes metas:

- Levantamento de requisitos em UML (*Unified Modeling Language*);
- Diagrama de classes e seus relacionamentos, associações e multiplicidades;
- Interações entre objetos através do Diagrama de Sequências;
- Modelagem de Estados e Modelagem de Atividades;
- Modelo de Implementação e Modelo de Implantação;
- Modelagem conceitual da base de dados (ER);
- Uso de técnicas para desenvolvimento de IHC.

4) Testes e Validação: Apesar da disciplina contemplar o eixo de Modelagem e Projeto de Software, e a fase de testes ser uma fase mais adiantada do processo de desenvolvimento de software, foi identificada a necessidade de dedicar uma etapa do projeto a fase de testes e validação de software. Isto porque o software seria aplicado em um ambiente real depois de pronto. É importante notar que os próprios alunos sentiram falta dessa fase do projeto. A medida que o software estava sendo desenvolvido, eles perceberam que só o cliente poderia realmente identificar se o que estava sendo desenvolvido estava de acordo com a proposta inicial.

5) *Checkpoint II*: A última etapa da disciplina foi a apresentação final do projeto. Nela, os alunos apresentaram toda a modelagem e o projeto do software, além do software implementado e a apresentação de alguns testes. As seguintes metas foram exigidas no final do projeto:

- Sistema implementado e funcional;
- Programação Orientada a Objetos;
- Identificação e agregação de algum padrão no desenvolvimento do Sistema;
- Avaliação da Interface.

Além das etapas descritas acima, cada grupo junto com o seu respectivo tutor, distribuía tarefas e estipulava um cronograma para que a cada etapa, todas as metas pudessem ser alcançadas.

5. Discussões

A área de Engenharia de Software se mostrou bastante adequada à adoção de estratégias baseadas em resolução de problemas. O processo de resolução de problemas é especialmente útil quando se faz necessário desenvolver novos sistemas, justamente porque um novo sistema de informação é desenvolvido para solucionar um problema ou um conjunto de problemas enfrentados por uma organização.

Através de uma avaliação preliminar não formal foi possível perceber que houve uma melhora no desempenho dos alunos, principalmente em relação a iniciativa e ao trabalho em equipe. Alguns trabalhos não foram concluídos totalmente, entregando produtos com muitos requisitos incompletos, porém pode-se notar que os alunos trabalham mais motivados, procuram soluções para resolver os problemas encontrados durante a execução do projeto. E, especialmente, quando o cliente, foi alguém não ligado a área de computação, eles puderam perceber que a comunicação é fator determinante para se ter um produto adequado as necessidades do cliente.

Ainda em relação a postura dos alunos, eles costumam afirmar que o uso do metodologia APB permite perceber a diferença entre o conceito e a aplicação desse conceito numa situação real. Esse tipo de vivência agrega experiência, deixando-os mais bem preparados para o mercado real. Além de exercitar habilidades como responsabilidade e pró-atividade, o trabalho em equipe estimula a construção conhecimento uma vez que as discussões, durante a concepção da solução, fazem com que a equipe atinja um nível homogêneo de aprendizado.

No futuro, espera-se tirar proveito da estrutura do curso de ES que aproxima teoria e prática para estimular a integração Universidade e empresa, com vistas ao estabelecimento do parque tecnológico que está sendo implantado dentro da própria Universidade.

Referências

Araújo, U. F. and Sastre, G. (2009). *Aprendizagem baseada em Problemas no ensino superior*. Summus, 1 edition.

Associação Brasileira de Empresas de Tecnologia da Informação e Comunicação (2012). Setor de tecnologia da informação tem déficit de 115 mil trabalhadores.

Atkinson, L. D. and Murray, M. E. (1989). *Fundamentos de Enfermagem: Introdução ao Processo de Enfermagem*. Guanabara Koogan.

Billa, C. Z. (2012). O método APB aplicado ao curso de engenharia de software. In *Anais do Congreso Internacional PBL 2012*, Santiago de Cali, Colômbia.

Braga, J. C. (2009). Diretrizes para o ensino interdisciplinar de engenharia de software. In de Souza Gimenes, I. M. and Leite, J., editors, *Anais do FEES09 - Fórum de Educação em Engenharia de Software, , Fortaleza, Outubro 9, 2009*.

da Cunha, A. M., e Silva, G. B., de Almeida Monte-Mor, J., Domiciano, M. A. P., and Vieira, R. G. (2008). Estudo de caso abrangendo o ensino interdisciplinar de engenharia de software. In Werner, C. and Leite, J., editors, *Anais do FEES08 - Fórum de Educação em Engenharia de Software, Campinas-SP, Brasil, Outubro 17, 2008*.

de Oliveira Barros, M. and de Araujo, R. M. (2008). Ensinando construção de software aplicada a sistemas de informação do mundo real. In Werner, C. and Leite, J., editors,

Anais do FEES08 - Fórum de Educação em Engenharia de Software, Campinas-SP, Brasil, Outubro 17, 2008.

dos Santos, S. C., da Conceição Moraes Batista, M., Cavalcanti, A. P., Albuquerque, J. O., and Meira, S. (2008). Usando pbl na qualificação de profissionais em engenharia de software. In Werner, C. and Leite, J., editors, *Anais do FEES08 - Fórum de Educação em Engenharia de Software, Campinas-SP, Brasil, Outubro 17, 2008*.

Forno, M. H. D., Teixeira, H. R., Jaques, G. S., Cera, M. C., and Vieira, V. G. (2012). Aprendizagem baseada em problemas aplicado a engenharia de software: Suporte a disciplina de resolução de problemas I. In *Anais do Congreso Internacional PBL 2012*, Santiago de Cali, Colômbia.

IEEE Computer Society Professional Practices Committee (2004). Swebok - guide to the software engineering body of knowledge. available in http://www.computer.org/portal/web/swebok.

Martins, J. G. (2002). *Aprendizagem Baseada em Problemas Aplicada a Ambiente Virtual de Aprendizagem*. PhD thesis, Universidade Federal de Santa Catarina.

Prikladnicki, R., Albuquerque, A. B., von Wangenheim, C. G., and Cabral, R. (2009). Ensino de engenharia de software: Desafios, estratégias de ensino e lições aprendidas. In de Souza Gimenes, I. M. and Leite, J., editors, *Anais do FEES09 - Fórum de Educação em Engenharia de Software, , Fortaleza, Outubro 9, 2009*.

Santos, D. M. B. and Saba, H. (2010). Avaliação do componente curricular interdisciplinar de engenharia de software. In Barros, M., editor, *Anais do FEES10 - Fórum de Educação em Engenharia de Software, Salvador, Bahia, Setembro 27, 2010*.

Santos, J. A. M. and Angelo, M. F. (2009). Análise de problemas aplicados em um estudo integrado de programação utilizando pbl. In *Anais do Workshop sobre Educação e Informática*. SBC.

Schots, M., Santos, R., Mendonça, A., and Werner, C. (2009). Elaboração de um survey para a caracterização do cenário de educação em engenharia de software no brasil. In de Souza Gimenes, I. M. and Leite, J., editors, *Anais do FEES09 - Fórum de Educação em Engenharia de Software, , Fortaleza, Outubro 9, 2009*.

Schwaber, K. and Beedle, M. (2004). *Agile Software Development with Scrum*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

The Join Task Force on Computing Curricula - IEEE and ACM (2004). Software engineering 2004: Curriculum guidelines for ungraduated degree programs in software engineering, a volume of the curricula computing series. available in http://sites.computer.org/ccse/SE2004Volume.pdf.