Uma Experiência Integrada de Programação Orientada a Objetos, Estruturas de Dados e Projeto de Sistemas com PBL

Roberto Almeida Bittencourt, Carlos Alberto Rodrigues, Danila S. Santos Cruz

Universidade Estadual de Feira de Santana (UEFS)
Av. Transnordestina, s/n – 44.036-900 – Feira de Santana – BA – Brasil

{roberto,carod}@uefs.br, danilasilva@gmail.com

Abstract. *This paper describes a learning experience conducted over two academic terms that integrated the courses of object-oriented programming, data structures and system design, by using a problem- and project-based learning approach. Among the advantages of the approach, it is worth mentioning the integrated practice of different disciplines, in addition to the acquisition of high-level social and cognitive skills by students. We observed some problems with cognitive overload and difficulties with software evolution, which can be avoided through proper sequencing of problems. Observed benefits of more authentic learning routes and practices closer to professional environments are encouraging, although more experience needs to be acquired to better gauge the approach.*

Resumo. *Este artigo relata uma experiência de ensino-aprendizagem realizada durante dois semestres letivos que integrou as disciplinas de programação orientada a objetos, estruturas de dados e projeto de sistemas utilizando uma metodologia de aprendizagem baseada em problemas e projetos. Dentre as vantagens da abordagem, vale ressaltar a prática integrada de diferentes disciplinas, além da aquisição, pelos alunos, de competências sociais e cognitivas de nível elevado. Alguns problemas observados de sobrecarga cognitiva e dificuldades com a evolução de software podem ser evitados, através de uma sequência adequada dos problemas apresentados. Os benefícios observados de rotas de aprendizagem mais autênticas e mais próximas das práticas profissionais são animadores, ainda que mais experiências precisem ser realizadas para melhor calibrar a abordagem.*

1. Introdução

No início do século XXI, após a realização de diversos debates promovidos por diversas entidades o Conselho Nacional de Educação – CNE aprovou a resolução 11/2002 que instituiu as Diretrizes Curriculares Nacionais dos Cursos de Graduação em Engenharia [CNE 2002]. Essa resolução estabeleceu uma série de competências em seus artigos, dentre as quais podemos destacar a atuação crítica e criativa na identificação e resolução de problemas, capacidade para resolver problemas de engenharia, boa comunicação oral e escrita, dar mais ênfase ao trabalho individual e em grupo dos estudantes e estímulos para as atividades de pesquisa. Este novo engenheiro deve ser capaz de aprender a aprender, de inovar, de se comunicar em um contexto global, demonstrar criatividade,

trabalhar em grupo, dentre outras competências [Silveira 2005]. Em constraste com isso, ao invés de enfatizar tais competências, observamos que um dos problemas atuais do ensino de engenharia em disciplinas isoladas através de metodologias tradicionais de ensino centradas no professor é a excessiva preocupação com conteúdos. Apesar de o estudante ser capaz de demonstrar proficiência nas provas das disciplinas, os conteúdos são muitas vezes desvinculados da realidade, o que, em última instância, reduz a retenção do aprendizado [Svinicki e McKeachie 2010] [Woods 1994]. A utilização de metodologias de aprendizagem ativa como PBL, por sua vez, permite trabalhar com problemas mais complexos e contextualizados na realidade [Delisle 1997], desenvolvendo competências além da mera reprodução de conhecimentos em provas escritas.

Por outro lado, problemas complexos e embasados na realidade são naturalmente interdisciplinares. Neste sentido, a integração de várias disciplinas no currículo é necessária para permitir a introdução destes tipos de problemas. De modo geral, a integração curricular pode ser realizada através da associação de grupos de disciplinas com um eixo comum, compartilhando desafios que podem ser tratados utilizando os conhecimentos de um dado conjunto de disciplinas.

Com isso em mente, nosso curso de Engenharia de Computação vem realizando, desde sua criação há cerca de dez anos, um esforço de integração curricular e de práticas de aprendizagem ativa baseadas em problemas e projetos [Angelo e Bertoni 2012]. Diversos estudos integrados têm sido oferecidos como parte do currículo obrigatório do curso nas mais diversas áreas de formação do engenheiro de computação. Uma das primeiras experiências foi o estudo integrado de programação que inicialmente procurava integrar a aprendizagem de técnicas de programação orientada a objetos com estruturas de dados. Esta experiência foi repetida durante alguns anos e é descrita em detalhes em outro trabalho [Santos et al. 2008].

Recentemente, o currículo de nosso curso foi alterado, e a disciplina de projeto de sistemas foi incluída no estudo integrado de programação. Isto ocorreu pela necessidade de antecipar o estudo de aspectos de design que facilitem a prática de programação em um paradigma orientado a objetos [UEFS 2012]. Este trabalho relata os resultados dos dois primeiros períodos desta experiência de reformulação curricular. Em um estudo integrado semestral com carga horária de dez horas semanais, propusemos quatro desafios de aprendizagem na forma de problemas, os quais foram resolvidos pelos estudantes utilizando a metodologia PBL. Esta experiência permitiu a integração de conhecimentos através de experiências mais autênticas e a prática de produção de software mais disciplinada, além da aquisição de competências mais amplas de comunicação, trabalho em equipe e autodidatismo. Por outro lado, esta mudança trouxe algumas dificuldades na aprendizagem, em especial os acúmulos de deficiências em projetos de maior duração e a sobrecarga cognitiva ao tratar vários conceitos simultaneamente, as quais serão discutidas aqui.

Neste artigo, apresentamos primeiramente os fundamentos de nossa prática na Seção 2. Na Seção 3, apresentamos uma rápida introdução sobre como adaptamos a metodologia PBL no estudo integrado de Programação. Na Seção 4, relatamos nossa experiência em dois períodos letivos e discutimos os resultados. A Seção 5 sintetiza as

principais lições aprendidas. Finalmente, encerramos o artigo na Seção 6 com conclusões e direções para trabalhos futuros.

2. Fundamentos

Nossa experiência está ancorada em aprendizagem baseada em problemas e projetos, em um currículo integrado e em uma taxonomia de objetivos educacionais, cujos fundamentos são apresentados a seguir.

2.1. Aprendizagem baseada em problemas e projetos (PBL)

Aprendizagem baseada em problemas e projetos (PBL) é uma metodologia de aprendizagem ativa com origens na escola de Medicina da Universidade de McMaster [Barrows 1986]. No modelo proposto por Barrows, estudantes se reúnem em pequenos grupos tutoriais para resolver desafios surgidos da prática clínica, sendo geralmente guiados por um tutor neste processo. No processo, resgatam fatos conhecidos, formulam questões sobre o que ainda não sabem, geram hipóteses para solucionar o problema e propõem metas de aprendizagem. As metas dirigem a aprendizagem individual, que é posteriormente compartilhada com o grupo. Este processo é repetido ciclicamente até encontrar uma solução. PBL é uma das principais inovações educacionais do ensino superior no final do século XX [Svinicki e McKeachie 2010]. O modelo médico foi adaptado para a educação em engenharia [Woods 1994], ciências [Duch et al. 2001], computação [Uden e Beaumont 2006] e várias outras áreas. No Brasil, foi adotado inicialmente em cursos de medicina [Gomes et al. 2009]. Na computação, até onde conhecemos, as experiências pioneiras foram em nossa universidade [Bittencourt e Figueiredo 2003] [Santos et al. 2007a] e na Unisinos [Ferreira et al. 2003].

2.2. Integração curricular

Organizações curriculares de cursos de nível superior podem variar de acordo com as competências que se deseja formar. Organizações curriculares muito segmentadas, baseadas em disciplinas, nem sempre são eficazes para a formação das competências abrangentes demandadas para engenheiros do século XXI [Bittencourt e Figueiredo 2003]. Além disso, os problemas científicos e tecnológicos atuais são cada vez mais multidisciplinares ou interdisciplinares, o que requer integração ainda maior. Num currículo tradicional, alguns alunos conseguem fazer a ponte entre as várias disciplinas, mas a experiência sugere que nem sempre isto ocorre com a maioria dos egressos, o que geralmente traz dificuldades em sua transição para o mundo profissional.

Desde o início de nosso curso, em 2003, trabalhamos com uma organização curricular integrada, baseada em componentes curriculares chamados primeiramente de estudos temáticos e, atualmente, de estudos integrados [Bittencourt e Figueiredo 2003]. Um estudo integrado (EI) é um componente curricular de objetivo integrador girando ao redor de algum tema e organizado em módulos. Geralmente, cada estudo é composto por um ou mais módulos teóricos e um módulo integrador. Os módulos teóricos são recortes em determinados campos do conhecimento organizados de forma autocontida e coesa. Já o módulo integrador é o componente que permite realizar a integração dos conhecimentos através de problemas e projetos, o espaço central onde utilizamos a metodologia de grupos tutoriais de PBL.

2.3. Taxonomia de Bloom para objetivos educacionais

A taxonomia de objetivos educacionais de Bloom é uma classificação de objetivos educacionais em três domínios: cognitivo, afetivo e psicomotor [Bloom et al. 1976]. O domínio cognitivo, que abrange a aprendizagem intelectual, é dividido em seis níveis: os três mais baixos, de conhecimento, compreensão e aplicação, e os três mais elevados, de análise, síntese e avaliação. Bons problemas ou projetos em PBL devem desafiar os estudantes a desenvolver competências nos níveis mais elevados da escala do domínio intelectual desta taxonomia [Duch et al. 2001]. Por exemplo, devem ir além de relembrar conceitos, de explicar o significado de algo ou de aplicar um conceito na solução de um problema. Devem propor o desenvolvimento de habilidades de pensamento de nível superior, tais como dividir um problema em suas partes, produzir um artefato a partir de elementos componentes, ou fazer julgamentos de soluções alternativas a partir de critérios estabelecidos.

3. PBL e o Estudo Integrado de Programação

Em nosso curso de Engenharia de Computação, as disciplinas costumam ser oferecidas através de um estudo integrado (EI) que agrega disciplinas diferentes com um objetivo comum. Com relação a programação, há dois EIs no curso: No primeiro semestre, um estudo de Algoritmos, integrando as ideias de algoritmos, estruturas de dados básicas (arrays e registros) e programação estruturada em uma linguagem procedimental; no segundo semestre, um estudo de Programação, integrando programação orientada a objetos, algoritmos e estruturas de dados e projeto de sistemas. O EI de Programação é composto por três módulos teóricos de 30 horas cada (1. Algoritmos e Programação II, 2. Estruturas de Dados; 3. Projeto de Sistemas) e um módulo integrador de 60 horas (4. MI - Programação) com duas sessões tutoriais semanais de duas horas de duração. A introdução de Projeto de Sistemas no estudo integrado de Programação é recente, sendo aplicada desde 2012. Sua inclusão ocorreu para tratar a complexidade na mudança de paradigma, através da discussão de modelos conceituais e diagramas de classes de projeto. Adicionalmente, por questões de organização curricular, o EI, que era oferecido no terceiro período do curso, passou a ser oferecido no segundo.

O objetivo geral do EI de Programação é que o estudante seja capaz de projetar e desenvolver software orientado a objetos, utilizando apropriadamente algoritmos e estruturas de dados, com domínio dos conceitos e fundamentos subjacentes às metodologias e ferramentas utilizadas. Objetivos específicos são desdobrados do objetivo geral, a partir das competências específicas desejadas (e.g., ser capaz de escrever, compilar e depurar programas orientados a objetos em Java; ser capaz de escolher e implementar estruturas de dados apropriadas a um problema de busca).

O planejamento do EI de Programação segue uma espiral de complexidade crescente, onde habilidades e conhecimentos se acumulam até atingir as competências desejadas. Para isso, são propostos, de modo geral, problemas práticos embasados em situações reais, com duração apropriada para permitir a aquisição de conhecimentos e a prática de habilidades de programação. Geralmente, cada problema dura entre três e seis semanas e engloba um recorte dos conteúdos e habilidades específicos desdobrados, respectivamente, da ementa e do objetivo geral. Em paralelo ao módulo integrador, os módulos teóricos apresentam e discutem assuntos específicos, normalmente encadeados

com a teoria necessária ao desenrolar dos problemas, mas também envolvendo outros aspectos que eventualmente não sejam abordados nos problemas.

A execução do EI ocorre com a divisão dos estudantes em grupos tutoriais com até 10 alunos, sendo cada grupo tutorado por um professor. Nestes encontros, a dinâmica da sessão tutorial segue uma dinâmica adaptada do modelo proposto por Barrows (ver Subseção 2.1), descrita em outro artigo por Santos et al. (2007a). A cada semana, ocorre a reunião entre tutores que visa conhecer o andamento dos grupos, possíveis dúvidas e soluções adotadas. Além disso, são discutidas as não conformidades do problema em questão para garantir a uniformidade das avaliações e dos grupos.

No final da sessão, a avaliação do desempenho feita pelo tutor dá o *feedback* aos alunos de acordo com as discussões e a conduta de cada um. São avaliadas as seguintes competências: compartilhamento do aprendizado; participação na discussão; contribuição efetiva; comportamento no grupo e pontualidade. Além disto, são feitas ponderações sobre os papeis específicos assumidos por alguns alunos, como a dinâmica obtida pelo coordenador da sessão, as anotações do secretário de quadro e o relato do secretário de mesa.

Ao final das sessões de um problema ou projeto, cada aluno entrega um produto (e.g., código-fonte e documentação) em conjunto com um relatório. O objetivo do relatório é motivá-lo a escrever coerentemente sobre o que aprendeu e abalizar suas informações em referências bibliográficas apropriadas. Na discussão da solução, o aluno, dependendo do problema, deve informar sucintamente qual é a solução, suas partes e como elas interagem. Também deve explicar os desafios encontrados e mostrar como foram solucionados a partir dos conceitos teóricos aplicados.

A medida final de cada módulo teórico é calculada a partir de uma média ponderada das avaliações de conteúdo do módulo teórico, com peso 7, e dos produtos dos problemas, com peso 3. Já a medida final para o módulo integrador é dada por uma média ponderada dos produtos dos problemas, com peso 7, e do desempenho do aluno nas sessões tutoriais, com peso 3. Maiores detalhes sobre a avaliação são descritas por Santos et al. (2007b) em outro trabalho.

4. Relato de Experiência

Descrevemos, a seguir, a aplicação de nossa abordagem no estudo integrado de Programação nos dois períodos letivos de 2012.

4.1. Primeiro semestre

No primeiro semestre, foram realizados quatro problemas, divididos em dois projetos: uma clínica odontológica e uma rede social. Para cada problema, os alunos deveriam entregar um relatório e o código da implementação.

No primeiro problema, os alunos projetaram e implementaram um sistema para informatizar as clínicas odontológicas da universidade. Para isso, eles receberam *user stories*, testes de unidade e um modelo conceitual prontos, a partir dos quais eles projetaram e implementaram a solução, utilizando conceitos de orientação a objetos, listas encadeadas, filas e os padrões *Iterator* e *Façade*. Notamos que, para os alunos, a quantidade de novos conceitos foi grande, assim como a quantidade de trabalho na

implementação. Foi necessário estender o prazo original de quatro semanas para cinco. Embora os módulos teóricos apresentassem conceitos importantes em paralelo, nem todos os alunos conseguiram adquirir as competências em tempo hábil para aplicar estes conceitos na solução do problema. Entender modelos conceituais e diagramas de classes era relativamente simples para eles, mas, projetar sistemas usando estes artefatos mostrou-se mais difícil.

No segundo problema, *user stories* novas e modificadas eram o ponto de partida para a manutenção e evolução do sistema. Neste caso, eles escreveram seus próprios testes de unidade, modificaram o modelo conceitual e fizeram um diagrama de classes de projeto do sistema, a partir dos quais implementaram o código-fonte utilizando filas com prioridades e algoritmos de ordenação.

Já o segundo projeto, de uma rede social, envolveu mais dois outros problemas. O terceiro problema solicitava o armazenamento dos dados do perfil do usuário e de diferentes tipos de arquivos. Os requisitos eram expressos como *user stories* e testes funcionais em *JUnit*. Este problema introduziu conceitos de reuso de código, árvores binárias balanceadas, leitura e escrita de arquivos e interface de linha de comando.

Finalmente, no quarto problema, os alunos trocaram a interface de linha de comando por uma interface gráfica baseada em padrões de projeto e, adicionalmente, utilizaram grafos para representar as amizades na rede social e realizar buscas.

4.2. Segundo semestre

No segundo semestre, foi feita uma simplificação dos primeiros problemas para diminuir a complexidade das tarefas para os novos aprendizes. Para tanto, reduzimos a quantidade de *user stories* e o tamanho do modelo conceitual. Resolvemos ainda utilizar um projeto para todo o semestre, dividido em vários problemas, objetivando a implementação de um sistema de leilões eletrônicos.

No primeiro problema, os alunos receberam a especificação e análise de um problema de leilão na forma de *user stories*, um modelo conceitual e testes de unidade. A partir destes, implementaram o código utilizando classes e objetos, listas encadeadas e o padrão *Iterator* e *Façade*. Além do código, os alunos deveriam apresentar ainda o diagrama de classes de projeto do sistema.

No segundo problema, foram adicionadas novas funcionalidades ao sistema de leilões, expressas como *user stories* e testes funcionais em *JUnit*. Os alunos fizeram novos testes de unidade ou modificaram os antigos em conjunto com a implementação do código, que envolveu conceitos de filas, ordenação, herança, interfaces, padrão *Singleton* e uma interface de linha de comando. Ao final, além do código, deveriam entregar o modelo conceitual e o diagrama de classes de projeto atualizados.

No terceiro problema, foram dados aos alunos testes funcionais para garantir a persistência dos dados dos leilões. Além disso, novas *user stories* solicitavam a implementação de buscas eficientes com árvores binárias balanceadas e a utilização de uma interface gráfica. Os alunos implementaram o código utilizando estes conceitos, além do uso da API Java para arquivos.

Neste momento, decidimos, para o quarto problema, começar um novo projeto, de determinação de rotas de um metrô. Neste, os alunos tinham mais liberdade para

definir seus requisitos, desde que utilizassem uma interface gráfica e um algoritmo de caminho mínimo em grafos para a determinação das rotas.

4.3. Discussão

Desde o primeiro semestre de 2012, o EI de Programação passou a contar, com outro módulo teórico, o de Projeto de Sistemas, cujos conteúdos foram parcialmente trazidos do antigo módulo de Análise e Projeto de Sistemas, agora oferecido no quarto semestre com a denominação de Análise de Sistemas. Discutimos aqui o impacto destas mudanças no planejamento do EI e na aprendizagem dos alunos neste novo currículo.

De modo geral, pudemos observar, nos dois períodos, algumas vantagens e desvantagens desta abordagem. Como vantagens, podemos citar a familiarização precoce com diagramas de projeto; os benefícios de usar uma abordagem de aprendizagem *objects-first*; uma mudança de perspectiva no processo de desenvolvimento com a utilização de testes de unidade desde o início da aprendizagem de programação; além de uma visão integrada dos conceitos teóricos aplicados aos projetos realizados. Já as desvantagens observadas foram o abandono de parte dos alunos nos primeiros problemas por não conseguirem assimilar todos os conceitos apresentados, pouco tempo para o desenvolvimento de um dos problemas, havendo a necessidade do adiamento da entrega do produto, dificuldade dos alunos em perceberem a necessidade dos padrões de projeto e a utilização tardia da interface gráfica nos projetos. Além disso, percebemos que alguns alunos, ao terminar um problema sem completar todos os requisitos, tiveram dificuldades em dar continuidade ao mesmo projeto no problema subsequente.

No primeiro semestre, o problema inicial demandava a implementação de uma grande quantidade de requisitos e a abordagem de vários conceitos novos, o que levou a maioria dos alunos a não conseguirem entregar a solução dentro do tempo previsto. Corrigimos esta questão no segundo período, reduzindo o tamanho do modelo conceitual a quantidade de requisitos, tornando a resolução menos complexa para os alunos. Além disso, como os conceitos de projeto de sistemas estavam sendo trabalhados por alunos que estavam aprendendo programação orientada a objetos, percebemos que havia um excesso de conceitos de projeto para o estágio em que eles se encontravam. Assim, no segundo semestre, reduzimos vários dos conceitos de projeto de sistemas para o mínimo essencial: diagramas de classes, apenas um diagrama comportamental, projeto com reuso e uma seleção mínima de padrões de projeto.

Apesar das melhorias no segundo semestre, ainda surgiram outros problemas na organização. A decisão de fazer um só projeto para todo o período mostrou-se complexa demais para os alunos, pelas dificuldades próprias de manutenção de software, levando-nos a iniciar um novo projeto no quarto problema. Outra opção seria fazer cada novo problema ser um novo projeto, evitando acúmulo de erros e problemas vindos de um problema anterior.

Por outro lado, embora ainda estejamos procurando a dosagem correta de competências, habilidades e conceitos para o estudo integrado, vale ressaltar os benefícios para os alunos que foram bem-sucedidos nestes dois períodos. Primeiro, eles tiveram um amadurecimento acelerado e proficiência precoce em desenvolvimento de software, trabalharam com ferramentas utilizadas por profissionais em situações de

trabalho, desenvolveram competências de modelagem e projeto que foram utilizadas desde o princípio na criação de seus programas, além de terem tido uma experiência significativa ao realizar projetos motivadores e a gerenciar os prazos de execução de projetos de modo eficiente.

Em relação à avaliação, aspectos tanto processuais como de produto foram levados em conta, sendo observados, entretanto, apenas da perspectiva do tutor. De modo geral, os produtos dos problemas eram descritos através de um relatório técnico acompanhado do código-fonte, documentação em *javadoc*, além dos diagramas de classes de projeto. Já o desempenho era observado durante as sessões tutoriais, levando em conta contribuições efetivas, participação, papéis desempenhados e boa dinâmica de grupo. Embora tenhamos conseguido capturar muito da dinâmica de aprendizagem na avaliação, percebemos a concentração dos processos avaliativos nos relatórios e software entregues, sempre sob o ponto de vista do tutor. Com a experiência acumulada, parece-nos mais adequado modificar estes processos de modo a contemplar aspectos de autoavaliação e avaliação pelos pares, além de uma variedade maior nos tipos de produtos entregues, além do modelo relatório/software.

5. Lições Aprendidas

Integração de conhecimentos. Um dos maiores benefícios observados em nossa abordagem foi a integração de conhecimentos e competências para a produção de software. Em um único semestre, os alunos puderam se familiarizar com programação orientada a objetos, estruturas de dados e projeto de sistemas, em uma organização pedagógica onde teoria e prática caminhavam de mãos dadas. Isto permitiu que eles realizassem projetos mais autênticos, adquirindo experiência de várias etapas do ciclo de vida do software em um modo de trabalho mais disciplinado, o que será útil no desenvolvimento de seus cursos e em suas vidas profissionais.

Competências de PBL. A utilização da metodologia de aprendizagem baseada em problemas e projetos permitiu adquirir um conjunto de competências de comunicação, colaboração, liderança, autodidatismo e raciocínio crítico, dentre outras. A metodologia demanda um esforço maior do estudante, porém resulta na aquisição de mais experiência e independência em comparação a metodologias de ensino tradicionais.

Sequência de problemas. Embora pareça, a princípio, vantajoso utilizar um único projeto ao longo de todos os problemas num dado semestre, problemas típicos de manutenção de software se acumulam com o passar do tempo, gerando dificuldades no processo de aprendizagem. Por não conseguirem concluir problemas anteriores, ao iniciar novos problemas que continuavam o mesmo projeto, caiu o nível de motivação para alguns estudantes, o que levou a algumas desistências.

Excesso de conceitos. Uma dificuldade que estamos tentando equacionar é o excesso de novos conceitos tratados nos problemas por conta da integração de três diferentes componentes curriculares no segundo semestre de um curso de Engenharia de Computação. Por exemplo, é difícil, num primeiro momento, adquirir conceitos de projeto, implementação, testes, estruturas de dados e padrões de projeto. Observamos que alguns alunos tem dificuldade de perceber a aplicabilidade de determinados conceitos nos projetos. A integração de várias disciplinas demanda um melhor planejamento, sendo de especial importância a dosagem de novos conceitos em níveis

de complexidade crescentes e adequados ao estágio do estudante em seu processo próprio de aprendizagem.

6. Conclusões e Trabalhos Futuros

Este artigo relatou uma experiência de integração das disciplinas de programação orientada a objetos, estruturas de dados e projeto de sistemas em um estudo integrado semestral com dez horas semanais, utilizando uma metodologia de aprendizagem baseada em problemas e projetos (PBL). A experiência foi aplicada em dois semestres letivos, e este relato detalha o seu planejamento e implementação, além de uma discussão dos resultados e uma síntese com lições aprendidas.

As principais lições aprendidas foram: a integração de conhecimentos possibilita experiências mais autênticas e práticas de produção de software mais disciplinadas; a metodologia PBL permite adquirir competências mais amplas de comunicação, trabalho em equipe e autodidatismo; problemas de manutenção de software podem reduzir a motivação por acúmulo de deficiências e devem ser evitados; a dosagem de novos conceitos nos problemas propostos deve respeitar um processo gradual e a capacidade de assimilação dos alunos.

Atualmente, está em andamento uma terceira oferta deste estudo integrado, onde reduzimos a quantidade de novos conceitos e equacionamos melhor a divisão dos problemas. Por outro lado, entendemos que alguns conceitos de projeto de sistemas são muito precoces para alunos do segundo período e estamos trabalhando numa reformulação curricular que propõe um componente curricular intermediário entre o atual EI de programação e o EI de engenharia de software, deixando alguns conceitos mais simples no segundo semestre, e elaborando mais sobre conceitos de projeto num novo EI no terceiro semestre.

Finalmente, esta experiência realça a possibilidade de um currículo integrado que permite rotas de aprendizagem mais autênticas e mais próximas das práticas profissionais, trilhadas com o auxílio de uma metodologia de aprendizagem ativa baseada em problemas e projetos, onde os aprendizes são agentes ativos de seu próprio desenvolvimento cognitivo e profissional.

Referências

Angelo, M. F. e Bertoni, F. C. (2012). Análise da aplicação do método PBL no processo de ensino e aprendizagem em um curso de engenharia de computação. Revista de Ensino em Engenharia, 30(2):35–42.

Barrows, H. S. (1986). A taxonomy of problem-based learning methods. Medical education, 20(6):481–486.

Bittencourt, R. A. e Figueiredo, O. A. (2003). O currículo do curso de engenharia de computação da UEFS: Flexibilização e integração curricular. In XI Workshop de Educação em Computação –Anais do XXIII Congresso da Sociedade Brasileira de Computação.

Bloom, B. S., Englehart, M. D., Furst, E. J., Hill, W. H., e Krathwohl, D. R. (1976). Taxonomia de objetivos educacionais. Porto Alegre: Globo.

Boud, D. e Feletti, G. (1998). The challenge of problem based learning. Routledge.

CNE. (2002). Resolução CNE/CES 11/2002. Diário Oficial da União, Brasília, 9 de abril de 2002. Seção 1, p. 32.

Delisle, R. (1997). How to use problem-based learning in the classroom. ASCD.

Duch, B. J., Groh, S. E., e Allen, D. E. (2001). The power of problem-based learning: a practical "how to" for teaching undergraduate courses in any discipline. Stylus.

Ferreira, A., Heinen, F. J., Gaspary, L., and Lemke, N. (2003). Engenharia da Computação: uma proposta transdisciplinar visando o desenvolvimento regional. In XI Workshop de Educação em Computação – Anais do XXIII Congresso da Sociedade Brasileira de Computação.

Gomes, R., Francisco, A. M., Tonhom, S. F. R., Costa, M. C. G., Hamamoto, C. G., Pinheiro, O. L., Moreira, H. M., Hafner, M. L. M. B. (2009). A formação médica ancorada na aprendizagem baseada em problemas: uma avaliação qualitativa. Interface: comunicação, saúde, educação, 13(28):71–83.

Oliveira, W. L. A., de Arruda, G. H. M., e Bittencourt, R. A. (2007). Uso do método PBL no ensino de arquitetura de computadores. In ICECE'2007 – International Conference on Engineering and Computer Education.

Santos, D. M. B., Pinto, G. R. P. R., Sena, C. P. P., Bertoni, F. C., e Bittencourt, R. A. (2007a). Aplicação do método de aprendizagem baseada em problemas no curso de engenharia de Computação da Universidade Estadual de Feira de Santana. In COBENGE 2007 XXXV Congresso Brasileiro de Educação em Engenharia.

Santos, D. M. B., Saba, H., Rocha Jr., J. B., and Sarinho, V. T. (2007b). Integrando as disciplinas de engenharia de software, análise e projeto de sistemas e banco de dados utilizando PBL. In XV Workshop sobre Educação em Computação – Anais do XXVII Congresso da Sociedade Brasileira de Computação, p. 66–75.

Santos, J. A. M., Angelo, M. F., e Loula, A. C. (2008). Experiências em um estudo integrado de programação usando PBL. In XVI Workshop de Educação em Computação – Anais do XXVIII Congresso da Sociedade Brasileira de Computação.

Santos, J. A. M. e Angelo, M. F. (2009). Análise de problemas aplicados em um estudo integrado de programação utilizando PBL. In XXIX Congresso da Sociedade Brasileira de Computação - XVII Workshop sobre Educação em Computação.

Silveira, M.A. (2005). A formação do engenheiro inovador: uma visão internacional. Rio de Janeiro: PUC-Rio.

Svinicki, M. e McKeachie, W. (2010). McKeachie's teaching tips: Strategies, research, and theory for college and university teachers. Wadsworth Pub Co.

Uden, L. e Beaumont, C. (2006). Technology and problem-based learning. Information Science Pub.

UEFS (2012). Currículo do curso de Engenharia de Computação da UEFS. Disponível em <http://www.ecomp.uefs.br>. Acessado em 10/03/2012.

Woods, D. R. (1994).Problem-based learning: How to gain the most from PBL. Donald R. Woods.