ANAIS PROCEEDINGS

CBSOFT * 2015

BRAZILIAN CONFERENCE ON SOFTWARE: THEORY AND PRACTICE

BELO HORIZONTE

FEES 2015

VIII FÓRUM DE EDUCAÇÃO EM ENGENHARIA DE SOFTWARE

CBSOFT.ORG

Sponsors:

ThoughtWorks · AvenueCode · Google Brasil · take.net · RARO LABS · Avanti · IBM

Promotion:

SBC Sociedade Brasileira de Computação

Organizing Institutions:

UFMG · PUC Minas · CEFET-MG

CNPq Conselho Nacional de Desenvolvimento Científico e Tecnológico · CAPES · FAPEMIG

FEES 2015

VIII FÓRUM DE EDUCAÇÃO EM ENGENHARIA DE SOFTWARE
24 e 25 de setembro de 2015
Belo Horizonte – MG, Brasil

VOLUME 01
ISSN: 2175-9677

ANAIS | *PROCEEDINGS*

COORDENADORES DO COMITÊ DE PROGRAMA DO FEES 2015 | *PROGRAM COMMITTEE CHAIRS OF FEES 2015*
Ingrid Nunes (UFRGS)
Jair Cavalcanti Leite (UFRN)

COORDENADORES GERAIS DO CBSOFT 2015 | *CBSOFT 2015 GENERAL CHAIRS*
Eduardo Figueiredo (UFMG)
Fernando Quintão (UFMG)
Kecia Ferreira (CEFET-MG)
Maria Augusta Nelson (PUC-MG)

REALIZAÇÃO | *ORGANIZATION*
Universidade Federal de Minas Gerais (UFMG)
Pontíficia Universidade Católica de Minas Gerais (PUC-MG)
Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)

PROMOÇÃO | *PROMOTION*
Sociedade Brasileira de Computação | Brazilian Computing Society

APOIO | *SPONSORS*
CAPES, CNPq, FAPEMIG, Google, RaroLabs, Take.net,
ThoughtWorks, AvenueCode, AvantiNegócios e Tecnologia.

APRESENTAÇÃO

A educação em Engenharia de Software que tradicionalmente ocorre em cursos de ciência e engenharia de computação tem se expandido para cursos tecnológicos, bacharelados em Engenharia de Software, residências, mestrados profissionais e treinamentos corporativos. Todas essas diferentes formas de processos educativos requerem orientações provenientes tanto da academia quanto da indústria e tem sido apoiada e alavancada por esforços de diversas instituições e sociedades educacionais e científicas da área da computação. Nos últimos anos, no Brasil, foram criados alguns bacharelados em Engenharia de Software. Com esse cenário de expansão e avanços do ensino-aprendizagem e pesquisa em Engenharia de Software no Brasil, é importante reunir educadores/professores, pesquisadores e estudantes para trocarem suas experiências e conhecimentos.

O Fórum de Educação em Engenharia de Software (FEES) visa ser um espaço e um momento em que, através das publicações, apresentações e painéis das nossas pesquisas e relatos de experiências, podemos como comunidade científica e educacional, avançarmos no ensino-aprendizagem em Engenharia de Software, em particular no contexto brasileiro.

O FEES 2015 foi motivado por importantes desafios, dentre as quais destacamos:

- Indicar diretrizes curriculares para os Bacharelados em Engenharia de Software
- Identificar perfil do profissional de software na indústria brasileira
- Discutir a formação de profissionais para um mundo globalizado
- Discutir a rápida evolução tecnológica nos cursos e como isto afeta as estruturas e os componentes curriculares
- Refletir sobre os paradigmas de “design thinking”, “engineering thinking e “system thinking” no contexto do ensino em computação e engenharia de software
- Discutir técnicas e práticas de ensino para as situações que envolvem sistemas complexos, trabalhos em times e em longo prazo, em outras palavras, ensinar em pequena escala o que só é relevante em grande escala

A partir destas motivações, o comitê de programa do FEES 2015 selecionou 6 artigos completos e 3 artigos curtos de um total de 17 submissões. Todos os artigos estão publicados nestes anais. Os temas relacionados ao ensino e engenharia de software (técnicas, ferramentas, práticas e experiências) foram os predominantes nas submissões.

Na programação do FEES 2015, além das apresentações dos trabalhos, temos uma palestra convidada *Teaching Practical Software Engineering in the 21st Century* proferida pelo professor Armando Fox, professor no departamento de engenharia elétrica e ciência

da computação da Universidade da Califórnia, em Berkeley, e é um líder reconhecido na educação online e em MOOC's (Massive Open Online Courses).

Ainda faz parte da programação a seção sobre os refinamentos de habilidades e competências dos egressos dos cursos de engenharia de softwares, coordenada pelo Prof. Daltro Nunes (UFRGS).

Gostaríamos de agradecer aos autores e revisores, cujo cuidadoso trabalho permitiu a seleção de artigos de qualidade em temas que certamente vão promover a discussão sobre a educação e estão aqui publicados para futuras referências. Agradecemos, também, o apoio dos coordenadores do CBSOFT 2015 Eduardo Figueiredo, Fernando Pereira, Kecia Ferreira e Maria Augusta Nelson, e ao coordenador do SBES 2015, Leonardo Murta.

Esperamos que o FEES 2015 seja uma excelente experiência para todos.

Belo Horizonte, setembro de 2015.

Ingrid Nunes (UFRGS)

Jair Cavalcanti Leite (UFRN)

Coordenadores do Comitê de Programa do FEES 2015

COMITÊ DE ORGANIZAÇÃO | *ORGANIZING COMMITTEE*

CBSOFT 2015 GENERAL CHAIRS

Eduardo Figueiredo (UFMG)
Fernando Quintão (UFMG)
Kecia Ferreira (CEFET-MG)
Maria Augusta Nelson (PUC-MG)

CBSOFT 2015 LOCAL COMMITTEE

Carlos Alberto Pietrobon (PUC-MG)
Glívia Angélica Rodrigues Barbosa (CEFET-MG)
Marcelo Werneck Barbosa (PUC-MG)
Humberto Torres Marques Neto (PUC-MG)
Juliana Amaral Baroni de Carvalho (PUC-MG)

WEBSITE AND SUPPORT

Diego Lima (RaroLabs)
Paulo Meirelles (FGA-UnB/CCSL-USP)
Gustavo do Vale (UFMG)
Johnatan Oliveira (UFMG)

COMITÊ TÉCNICO | *TECHNICAL COMMITTEE*

COORDENADORES DO COMITÊ DE PROGRAMA | *PC CHAIRS*

Ingrid Nunes (UFRGS)
Jair Cavalcanti Leite (UFRN)

COMITÊ DE PROGRAMA | *PROGRAM COMMITTEE*

Ana Paula Bacelo (PUCRS)
Ana Regina Rocha (COPPE/UFRJ)
Christina Chavez (Universidade Federal da Bahia)
Claudia Werner (COPPE/UFRJ)
Daltro Nunes (UFRGS)
Edmundo Spoto (UFG - Universidade Federal de Goiás)
Eduardo Almeida (C.E.S.A.R/RiSE)
Eduardo Figueiredo (Federal University of Minas Gerais (UFMG))
Ellen Francine Barbosa (ICMC-USP)
Francisco Dantas (State University of Rio Grande do Norte)
Heitor Costa (Federal University of Lavras)
Leila Ribeiro (Universidade Federal do Rio Grande do Sul)
Marcelo Barbosa (PUC Minas)
Marcelo Pimenta (Universidade Federal do Rio Grande do Sul)
Marcelo Yamaguti (PUCRS)
Marco Wehrmeister (UTFPR)
Marco Aurélio Graciotto Silva (Universidade Tecnológica Federal do Paraná)
Mauricio Aniche (IME/USP)
Milene Serrano (Universidade de Brasília (UnB/FGA))
Paulo Meirelles (Universidade de Brasília)
Rafael Prikladnicki (PUCRS)
Regina Braga (Universidade Federal de Juiz de Fora)
Rodolfo Resende (DCC/UFMG)
Rodrigo Reis (UFPA)
Sandro Andrade (Federal Institute of Education, Science and Technology of Bahia (IFBa))
Simone Barbosa (PUC-Rio)
Tayana Conte (UFAM)
Thais Vasconcelos Batista (UFRN)
Thiago Mendes (IFBA - Instituto Federal de Educação, Ciência e Tecnologia da Bahia)
UiráKulesza (UFRN - Universidade Federal do Rio Grande do Norte)
Valter Camargo (Universidade Federal de São Carlos)
Vera Werneck (Universidade do Estado do Rio de Janeiro)

REVISORES EXTERNOS | *EXTERNAL REVIEWERS*

Claudia Susie Rodrigues (COPPE/UFRJ)
Gustavo Vale (Universidade Federal de Minas Gerais)
Ivonei Silva (Universidade Estadual do Oeste do Paraná - Unioeste)
Mauricio Arimoto (University of São Paulo (ICMC/USP))

PALESTRA CONVIDADA | *INVITED TALK*

Teaching Practical Software Engineering in the 21st Century

ArmandoFox(University of California at Berkeley, USA)

Abstract: From 2008 to 2010, Armando Fox and David Patterson refocused UC Berkeley's 14-week undergraduate software engineering course on agile development, emphasizing behavior-driven design (BDD), automated testing, and team skills for working with nontechnical customers, all with a strong "learn by doing" implementation. This talk describes how the classroom experience of the course imparts the fundamental skills a software developer must have in the 21st century. In addition to a series of scaffolded and autograded programming assignments (including one on enhancing legacy code), a key part of the course is an open-ended project in which students work in "two-pizza teams" (6 per team in our case) building a real SaaS prototype application for a nonprofit, NGO, or campus unit. The project not only lets students practice their new skills in a realistic setting, but also imparts the critical experience of working with nontechnical customers. Students convert customers' business needs into a design and implementation via an incremental process based on user stories and lo-fi mockups with customer participation and approval; they meet with both their customer and their Teaching Assistant during each 2-week iteration. Projects are deployed on the public cloud, so the customer can always see latest version, students can point to it as part of their work, and many customers continue to use the app after the course ends.
I will describe how the automation around the scaffolded assignments has allowed us to "scale up" project supervision and offer the course to many more students. Each 20-hour-per-week teaching assistant can comfortably handle both the scaffolded assignments and the team projects for up to 50 students. This scale-up is critical since students, instructors, and employers agree that the project is a critical element of the learning experience. The course has led to a successful inexpensive textbook that can be used to teach the material in a manner that fulfills the 2013 ACM/IEEE curriculum guidelines for software engineering. The result is a practical approach to teaching software engineering that scales well, can be readily transferred to other instructors and classrooms using SPOCs (Small Private Online Courses, which exploit MOOC technology), and is praised by students, instructors, and employers.

Armando Fox (fox@berkeley.edu) is a Professor in the Electrical Engineering and Computer Science Department of the University of California at Berkeley. He is also the Faculty Advisor to the UC Berkeley MOOCLab and a recognized thought leader on MOOCs and online education, topics on which he has had the honor of addressing the California legislature, the China Ministry of Education, and the Japan Top Global University Project, as well as numerous US and international universities. He also serves on the Technical Advisory Committee of edX, helping to set the technical direction of their open MOOC platform, and the Google Online Education Advisory Council. With his colleague David Patterson, he co-designed and co-taught Berkeley's first Massive Open Online Course on

"Engineering Software as a Service", offered through edX, through which over 10,000 students in over 120 countries have earned certificates of completion. The MOOC is based on Fox and Patterson's overhaul of Berkeley's software engineering course, focusing on agile projects with real customers and heavily guided by modern software practice. Their course and accompanying textbook, "Engineering Software as a Service" (now available in several languages), influenced the ACM/IEEE 2013 Curriculum Guidelines for Software Engineering and the 2014 IEEE Software Engineering Competency Model (SWECOM). He has given numerous keynotes about these topics at SIGCSE, ICSE, CSEET, and other leading venues on computer science education.

His current research in online education includes automatic grading of students' computer programs for style as well as improving engagement and learning outcomes in MOOCs. He is a co-founder of the new conference Learning At Scale, with colleagues Prof. Marti Hearst (UC Berkeley) and Prof. Michelene Chi (Arizona State University).

His other computer science research in the Berkeley ASPIRE project focuses on highly productive parallel programming. While at Stanford he received teaching and mentoring awards from the Associated Students of Stanford University, the Society of Women Engineers, and Tau Beta Pi Engineering Honor Society. He is an alumnus of MIT (BS in EECS), the University of Illinois (MS in EE), and UC Berkeley (PhD in CS). He is a classically-trained musician and performer, an avid musical theater fan and freelance Music Director, and bilingual/bicultural (Cuban-American) New Yorker living in San Francisco.

ARTIGOS COMPLETOS |*FULL PAPERS*

ARTIGOS CURTOS | *SHORT PAPERS*

Desafios do desenvolvimento de atividades práticas de Engenharia de Software em grupo em cursos a distância

Marcelo Werneck Barbosa[1], Maria Augusta Vieira Nelson[1]

[1]Instituto de Ciências Exatas e Informática – Pontifícia Universidade Católica de Minas Gerais (PUC Minas) Belo Horizonte – MG – Brasil

{mwerneck, guta}@pucminas.br

Abstract. *The goal of this paper is to present the challenges and experiences of running a group assignment spanning three courses of a distance education program that involve a typical software development cycle. At the end of these courses, the participating students answered a questionnaire. It was possible to identify that the students recognize how the project contributed to their theoretical and practical learning, though they questioned the presence of group work in a distance course. In general, students invested a large amount of time in the project and recommended it to upcoming cohorts. We also present lessons learned after conducting this project.*

Resumo. *Este artigo tem como objetivo apresentar as experiências e desafios da condução de um trabalho realizado ao longo de três disciplinas de um curso a distância que envolvem o ciclo típico de desenvolvimento de software. Ao final destas disciplinas, os alunos participantes responderam a um questionário. Foi possível identificar que os alunos reconhecem a contribuição do trabalho para seu aprendizado teórico e prático, apesar de questionarem a existência de um trabalho em grupo em um curso a distância. Os alunos, em geral, investiram uma carga horária considerável no trabalho e informaram recomendá-lo para as turmas seguintes. A execução deste trabalho também permitiu a coleta de algumas lições aprendidas.*

1. Introdução

A educação vem passando por uma fase expressiva de mudanças. O papel do professor, a metodologia de ensino e até mesmo a função da escola têm sido objeto de discussão no mundo inteiro. Nesse sentido, as inovações tecnológicas, principalmente a Internet e a sua inserção no cotidiano da sociedade, assumem uma posição cada vez mais importante. A Educação a Distância (EaD) é uma modalidade que tem buscado conciliar o uso das tecnologias ao processo educacional, com o intuito de ampliar as possibilidades educacionais. No contexto da EaD, ainda são muitas as questões em análise, como, por exemplo, quais são as atribuições do docente, o que se altera no seu fazer e como ele lida com as novas tecnologias educativas; qual o impacto das interfaces informatizadas no processo ensino-aprendizagem, do ponto de vista cognitivo; quais as estratégias elaboradas pelos alunos para navegar nas interfaces informatizadas; e, ainda, como a análise da atividade pode contribuir para melhorar as condições de trabalho dos professores e a qualidade das ferramentas disponibilizadas (FLEURY et al., 2014).

No ensino presencial, o convívio entre as pessoas e a troca de experiências por meio de diálogo auxiliam no processo de ensino e podem fornecer a bagagem necessária para os desafios que serão enfrentados após a conclusão do curso. Contudo, paralelo a isso, no ensino a distância, o aluno tem a possibilidade de regrar o seu estudo e conciliar com sua condição de trabalho ou financeira, não tendo a necessidade de contato direto com outras pessoas diariamente, como no ensino presencial (COSTA et al., 2014).

O ensino a distância viabiliza formas de aprendizagem para cada tipo de assunto para que o aluno desenvolva sua autonomia intelectual no processo de aquisição de conhecimento. No processo de aprendizagem democrático da EaD, pode-se dizer que a autonomia requer disciplina, organização, persistência, responsabilidade e automotivação. O ensino a distância possibilita a eficácia de cada aluno por flexibilizar o horário para o estudo e respeitar o período de concentração e interesse individual de estudo, potencializando o desenvolvimento intelectual. Neste contexto, visando à aprendizagem, metodologias estão sempre se inovando, as avaliações da aprendizagem sendo aperfeiçoadas constantemente e respeitando as normatizações da educação à distância (FONSECA et al., 2014).

O processo de ensino e aprendizagem de Engenharia de Software (ES) tem passado por questionamentos; mas há um consenso de que o ensino de ES, tradicional e focado em metodologias, deve ser transformado para refletir a demanda por software mais complexo preparando os estudantes para participações efetivas em ambientes colaborativos e interdisciplinares (SANTOS et al., 2008). A qualidade da educação em Engenharia de Software pode contribuir significativamente para a 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. 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. As abordagens mais comuns para ensinar Engenharia de Software 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, dinâmicas de grupo e o uso de jogos (PRIKLADNICKI et al., 2009).

Atualmente a disciplina de Engenharia de Software é ensinada nos cursos da área de computação no Brasil, sejam eles cursos cuja computação é uma atividade-meio (Sistemas de Informação e Licenciatura em Computação) ou cursos cuja computação é uma atividade-fim (Ciencia da Computação e Engenharia de Computação) (SOARES, 2004) assim como em cursos de pós-graduação (CUNHA et al., 2008) (PRIKLADNICKI et al., 2009). Os alunos egressos da disciplina de Engenharia de Software devem demonstrar habilidades para gerenciar projetos de software e analisar, projetar, verificar, validar, implementar e manter sistemas de software (SOARES, 2004).

Muitas vezes, o ensino de tais habilidades se dá por meio de projetos. A prática pedagógica por meio do desenvolvimento de projetos é uma forma de estabelecer um ambiente de aprendizagem criado para promover a interação entre todos seus elementos, propiciar o desenvolvimento da autonomia do aluno e a construção de conhecimentos de distintas áreas do saber, na busca de informações significativas para compreensão e

resolução de uma situação-problema (CUNHA e JUNIOR, 2007). Assim,neste contexto, além das habilidades técnicas, tais como conhecimento de métodos, ferramentas, métricas e linguagens de programação, os alunos devem ter habilidades de comunicação e de trabalho em equipe, que sao típicas de um ambiente de desenvolvimento de software (SOARES, 2004). Enquanto o aprendizado tradicional é apresentado para habilidades específicas do indivíduo, a maioria das competências virtuais necessitam ser desenvolvidas em equipes, através de uma rede (MATTOS e TAVARES, 1999).

A prática de técnicas de desenvolvimento de software em ambientes virtuais de ensino possui características comuns com o desenvolvimento distribuído de software. Em ambos os contextos, um dos aspectos chave é a comunicação entre as equipes. A comunicação afeta o desempenho das atividades no projeto e gera atrasos na sua execução. Como as equipes podem ter um tempo de sobreposição de horário muito curto, a identificação de responsáveis por responder dúvidas ou equívocos entre os desenvolvedores (e alunos) pode ser uma grande oportunidade para diminuir os atrasos gerados na comunicação assíncrona entre equipes de projetos distribuídos (TRINDADE, MORAES e MEIRA, 2008).

Neste sentido, este artigo tem como objetivo apresentar as experiências de condução de um trabalho realizado ao longo de três disciplinas do curso a distância de Desenvolvimento de Aplicações Web da PUC Minas. O trabalho envolveu disciplinas do eixo da Engenharia de Software relacionadas a Requisitos de Software, Projetos de Sofware e Testes e Implantação de Software.

O restante do texto está organizado da seguinte forma. A Seção 2 discute o referencial do trabalho com destaque para outras experiências similares de outros trabalhos. A Seção 3 explica a estrutura do projeto do trabalho. A Seção 4 apresenta os resultados alcançados enquanto a Seção 5 conclui e discute os trabalhos futuros.

2. Fundamentação Teórica

A educação a distância traz consigo alguns conceitos que precisam ser apresentados para que os desafios enfrentados pelos alunos e professores na condução das atividades em grupo, fiquem esclarecidos. Esta seção apresenta trabalhos relacionados aos principais conceitos base da educação a distância no tocante à experiência apresentada neste trabalho.

2.1 Ambientes Virtuais de Aprendizagem (AVA)

Nos últimos anos, os Ambientes Virtuais de Aprendizagem (AVAs) estão sendo cada vez mais utilizados no âmbito acadêmico e corporativo como uma opção tecnológica para atender uma demanda educacional. Os AVAs consistem em mídias que utilizam o ciberespaço para veicular conteúdos e permitir interação entre os atores do processo educativo. Os AVAs provem recursos para dispor grande parte dos materiais didáticos nos mais diferentes formatos, podendo ser elaborados na forma escrita, hipertextual, oral ou áudio-visual. Qualquer ambiente virtual de aprendizagem deve permitir diferentes estratégias de aprendizagem, não só para se adequar ao maior número possível de pessoas, que terão certamente estratégias diferenciadas, mas também porque as estratégias utilizadas individualmente variam de acordo com fatores como interesse, familiaridade com o conteúdo, estrutura dos conteúdos, motivação e criatividade, entre

outros. Além disso, devem proporcionar a aprendizagem colaborativa, interação e autonomia (MESSA, 2010).

2.2 Interdisciplinaridade

O ensino disciplinar, fragmentado, baseado em informações isoladas e, em muitos casos, distante de um contexto mais abrangente, não atende mais às exigências do mundo atual. A centralização do ensino nas disciplinas é reflexo de um paradigma anterior, industrial, que já não consegue construir um sujeito preparado para o trabalho contemporâneo ou, mais além, para a vida (Pinto et al, 2010). É salutar migrar de uma visão fragmentada para uma globalizada; do disciplinar para o inter e transdisciplinar; e assim permitir o desenvolvimento das potencialidades intelectuais que conduzam o aluno ao paradigma do aprender a aprender para que ele venha a ser sujeito de sua própria aprendizagem (CUNHA et al, 2008). Nesta perspectiva, a interdisciplinaridade é adotada como uma estratégia capaz de romper as estruturas de cada disciplina isolada para alcançar uma visão unitária e comum do saber trabalhado.

2.3 Trabalhos Relacionados

Existem na literatura alguns relatos sobre a execução de trabalhos interdisciplinares em cursos de informática presenciais assim como alguns trabalhos realizados em cursos à distância. Nesta seção, é apresentado um resumo destes bem como uma comparação com o trabalho proposto.

Em Pinto et al (2010), são apresentados resultados do projeto de aplicação de interdisciplinaridade no curso de Ciência da Computação da UNIFESO. A metodologia utilizada é a Aprendizagem Baseada em Problemas (ABP). Neste curso, a prática interdisciplinar tem se configurado como significativa para a formação discente, mas questões levantadas por professores e alunos mostram que muitos ajustes ainda são necessários para atingir uma interdisciplinaridade plena. Os alunos reportaram preferência por uma condução mais diretiva do processo ensino-aprendizagem; mas reclamaram da falta de interesse de alguns colegas durante as atividades em equipe.

Cunha e Junior (2007) apresentaram resultados qualitativos e quantitativos de projetos interdisciplinares sob a perspectiva dos alunos e compararam estes resultados com a visão dos professores no CEFET-AL. Os alunos reportaram dificuldade em obter atenção dos professores em horários extra-classe para esclarecer dúvidas; demandaram a participação igualitária dos membros da equipe nas tarefas e acreditam que há concentração de atividades no final do período. Além disso, apontaram como problema ainda a ausência de modelos de projetos para orientar os alunos e a existência de divergências entre os professores sobre as exigências e forma de avaliação do projeto.

Barbosa, Nelson e Alonso (2012) apresentam um trabalho interdisciplinar no curso de graduação de Sistemas de Informação da PUC Minas, orientado por um processo de gerência de projetos. A proposta deste trabalho interdisciplinar envolveu inicialmente três disciplinas do curso: Projeto de Sistemas de Informação (PSI - quinto período), Gerência de Projetos de Software (GPS) e Engenharia de Software II (ESII). A interdisciplinaridade é realizada em parte através da definição de um processo formal centrado nas atividades de gerência de projetos, documentado na ferramenta EPF (*Eclipse Process Framework*) e disponibilizado aos alunos em HTML. O projeto

consiste no desenvolvimento de um sistema com base em uma especificação de requisitos. Os alunos são divididos em grupos e seu trabalho é gerenciado por um grupo de gerentes (da disciplina de Gerência de Projetos). Os alunos de Engenharia de Software II participam provendo a planilha de pontos de função e atuando em atividades relacionadas à qualidade de software, como inspeção de código.

O foco do estudo descrito em Araújo e Cunha (2014) recai sobre o papel do professor como mediador, nos fóruns de discussão realizados em Ambientes Virtuais de Aprendizagem (AVA), tendo-se como contexto um curso de formação continuada de professores, em nível de Especialização, na modalidade EaD. Foram investigados as categorias e os indicadores específicos de mediação, considerando-se, ainda, os elementos concernentes às Presenças Cognitiva, Social e de Ensino, dentro de uma comunidade de aprendizagem/investigação. De modo geral, pode-se verificar que as mediações têm sido realizadas, primordialmente, pelo professor, consistindo de feedback positivo (elogio em relação à participação discente), bem como de expressões de incentivo ao debate. A partir de tais intervenções, notou-se um certo incremento na participação dos alunos, por meio de postagens de turnos de efetiva interação dialógica professor-aluno/aluno-aluno. Destacaram-se ainda os turnos de mediação nos quais o professor propõe situações colaborativas de aprendizagem, promove interações dialógicas, orientando o processo de construção de saberes e inventando, assim, outros/novos modos de implicar e envolver os alunos no ambiente virtual de aprendizagem.

Em Prikladnicki et al. (2009), a partir de experiências reais de quatro diferentes Instituições, apresentam-se estratégias de ensino de Engenharia de Software usando uma abordagem participativa com foco no aluno. Os autores recomendam 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.

Não foram encontrados na literatura trabalhos que propussem e avaliassem estratégias para trabalhos práticos em disciplinas virtuais de Engenharia de Software. Considerando a experiência dos professores-autores com disciplinas presenciais de Engenharia de Software assim como também em disciplinas virtuais do mesmo tema, este trabalho descreve a proposta e execução de trabalho interdisciplinar em um curso de especialização a distância sobre Desenvolvimento de Aplicações Web. A proposta de trabalho é discutida na próxima seção.

3. Metodologia

Este trabalho adotou a abordagem de pesquisa-ação envolvendo os professores do curso a distância de pós-graduação em Desenvolvimento de Aplicações Web da PUC Minas. A pesquisa-ação é uma abordagem voltada para o estudo da resolução de questões sociais e organizacionais diretamente com quem as vivencia. Assim, a pesquisa-ação não tem como objetivo encontrar respostas gerais, no sentido positivista do termo, mas produzir conhecimento particular, situacional e ligado à práxis. A pesquisa-ação educacional é principalmente uma estratégia para o desenvolvimento de professores e pesquisadores de modo que eles possam utilizar suas pesquisas para aprimorar seu ensino e, em decorrência, o aprendizado de seus alunos (WARDEN et

al., 2013). Como instrumentos de coleta de dados, foram elaborados e aplicados questionários de satisfação em relação às atividades propostas aos participantes.

O projeto envolvendo as disciplinas na sua modalidade a distância teve como ponto de partida a análise das dinâmicas de ensino presenciais; e, ao mesmo tempo, buscou verificar se a incorporação das tecnologias educacionais promovidas pelo ensino a distância permitiria o desenvolvimento deste tipo de trabalho. A descrição do trabalho interdisciplinar é apresentada na seção 4.2. O questionário de avaliação do trabalho foi elaborado com o objetivo de identificar a percepção dos alunos quanto a forma de realização do trabalho (individual ou em grupo); a carga de trabalho demandada, a contribuição para seu aprendizado teórico e prático, a comunicação com o professor; as informações providas e sua satisfação geral.

4. Proposta do Trabalho

Esta seção apresenta a descrição do trabalho utilizada na condução das atividades em grupo que foram propostas aos alunos.

4.1. A Especialização a Distância em Desenvolvimento de Aplicações Web

O curso de especialização em Desenvolvimento de Aplicações Web busca capacitar os alunos para o domínio das tecnologias Web, em especial aquelas relacionadas ao desenvolvimento de aplicações, bem como para a adoção de melhores práticas em Engenharia de Software para a Web. Além disso, tem-se observado uma busca por novos serviços e aplicações para a Web, que exigem que os profissionais da área, além de serem capazes de lidar com as tecnologias, saibam empregar os métodos, as técnicas e os padrões de projeto da Engenharia de Software para aumentar o grau de sucesso dos seus projetos.

O curso tem uma duração de 3 semestres. Em seu segundo semestre, são oferecidas disciplinas da Engenharia de Software chamadas de Modelagem de Aplicações Web, Projeto de Aplicações Web e, por fim, Construção e implantação de aplicações Web. A disciplina de Modelagem de Aplicações Web trata de temas relacionados à introdução à engenharia de software, como processos e ciclos de vida, assim como requisitos e análise de software. A disciplina de Projeto de Aplicações Web aborda o projeto de sistemas, a arquitetura de software, a arquitetura da informação e projeto de interfaces e usabilidade. A última disciplina do eixo, Construção e Implantação de Aplicações Web cuida de temas da gerência de configuração, testes de software e métricas do desenvolvimento de software.

Cada disciplina foi dividida em quatro unidades e teve a duração de quatro semanas. Para cada unidade, foram gravados vídeos no tempo total de uma hora por unidade aproximadamente. Os alunos apresentaram suas dúvidas e discussões por meio de fóruns. Como mecanismos de avaliação, cada unidade, ao final, possuiu uma atividade objetiva com seis questões e, ao longo da disciplina, os alunos realizaram a atividade aberta, que foi entregue ao final da disciplina. Esta atividade aberta é o foco deste estudo por ser o trabalho realizado entre as três disciplinas. Ao final do semestre, os alunos foram submetidos a uma prova presencial.

4.2. O Trabalho Interdisciplinar

A proposta do trabalho consistiu em organizar os alunos em grupos de até 5 membros para o desenvolvimento de um sistema cujo propósito geral foi fomentar o turismo no recém-reinaugurado Cine Theatro Brasil Vallourec em Belo Horizonte por meio da divulgação de informações atuais e históricas sobre o Cine Theatro, que é uma das construções que compõem o patrimônio histórico da cidade de Belo Horizonte.

Com o propósito de contribuir para a divulgação do local e de sua história, cada grupo foi convidado a desenvolver um sistema que permitisse que o visitante conhecesse melhor os espaços do Cine Brasil e também de coletar estórias que pessoas viveram relacionadas ao cinema. Segundo o cenário recebido pelos alunos, serão instalados computadores no saguão do teatro com acesso a esta aplicação para o público geral. Este módulo, chamado de Módulo Usuário, também poderá ser acessado via internet. Deve ser construído ainda um Módulo Administrador para cadastrar algumas informações no sistema, que deverão ser acessadas pelo Módulo Usuário. Os alunos receberam os requisitos da aplicação. Para cada uma das três disciplinas, os alunos tiveram que, de acordo com os conteúdos e técnicas apresentados nas vídeoaulas e discussões dos fóruns, produzir um conjunto de entregáveis. Para a atividade aberta em cada disciplina, foi criado um fórum específico para discussão do trabalho entre os membros do grupo. O professor teve acesso às discussões de todos os grupos e respondeu dúvidas do grupo ou apontou direções aos grupos durante a confecção do trabalho.

Em geral, a primeira semana da disciplina foi utilizada para disponibilizar o enunciado do trabalho, organizar os alunos em grupo (em alguns casos os grupos precisaram ser reorganizados de uma disciplina para a outra, mas na maioria dos casos foram mantidos), esclarecer dúvidas a respeito da dinâmica. As três semanas seguintes eram utilizadas nas discussões em grupo no fórum de discussão onde cada grupo teve seu espaço privado. Ali os alunos postavam soluções parciais que eram pré-avaliadas pelo professor e este provia direções indicando o que precisava ser melhorado ou o que tinha sido omitido. Desta forma, os alunos completavam as soluções e podiam fazer diversas iterações desenvolvendo os entregáveis, até se darem por satisfeitos. Como várias partes do trabalho consistiam em diagramas e modelos sugeriu-se aos alunos utilizarem a ferramenta de modelagem Astah Community (versão gratuita), para permitir trocarem seus modelos e completarem os modelos dos colegas.

Observou-se que muitas vezes os alunos queriam trabalhar de forma síncrona e isso foi desencorajado exatamente porque se sabe que várias equipes no mundo real não podem trabalhar de forma síncrona. O fórum de discussão não é uma ferramenta que exige sincronismo. Muitas vezes os alunos procuravam por ferramentas síncronas fora do ambiente de fórum de execução. Entre estas ferramentas podemos citar o WhatsApp. Tais ferramentas não foram proibidas, mas os alunos sabiam que só teriam *feedback* do professor naquilo que fosse explicitamente postado no fórum. Os professores de cada disciplina visitam os fóruns dos grupos pelo menos uma vez por dia para responder as dúvidas ou prover alguma orientação.

Um dos diferenciais da proposta deste trabalho foi justamente promover o desenvolvimento do software, a partir de um ambiente virtual, em grupo uma vez que os alunos, muitas vezes, estarão submetidos a condições similares em sua vida

profissional: distância física entre os membros de uma equipe, diferentes horários de trabalho, uso de ferramentas alternativas de comunicação e divisão de tarefas.

A Tabela 1 exibe os entregáveis de cada uma das disciplinas trabalhadas.

Tabela 1. Entregáveis por Disciplina

Disciplina	*Entregáveis*
Modelagem de Aplicações Web	• Diagrama de contexto do sistema; • Diagrama de casos de uso do sistema; • Detalhamento de um dos casos de uso identificados; • Especificação dos requisitos da interface gráfica deste caso de uso; • Identificação das classes de domínio do sistema;
Projeto de Aplicações Web	• Diagrama de sequência do sistema para o cenário do fluxo principal do caso de uso escolhido anteriormente; • Diagrama de diagrama de interação detalhando uma operação mais complexa do fluxo; • Projeto do leiaute e da navegação das páginas de interface Web relativas apenas ao caso de uso escolhido; • Diagrama de classes de projeto • Implementação do caso de uso escolhido em um framework MVC selecionado pelo grupo;
Construção e implantação de Aplicações Web	• Plano de testes contendo casos de testes do caso de uso especificado na disciplina Modelagem de Aplicações Web; • Planilha de estimativa usando a técnica Análise de Pontos de Testes;

5. Resultados

O trabalho interdisciplinar foi executado da forma como descrita neste texto no segundo semestre de 2014. Ao final da realização das três disciplinas envolvidas, os 78 alunos foram convidados a responder a um questionário de avaliação do trabalho. Foram obtidas 30 respostas e este resultado bem como sua análise são apresentados na Tabela 2 e discutidos a seguir.

Pode-se observar que os alunos, em geral, reconheceram a contribuição do trabalho para o aprendizado e aplicação das técnicas na prática. 56,7% dos alunos consideraram alta ou muito alta a contribuição para o aprendizado enquanto que 50,0% dos respondentes entenderam como alta ou muito alta a contribuição para aplicação das técnicas na prática. No geral, pode-se ver também que não houve problemas com o enunciado e informações do trabalho e que a comunicação com os professores foi considerada como ao menos razoavelmente eficiente e suficiente por 86,7% dos respondentes e ainda 90,0% entendem que as respostas às suas dúvidas foram ao menos razoavelmente completas e claras.

Quando questionados sobre a contribuição para o desenvolvimento de habilidades de trabalho em equipe, a maioria, 63,3%, entendeu que o trabalho contribui pouco ou nada para este objetivo.

Tabela 2. Resultados do Questionário Aplicado

Questão / Alternativas	*Respostas*
Você considera que a contribuição do trabalho para seu aprendizado foi?	
Muito alta	3,3%
Alta	53,4%
Baixa	33,3%
Muito baixa	10,0%
Você considera que a contribuição do trabalho para que você possa aplicar as técnicas aprendidas na prática foi?	
Muito alta	0,0%
Alta	50,0%
Baixa	40,0%
Muito baixa	10,0%
Você considera que enunciados e informações para realização do trabalho foram?	
Bem detalhados	26,7%
Razoavelmente detalhados	60,0%
Pouco detalhados	10,0%
Nada detalhados	3,3%
A comunicação com os professores para elucidar dúvidas sobre o trabalho foi?	
Eficiente e suficiente	60,0%
Razoavelmente eficiente e suficiente	26,7%
Pouco eficiente e suficiente	13,3%
Nada eficiente e suficiente	0,0%
Você considera que as respostas às suas dúvidas foram?	
Completas e claras	50,0%
Razoavelmente completas e claras	40,0%
Pouco completas e claras	6,7%
Nada completas e claras	3,3%
Como você avalia a realização do trabalho em grupo?	
contribui muito para desenvolver habilidades de trabalho em equipe	10,0%
contribui razoavelmente para desenvolver habilidades de trabalho em equipe	26,7%
contribui pouco para desenvolver habilidades de trabalho em equipe	33,3%
contribui nada para desenvolver habilidades de trabalho em equipe	30,0%
Como você acredita que o trabalho deveria ser realizado?	
Em equipe, exatamente como foi neste curso	23,3%
Em equipe, porém com equipes maiores	6,7%
O aluno deveria escolher se deseja fazer em equipe ou individualmente	50,0%
Somente individualmente	20,0%
Você recomendaria fazer estas disciplinas com este trabalho como foi realizado este semestre a outro colega?	
Sim	63,3%
Não	36,7%

Durante o período de realização da disciplina, alguns grupos manifestaram sua opinião em relação à realização do trabalho em grupo. Alguns alunos entenderam que o trabalho em grupo não era adequado para uma disciplina virtual e outros ainda relataram dificuldades de comunicação com os integrantes do grupo. Por outro lado, houve grupos que relataram a realização do trabalho como muito proveitosa e bem sucedida. Frente a este dilema, os alunos foram questionados sobre a forma de realização do trabalho. Metade dos respondentes afirmou que entende que o aluno deveria poder se manifestar quanto a seu desejo de realizar o trabalho em grupo ou individualmente. 20% dos alunos afirmou preferir a realização individual somente; enquanto 23,3% se mostraram satisfeitos com a realização do trabalho em grupo.

Os alunos foram questionados quanto a sua dedicação ao trabalho. Em cada disciplina, 6,7% dos alunos relataram ter gastado até 5 horas enquanto 33,3% disseram ter investido até 10 horas no trabalho. Pode-se dizer que os alunos dedicaram uma carga horária expressiva, pois ainda 23,3% informaram ter gastado até 15 horas e 36,7% gastaram mais de 15 horas com sua participação. Ainda sobre a carga horária, 26,7% dos respondentes acharam esta carga elevada, enquanto a maioria, 56,7% entendem que ela foi apropriada e apenas 16,7% pensam que foi abaixo do esperado.

Mesmo relatando alguns problemas com a organização e comunicação entre o grupo, 63,3% dos alunos afirmaram que recomendariam cursar estas disciplinas com este trabalho para outro colega.

Por fim, o questionário apresentou duas perguntas abertas sobre a falta ou necessidade de ferramentas para a realização do trabalho e ainda se o aluno teria alguma sugestão de melhoria para o mesmo. Os alunos usaram estes espaços para fazer recomendações não somente ao trabalho, mas ao curso como um todo. Algumas observações mais frequentes e relevantes foram: o conhecimento heterogêneo entre os membros dos grupos, aulas muito teóricas e com poucos exemplos práticos, o desejo de escolher seus próprios grupos, dificuldades de horário comum para comunicação entre os membros, um alto grau de exigência do trabalho em relação ao conteúdo das aulas teóricas, baixa participação de alguns alunos, possibilidade de se estabelecer um líder para cada grupo e promoção de mais incentivo à participação.

5.1. Lições Aprendidas

Observando as discussões realizadas nos fóruns das disciplinas e a avaliação dos dados obtidos com o questionário apresentado anteriormente, foi possível identificar algumas lições aprendidas assim como ações a serem implantadas na próxima oferta das disciplinas:

- Deve ser incentivado o uso de ferramentas de comunicação mais eficientes entre os alunos, pois o fórum de discussão não se mostrou muito apropriado para a agilidade que o trabalho exige. De qualquer forma, os alunos devem ter em geral um maior comprometimento com as disciplinas e maior dedicação em horas ao longo dos dias de sua realização. Além disso, deve ser incentivada a interação também assíncrona entre os alunos;
- As vídeoaulas devem conter demonstrações de técnicas ou ainda do uso de ferramentas, além da apresentação de conceitos teóricos;

- Além do enunciado das atividades em grupo deve-se fornecer ao aluno um exemplo completo de um trabalho a ser entregue. Alguns alunos relataram entender o que estava sendo pedido, mas tinham dificuldade de concretizar em um produto entregável. O exemplo ajudaria os alunos a enxergarem o alvo final do trabalho e a não perderem tempo discutindo o que é esperado como resultado entregável.
- A realização dos trabalhos em grupo deve ser informada aos alunos no início do curso. Neste caso, os alunos compararam o trabalho destas três disciplinas com trabalhos individuais feitos em disciplinas anteriores;

6. Conclusões e Trabalhos Futuros

Este trabalho apresentou os desafios e dificuldades da condução de um trabalho realizado ao longo de três disciplinas do curso à distância de Desenvolvimento de Aplicações Web da PUC Minas que envolveu disciplinas do eixo da Engenharia de Software relacionadas a Requisitos de Software, Projeto de Sofware e Testes e Implantação de Software. É evidente que mesmo em um curso a distância, os alunos devem ser incentivados a trabalhar em equipe e esta capacidade deve fazer parte de sua avaliação. Entretanto, as diferenças entre um curso presencial e um virtual trazem novos desafios à realização de um trabalho que envolva desenvolvimento de software no contexto do ensino a distância.

De acordo com pesquisa realizada com os alunos, foi possível identificar que provavelmente frente a algumas dificuldades de comunicação e participação dos grupos, alguns alunos questionaram a realização de um trabalho em grupo em um curso virtual. Mesmo assim, mais da metade dos alunos entendeu que o trabalho traz contribuições para seu aprendizado e prática das técnicas estudadas e ainda que a comunicação com os professores e o volume de informações disponibilizadas sobre o trabalho foi, em geral, suficiente para sua realização. Além disso, a maioria afirmou que recomendaria o curso com este trabalho para outro colega.

Como trabalhos futuros, pretende-se realizar algumas ações identificadas como lições aprendidas, tais como gravação de vídeoaulas mais práticas, incentivos à comunicação e participação entre os grupos mesmo que de forma assíncrona e ainda esclarecimentos sobre o trabalho logo no início do curso e exemplos de produtos entregáveis.

7. Referências

Araújo, B. M. O.; Cunha, A. P. A (2014). O papel do professor como mediador nas interações em fóruns online: possibilidades de um aprender colaborativo. XI Congresso Brasileiro de Educação Superior à Distância (ESUD), 2014.

Costa, V. M. F.; Schaurich, A.; Stefanan. A.; Sales, E.; Richter, A (2014). Educação a distância x educação presencial: como os alunos percebem as diferentes características. XI Congresso Brasileiro de Educação Superior à Distância (ESUD), 2014.

Cunha, A. M.; Braga e Silva, G.; Monte-Mor, J. A.; Domiciano, M. A. P.; Vieira, R. G. (2008) Estudo de Caso Abrangendo o Ensino Interdisciplinar de engenharia de Software. Fórum de Educação em Engenharia de Software.

Cunha, M. X. C.; Souza Júnior, M.F. e Almeida, H.O. (2007) Análise dos Resultados da Aplicação de Projetos Interdisciplinares em um Curso de Tecnologia sob a Perspectiva dos Alunos. XV Workshop sobre Educação em Computação - WEI, Rio de Janeiro-RJ. Anais do XXVII Congresso da Sociedade Brasileira de Computação.

Fleury, A. L.; Abrahão, J. I.; Montedo, U. B.; Mascia, F. L.; Pessoa, M. S. P.; Gonçalves, R. F (2014). Uma experiência de Ensino de Estatística a Distância para um Curso de Engenharia. Revista de Ensino de Engenharia, v. 33, n. 1, pp. 37-47, 2014.

Fonseca, R. C.; Stockmanns, J. I.; Rutecki, L. K.; Lima, C. V (2014). A autonomia intelectual no exercício do aprender a aprender na educação à distância. XI Congresso Brasileiro de Educação Superior à Distância (ESUD), 2014.

Mattos, M. M.; Tavares, A. C (1999). Vxt: Experiência de desenvolvimento cooperativo de um ambiente didático. Congresso Iberoamericano de Educação Superior em Computação (VII CIESC), Assunção, Paraguai, 1999.

Messa, W. C. (2010). Utilização de Ambientes Virtuais de Aprendizagem - AVAS: A Busca por uma Aprendizagem Significativa. Revista Brasileira de Aprendizagem Aberta e a Distância, Vol 9, 2010

Prikladnicki, R., Albuquerque, A. B., Wangenheim, C. G., Cabral, R., Ensino de Engenharia de Software: Desafios, Estratégias de Ensino e Lições Aprendidas, FEES 2009, Fórum de Educação em Engenharia de Software, 2009.

dos Santos, R. P., dos Santos, P. S. M., Werner, C. M. L., & Travassos, G. H. (2008). Utilizando Experimentação para Apoiar a Pesquisa em Educação em Engenharia de Software no Brasil. Fórum de Educação em Engenharia de Software (FEES).

Soares, M. S., Uma Experiência de Ensino de Engenharia de Software Orientada a Trabalhos Práticos, XXIV Congresso do SBC, XII WEI - Workshop de Educação em Computação, 2004.

Pinto, C. L. Q.; Rocha, C. R. C.; Vilarim, G. (2010). Desafios da Prática da Interdisciplinaridade em Cursos de Ciência da Computação: a Experiência da UNIFESO. XVIII Workshop sobre Educação em Computação - WEI, Belo Horizonte - MG. Anais do XXX Congresso da Sociedade Brasileira de Computação.

Trindade, C. C.; Moraes, A. K. O.; Meira, S. L. (2008). Comunicação em Equipes Distribuídas de Desenvolvimento de Software: Revisão Sistemática. 5th Experimental Software Engineering (ESELAW). Salvador.

Warden, C. A.; Stanworth, J. O.; Ren, J. B.; Warden, A. R. (2013). Synchronous learning best practices: An action research study. Computers & Education, v.63, pp. 197-207.

A METODOLOGIA SCRUM COMO MOBILIZADORA DA PRÁTICA PEDAGÓGICA: UM OLHAR SOBRE A ENGENHARIA DE SOFTWARE

Fabio Gomes Rocha[1], Rosimeri Ferraz Sabino[2], Ronald Henrique Leal Acipreste[3]

[1]Departamento de Computação – Universidade Federal deSergipe (UFS) e Universidade Tiradentes (Unit)
Sergipe, SE - Brazil.

[2]Departamento de educação– Universidade Federal de Sergipe (UFS)
Sergipe, SE - Brazil.

[3]Senac – Brasilia, DF – Brazil

fabio_gomes@unit.br, rf.sabino@gmail.com, ronald.acipreste@gmail.com

Abstract. *The study, descriptive and exploratory, deals about the Scrum methodology as an alternative to pedagogical practices , presenting research with students of software engineering discipline of a technical course in computer science in technical school of the brazilian northeast. It sought to analyze student performance on the practices with Scrum and verify the perception of teaching using this methodology, knowing the positive and negative aspects attributed to the learning experience. The conclusions indicate the Scrum methodology as potentially suitable for the promotion of learning, besides adding to the student satisfaction.*

KeyWords:*Scrum. Software Engineering. learning.*

Resumo. *O trabalho, de cunho descritivo e exploratório, trata sobre a metodologia Scrum como alternativa para práticas pedagógica, apresentando investigação com alunos da disciplina de engenharia de software de um curso técnico em informática, em escola profissionalizante da região nordeste brasileira. Buscou-se analisar o desempenho dos estudantes diante das práticas com o Scrum, bem como verificar a percepção sobre o ensino com essa metodologia, conhecendo os aspectos positivos e negativos atribuídos à experiência pedagógica. As conclusões indicam a metodologia* Scrum como *potencialmente adequada à promoção da aprendizagem, além de agregar satisfação ao aluno.*

Palavras-chave*: Scrum. Engenharia de Software. Ensino.*

1. Introdução

Na medida em que o mundo atual se vê em uma interação global, promovida pelas tecnologias cada vez mais sofisticadas e disponíveis, também demonstra, em todos os âmbitos das práticas sociais, que os comportamentos, as expectativas e a própria geração

de conhecimento desenvolvem-se sob novas dinâmicas. Nesse contexto, a educação enfrenta o constante desafio de buscar meios de estímulos à aprendizagem.

Em um cenário esboçado na cultura digital, com facilidade de acesso às informações e redes de comunicação instantânea, é necessário atentar para uma superficialidade do saber, onde informação é confundida com conhecimento, desprezando uma aprendizagem significativa (MASINI, MOREIRA, 1982; NOVAK, 2010). Essa última é obtida na preparação do indivíduo não só para uma situação específica de sua vida, mas, também, para o pensamento crítico e a busca de soluções em contextos diversos. Isso se revela já no conceito inicial de Ausubel (1963, 1968) sobre aprendizagem significativa, com a definição de um processo pelo qual um conhecimento se relaciona, de forma não-arbitrária e substantiva, à estrutura cognitiva do aprendiz.

Assim, um conhecimento agrega-se a anteriores, possibilitando uma interação para a construção de novos conhecimentos. Para isto, no entanto, há de existir uma disposição do aprendiz em relacionar o que aprende a um conhecimento prévio, e não apenas memorizá-lo de forma mecânica. Tal comportamento desejado encontra amparo no entendimento de Novak (1984) de que um processo de aprendizagem resulta-se construtivo quando há uma experiência afetiva positiva. Isso remete às relações durante o ensino e a aprendizagem, envolvendo emoções e sentimentos entre aluno e professor.

Disso se depreende a necessidade de propostas pedagógicas que se alinhem aos aspectos de colaboração, compartilhamento, autonomia e experimentação. Com a orientação do professor, os alunos podem experimentar situações em que são chamados à participação efetiva, com a articulação de ideias entre os colegas, possibilitando o desenvolvimento de processos mentais, habilidades cognitivas e de relacionamento social.

Tais pressupostos estão presentes na metodologia de aprendizagem colaborativa, calcada na teoria construtivista de Piaget (1982) e perspectiva sociocultural de Vigotsky (1998). Embora se encontre os termos aprendizagem colaborativa como similar à aprendizagem cooperativa, a adoção da primeira demonstra-se mais adequada à própria abrangência do processo educacional. Em estudos sobre essas expressões, Panitz (1996) concluiu que a colaboração ampara-se no envolvimento pessoal, enquanto que a cooperação é projetada para a realização ou entrega de um produto. Em seu conceito, a aprendizagem colaborativa é [...] uma filosofia de vida, e não apenas uma técnica para aula. Em todas as situações onde as pessoas se reúnem em grupos, sugere uma maneira de lidar com as pessoas que respeita e destaca no grupo as habilidades e contribuições individuais dos seus membros (PANITZ, 1996, p. 3) (tradução nossa). Nessa aprendizagem há o compartilhamento de autoridade e responsabilidades entre os membros, diferenciando-se de uma competição em que se destacam os melhores do grupo.

A perspectiva da aprendizagem colaborativa corrobora, portanto, metodologias de aulas que visem um processo de ensino-aprendizagem centrado no aluno, com dinâmicas, ambientes e contextos que proporcionem a interação e criatividade. Sob essa perspectiva, a metodologia Scrum apresenta-se como alternativa para práticas

pedagógicas. O Scrum, criado em 1993 por Ken Schwaber e Jeff Sutherland, tem a origem de seu nome no "jogo de rúgbi e se refere à maneira como um time trabalha junto para avançar com a bola no campo. Alinhamento cuidado, unidade de propósito, clareza de objetivo, tudo se unindo" (SUTHERLAND, 2014, p. 16).

Essa metodologia é um arcabouço de recursos e modelos para gerenciamento de projetos e, conforme os seus idealizadores, "[...] não é um processo ou uma técnica para construir produtos; em vez disso, é um framework dentro do qual você pode empregar vários processos ou técnicas" (SCHWABER, SUTHERLAND, 2013, p. 3), visando melhorar o desempenho das pessoas no trabalho em conjunto em qualquer empreendimento. A aplicação do Scrum na educação vem sendo alvo de estudos que avaliam essa metodologia em cursos de campos e níveis diversos, como no ensino superior e médio, nos campos de gestão ou computação (PERSSON et al, 2011; WANGENHEIM, BORGATTO, 2013; CUBRIC, 2013), indicando uma aprendizagem significativa que prepara o aluno para outras situações em sua vida. A intervenção educativa ocorre com a incorporação de novos conteúdos aos conhecimentos prévios do aluno, distanciando-se de uma aprendizagem mecânica (MASINI, MOREIRA, 1982; NOVAK, 2010).

A partir do entendimento sobre esses aspectos da aprendizagem e do modelo Scrum, iniciou-se a investigação tratada neste trabalho, com alunos da disciplina de engenharia de software de um curso técnico em informática, em uma escola profissionalizante da região nordeste brasileira. A escolha da disciplina deve-se ao fato dela ter em seu escopo o estudo dos processos de desenvolvimento de programas (PRESSMAN, 2007; SOMMERVILLE, 2011) possibilitando o uso do Scrum na aplicação das técnicas de elaboração de sistemas em um curto espaço de tempo e de forma colaborativa. Considerou-se, ainda, que a aprendizagem em engenharia de software vem recebendo estudos, no contexto brasileiro e internacional (COSTA, SANTOS, WERNER, 2010; SHAW, 2000), que apontam a pouca frequência de atividades práticas como fator de comprometimento das habilidades dos alunos na aplicação dos conhecimentos trabalhados em sala em cenário real.

Assim, definiu-se como objetivo da investigação a análise sobre o desempenho dos estudantes na disciplina de engenharia de software, com práticas pedagógicas a partir do uso do Scrum. Buscou-se, também, verificar a percepção do estudante sobre o ensino com essa metodologia, conhecendo os aspectos positivos e negativos atribuídos à experiência pedagógica. Embora se tratando do exame sobre um cenário e grupo específicos, entende-se que a observação pode agregar às discussões tanto sobre o viés das práticas pedagógicas em formações técnicas como sobre a própria engenharia de software. Mesmo analisada como uma disciplina, componente de um currículo para técnicos em informática, os alunos que a estudam podem despertar interesse em prosseguir formação específica nesse campo da engenharia. Dessa forma, investigações sobre sucessos ou dificuldades na aprendizagem ainda em nível de formação técnica podem, eventualmente, subsidiar o ensino em âmbitos subsequentes.

2. PROCEDIMENTOS METODOLÓGICOS

Como um estudo de caso empírico, tratando de uma população específica, a investigação caracteriza-se como descritiva, no tocante ao relato da aplicação da metodologia Scrum, e exploratória, sobre a verificação de adaptabilidade dos alunos e implicação no processo de ensino e da aprendizagem (YIN, 2014, 2012; SEVERINO, 2013). Na busca sobre como os investigados demonstravam e percebiam o seu desempenho nos exercícios da disciplina engenharia de software, se entendeu também possível identificar os motivos pelos quais eventuais dificuldades poderiam ocorrer. Os resultados da prática pedagógica foram analisados qualitativamente diante das bases conceituais adotadas.

A investigação envolveu uma população de duzentos e cinquenta e cinco alunos, de um curso técnico em informática em escola profissionalizante da região nordeste brasileira. Esses estudantes cursavam o ensino médio em escola pública concomitante ao ensino profissionalizante, a partir do penúltimo ano do ensino médio. Tal população, cuja média de idade é de dezoito anos, foi distribuída aleatoriamente pelo sistema de matrículas em cinco turmas de trinta alunos e três de trinta e cinco alunos.

Partindo-se da escolha do componente curricular "engenharia de software", definiu-se o tamanho mínimo da amostra da população em setenta participantes, diante do nível de confiança desejada (BRACARENSE, 2012; DOWNING, CLARK, 2011). Para tanto, procedeu-se o sorteio de duas turmas para a aplicação do Scrum no ensino da disciplina, obtendo-se a amostra de duas turmas de trinta e cinco alunos cada uma, totalizando setenta participantes da pesquisa, sendo 34 mulheres e 36 homens. O trabalho com essa amostra permitiu uma confiabilidade estatística de representação sobre a população total de 90%, com uma margem de erro de 10% para mais ou para menos. A margem de erro é o percentual máximo de erro de estimativa, ou seja, a diferença máxima provável entre a medida estimada na amostra e o valor total da população (CAMPOS, WODEWOTZKI, JACOBINI, 2011).

A disciplina de engenharia de software, desenvolvida e observada neste estudo no ano de 2014, possui uma carga horária de sessenta horas, ministrada em aulas de quatro horas, durante três semanas. O objetivo da disciplina é promover a compreensão sobre os processos e métodos de construção de sistemas computacionais, com foco na garantia de qualidade do produto, bem como desenvolver as habilidades para a aplicação dos saberes trabalhos. Diante disto, o conteúdo foi distribuído em quatro horas para explanação sobre a metodologia Scrum; quatro horas para explicação sobre o projeto a ser desenvolvido pelos alunos, e cinquenta e duas horas de uso do Scrum na prática pedagógica.

Inicialmente, sem explanações ou explicações sobre o Scrum, aplicou-se um questionário junto aos alunos, com três questões fechadas, na intenção de averiguar sobre eventual familiaridade do grupo com a metodologia pretendida para a prática pedagógica. Os resultados obtidos apontam que nenhum dos alunos possuía conhecimento prévio sobre o Scrum, sendo que 67% conheciam outras metodologias ágeis, como XP, FDD e DSDM, e 58% já haviam desenvolvido programas em grupo. A seguir, foi feita a explanação sobre o Scrum, abordando o seu funcionamento e como os alunos deveriam trabalhar com essa metodologia, definindo-se o seguinte escopo:

a) uma reunião inicial com o solicitante do produto;

b) uma reunião entre a equipe para o planejamento dos itens a serem produzidos durante o período de trabalho, denominado sprint. Esse período foi estabelecido na prática pedagógica como de três dias. Os alunos deveriam atribuir pontos para o grau de dificuldade no desenvolvimento de cada item, sendo 1 para o mais fácil e cinco para o mais difícil;

c) durante a sprint deveria ocorrer uma reunião diária com duração de dez minutos, com todos os participantes em pé. Após, a equipe deveria iniciar/dar andamento ao desenvolvimento dos itens propostos da reunião do item "b";

d) ao final dos três dias da sprint, deveria ocorrer uma reunião de retrospectiva, com o tempo definido de trinta minutos. Ao término desta reunião, deveria se retornar ao item "b" para a continuidade de planejamento dos próximos itens. Este ciclo deveria se repetir até a conclusão do projeto.

Em aula seguinte, foi apresentado o projeto que seria o produto da disciplina, explicando-se os papéis assumidos pelo professor e aluno. O primeiro atuaria como um "cliente" de aquisição de um software, denominado no Scrum como product owner . A ele caberia fornecer as características do produto que necessitava. Para o atendimento desse cliente, todos os alunos atuariam como uma única equipe de desenvolvimento, sendo necessário para isto a auto-organizarão do grupo para a busca de informações necessárias ao trabalho. A equipe deveria escolher um dos estudantes para o papel de "scrum master" , o qual recebeu explicações adicionais sobre as tarefas de sua responsabilidade e características sobre a sua postura: liderança da equipe, promoção da comunicação do grupo, postura facilitadora no sentido de suprir as necessidades da equipe para o desenvolvimento do trabalho, e acompanhamento para o cumprimento das regras definidas e dos prazos estabelecidos.

O acompanhamento pedagógico feito pelo professor em todas as aulas seguintes ocorreu no próprio papel de "cliente" do projeto, uma vez que o Scrum não só permite como incentiva a participação do solicitante do produto em todos os momentos do seu desenvolvimento. Dessa forma, as intervenções do professor estavam previstas para a orientação nas dúvidas dos alunos em relação ao produto e não sobre como fazê-lo. Os conhecimentos base para a construção de programas foram alvo de disciplinas anteriores, sendo que na engenharia software tais conhecimentos surgem sistematizados. Para o caso de necessárias intervenções sobre como fazer, os estudantes deveriam se dirigir ao scrum master, o qual solicitaria ao professor uma "visita técnica" à equipe.

Ao final da disciplina, os estudantes apresentaram o software criado, em funcionamento real. Esta atividade correspondeu a 70% da avaliação sobre a aprendizagem, associando-se à observação sobre o desempenho individual na equipe, no tocante ao cumprimento das regras do Scrum, bem como à colaboração com os colegas e responsabilidade sobre as tarefas. Para a avaliação sobre esses aspectos que constituíram 30% da aprendizagem foi efetuado o registro diário individual sobre os alunos. Também nessa última aula da disciplina, os alunos responderam a um questionário, composto por nove questões fechadas e uma aberta, com vistas a se obter a visão dos estudantes sobre a compreensão da metodologia Scrum, bem como a contribuição para o seu desempenho e motivação para as atividades. Buscou-se, ainda, verificar se a oportunidade de auto-organização auxiliou no desenvolvimento do projeto; se a responsabilidade sobre o resultado apresentou-se como positiva à aprendizagem; e

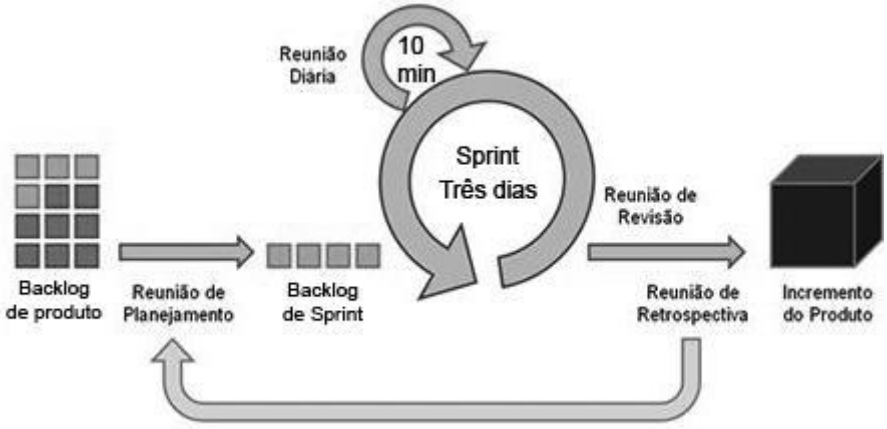
se a forma de organização do trabalho, a partir de reuniões diárias e identificação dos itens por grau de dificuldade, auxiliou na resolução dos problemas e administração do tempo.

3. A PRÁTICA PEDAGÓGICA: RESULTADOS E ANÁLISE

A aproximação entre a escola e a vida profissional do indivíduo constou do Relatório Delors (1996) indicando-a como necessária às propostas educacionais para o século XXI, tempo também do desafio de uma "educação ao longo da vida [...] como chave de acesso" a este século (DELORS, 2010, p. 13). Do documento, advém os quatro pilares para a educação: a) aprender a aprender, indicando a cultura geral como base de aprendizado para a vida; b) aprender a fazer, indicando o aprendizado de competência não só para uma profissão, mas, também, para situações não previstas; c) aprender a conviver, relacionado |à realização de projetos comuns, ao gerenciamento de conflitos e respeito à pluralidade de valores; e d) aprender a ser, relacionado à autonomia, discernimento e responsabilidade perante o coletivo (DELORS, 2010, p. 31).

Observa-se, portanto, que as práticas pedagógicas que visem atender tais recomendações devem incluir atividades que possibilitem a convergência entre os citados saberes. No caso da metodologia Scrum, encontra-se os princípios de autonomia, convivência em grupo, busca de solução e compartilhamento de conhecimento, caracterizando-se como um modelo em que indivíduo e interação são mais relevantes que processos e ferramentas (COHN, 2011; SUTHERLAND, 2014). Para esta investigação, o ciclo de vida original da metodologia aplicada foi adaptado para a realidade e o tempo disponível ao ensino da disciplina de engenharia de software, conforme a figura a seguir:

Figura 1 – Ciclo de vida do Scrum adaptado à investigação



Fonte: Elaborado pelos autores, com base em Cohn (2011).

A identificação sobre o desconhecimento de todos os alunos sobre a aplicação do Scrum exigiu um acompanhamento e orientação de forma mais efetiva nos três primeiros dias da atividade, registrando-se no diário individual o comportamento para um projeto coletivo e a adaptação ao modelo de trabalho. Sobre isso, observou-se uma

rápida assimilação e disposição para colaboração por parte de todos os alunos. A comunicação necessitou ser promovida pelo aluno que recebeu o papel de scrum master apenas durante a primeira reunião. Após, o grupo compreendeu a importância da comunicação fluir independente de uma liderança.

A partir do quarto dia, início do período configurado como segunda sprint, o professor, sob o papel de cliente, passou a intervir apenas quando procurado pelo aluno scrum master para informação de detalhes do produto solicitado. Também no quarto dia foi realizada a reunião de retrospectiva de sprint. Observou-se, então, que o tempo de trinta minutos, estipulado inicialmente para essa atividade, não seria suficiente, sendo alterado pelos próprios alunos para sessenta minutos. Para tal decisão, o grupo consultou o professor, obtendo a anuência desde que todas as demais reuniões de retrospectiva tivessem a mesma carga horária. A permissão para essa adaptação deveu-se à identificação da necessidade de tempo para a participação de todos os alunos.

Concluída tal reunião, iniciou-se a etapa de planejamento da sprint seguinte, cujo tempo não poderia exceder também a sessenta minutos. Durante tal planejamento, o grupo identificou os itens a serem implementados, verificando que o desenvolvimento de novos exigia melhorias em alguns já concluídos. Com isso, o grupo reduziu a quantidade de itens a serem desenvolvidos em relação ao primeiro clico de trabalho. Nesse ponto, revelou-se uma preocupação inicial dos estudantes em criar muitos itens sem considerar a qualidade. Na oportunidade da segunda reunião de sprint constatou-se um ganho de maturidade na avaliação do contexto que envolvia o produto, priorizando a qualidade e analisando as consequências para o resultado final.

Ao término da segunda sprint, a reunião de retrospectiva apontou que não havia necessidade de novos ajustes no processo de desenvolvimento, prosseguindo assim até o final da terceira sprint, com a conclusão do produto. Durante esse período, observou-se que na segunda sprint ocorreu a busca do equilíbrio entre a produtividade da equipe e a qualidade do produto. Já na terceira sprint os alunos demonstraram domínio sobre a metodologia Scrum, e elevada produtividade individual e em grupo, sem perda da qualidade nos resultados. Isto foi comprovado na apresentação do produto solicitado, em estágio de funcionamento real.

Os registros diários individuais apontaram um crescente comportamento colaborativo e a ampliação da comunicação. Também se identificou que os alunos se empenharam para o cumprimento das tarefas sob a sua responsabilidade, buscando um desempenho que, assegurasse agregar resultados ao grupo. Ocorreram situações cotidianas de compartilhamento de conhecimento e disposição para ajuda ao coletivo, indicando o compromisso e a responsabilidade individual perante o grupo. Nesse sentido, a prática pedagógica alinha-se às abordagens do construtivismo (PIAGET, 1982) e sociointeracionismo (VIGOTSKY, 1998) sobre a elaboração do conhecimento pela experiência e interação.

A proposta de um processo de ensino-aprendizagem que aproxima o aluno de situações de vida permite a mobilização de saberes, resultando em uma aprendizagem significativa para o indivíduo (PERRENOUD, 2002). Não obstante aos debates sobre uma educação utilitarista, é mister que a educação inclua em suas finalidades a preparação do indivíduo para os relacionamentos e situações que lhe serão impostas nas intervenções no mundo, pois a "sua missão consiste em permitir que todos, sem

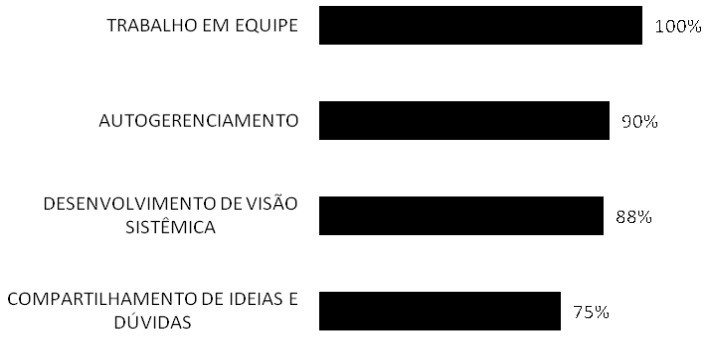
exceção, façam frutificar seus talentos e suas potencialidades criativas, o que implica, por parte de cada um, a capacidade de assumir sua própria responsabilidade e de realizar seu projeto pessoal" (DELORS, 2010, p. 10).

Para a averiguação sobre os aspectos que se pretendeu envolver na prática pedagógica investigada, buscou-se conhecer a percepção dos alunos sobre a metodologia. Por meio de questionário aplicado na última aula, a totalidade dos alunos apontou que obteve a compreensão sobre a metodologia Scrum, considerando-a uma prática motivadora para as atividades na disciplina. Os alunos também indicaram que a metodologia ampliou o desempenho como estudante, no tocante ao grau de interesse nas aulas em relação a todas as demais disciplinas do curso. A necessidade de auto-organização para o cumprimento das tarefas foi considerada um fator positivo por 67% dos alunos, contribuindo para o desenvolvimento das atividades.

A responsabilidade sobre o aprendizado foi apontada por 58% dos alunos como aspecto incentivado pela metodologia Scrum. O trabalho em grupo, com reuniões diárias e a possibilidade de selecionar as tarefas a serem desenvolvidas diante do grau de dificuldade atribuído pelo grupo, foram apontados pela totalidade dos alunos como fatores positivos para a aprendizagem. Isso não se limitou à disciplina, mas despertou um comportamento voltado para a busca de soluções e organização do próprio tempo em qualquer atividade.

Em penúltimo questionamento, de forma aberta, as respostas receberam o tratamento quantitativo sobre termos identificados como representativos ao estudo. Na questão, os alunos poderiam destacar aspectos sobre eventual satisfação ou insatisfação no uso da metodologia Scrum para a sua aprendizagem. Os resultados em relação à satisfação são expostos no gráfico a seguir:

Figura 2 – Aspectos que geraram satisfação nos alunos sobre a utilização da metodologia Scrum



Fonte: Elaborado pelos autores.

Já sobre eventual insatisfação, os alunos foram unânimes em afirmar que não encontrarem pontos negativos na aplicação da metodologia. Por fim, em questão fechada, questionou-se os alunos sobre a atuação do professor como orientador, em seu papel de "cliente" diante da metodologia Scrum. A totalidade dos alunos afirmou que a

ação docente sob essa metodologia constituiu uma prática de ensino positiva, contribuinte de maior aprendizagem.

4. CONSIDERAÇÕES FINAIS

A adoção da metodologia Scrum para o ensino na disciplina de engenharia de software, em nível profissionalizante, demonstrou-se potencialmente adequada à promoção da aprendizagem. Os resultados da investigação constatam o alcance dos objetivos da disciplina, a satisfação do aluno sobre a metodologia para a realização das tarefas solicitadas, bem como sobre o modelo de trabalho em equipe. A prática pedagógica oportunizou, ainda, o desenvolvimento de comportamentos voltados para o respeito à diversidade de opiniões, ao compartilhamento de ideias e à ajuda mutua, implicando no aprendizado para a convivência. Agrega-se a isto a percepção positiva dos alunos sobre a engenharia de software, o que poderia favorecer a disposição aos estudos de disciplinas correlatas em futura carreira.

Observa-se, assim, que o Scrum, como metodologia para a prática pedagógica, alinha-se aos pilares da educação para a contemporaneidade, no tocante à contribuição para o despertar do interesse no aprender, colocando o aluno diante da concretude do resultado para a sua vida. Isso se evidenciou neste estudo no comportamento receptivo dos alunos para o desenvolvimento das atividades. Uma vez compreendido o formato das ações solicitadas, os alunos demonstraram um empenho natural para a superação de obstáculos e conclusão das tarefas. As dificuldades encontradas pelo professor para a prática pedagógica residiram apenas em ajustes de prazos, os quais foram feitos em comum acordo com a turma, promovendo uma dinâmica participativa.

Entende-se que as limitações deste estudo a uma disciplina e grupos específicos não permitem a generalização dos resultados a outros campos, porém a própria metodologia Scrum não se restringe a tecnologias, podendo ser aplicada nas mais diversas áreas. No tocante à aprendizagem, a investigação pode subsidiar práticas pedagógicas que, mesmo com adoção de outras metodologias, priorizem uma aprendizagem significativa ao aluno.

Pode-se concluir que a metodologia Scrum constitui uma ferramenta para o ensino que desperta o protagonismo no aluno sobre a própria aprendizagem, ao tempo que o prepara como membro de uma coletividade. O aprender, portanto, torna-se uma ação colaborativa, onde a assimilação individual repercute e contribui no sucesso dos demais, viabilizando o crescimento de todos os envolvidos.

REFERÊNCIAS

AUSUBEL, David P. The psychology of meaningful verbal learning. New York: Grune and Stratton, 1963.

AUSUBEL, David P. Educational psychology: a cognitive view. New York: Holt, Rinehart and Winston, 1968.

BRACARENSE, Paulo Afonso. Estatística aplicada às ciências sociais. Curitiba: IESDE, 2012.

CAMPOS, Celso Ribeiro; WODEWOTZKI, Maria Lúcia Lorenzetti; JACOBINI, Otávio Roberto. Educação Estatística: teoria e prática em ambientes remodelagem matemática. Belo Horizonte: Autêntica, 2011.

COHN, Mike. Desenvolvimento de software com Scrum: aplicando métodos ágeis com sucesso. Tradução: Aldir José Coelho Corrêa da Silva. Porto Alegre: Bookman, 2011.

COSTA, Heitor Augustus Xavier; SANTOS, Rodrigo Pereira dos.; WERNER, Cláudia Maria Lima. Uma análise do processo de ensino e aprendizagem de engenharia de software: desafios e soluções no contexto brasileiro. In: XI International Conference on Engineering and Technology Education INTERTECH'2010, 2010, Ilhéus - BA. Proceedings International Conference on Engineering and Technology Education, 2010. v. 1. p. 367-371.

CUBRIC, Marija. An agile method for teaching agile in business schools. The International Journal of Management Education. Hampshire, n. 11, p. 110-131, 2013.

DELORS, Jacques. Learning: the treasure within - report to UNESCO of the International

Commission on Education for the twenty-first century. Paris: UNESCO, 1996.

_____. Educação: um tesouro a descobrir - relatório para a UNESCO da Comissão Internacional sobre educação para o século XXI. Disponível em <http://unesdoc.unesco.org/images/0010/001095/109590por.pdf>. Acesso em: 12 dez. 2010.

DOWNING, Douglas; CLARK, Jeffrey. Estatística aplicada. Tradução: Alfredo Alves de Farias. 3. ed. São Paulo: Saraiva, 2011.

MASINI, Elcie F. Salzano; MOREIRA, Marco Antonio. A aprendizagem significativa: a teoria de David Ausubel. 3. ed. São Paulo: Moraes, 1982.

NOVAK, Joseph Donald; GOWIN, D. Bob. Learning how to learn. Cambridge (UK): Cambridge University Press, 1984.

_____. Learning, creating and using knowledge: concept maps as facilitative tools in schools and corporations. 2. ed. New York: Taylor & Francis Group, 2010.

PANITZ, Theodore. Collaborative versus cooperative learning: a comparison of the two concepts. Publicado em dez. 1996. Disponível em: < http://eric.ed.gov/?id=ED448443> Acessado em 30 out. 2012.

PERRENOUD, Philipe. A prática reflexiva no ofício do professor: profissionalização e razão pedagócia. Porto Alegre: Artmed, 2002.

PRESSMAN, Roger. S. Software engineering: a practitioner´s approach. 6. ed. New York: MacGraw Hill, 2007.

PERSSON, Mia. et at. On the use of scrum in project driven higher education (2011) Disponível em: < http://worldcomp-proceedings.com/proc/p2011/FEC3416.pdf> Acesso em: 10 jan 2013.

PIAGET, Jean. O nascimento da inteligência na criança. Rio de Janeiro: Zahar, 1982.

SHAW, Mary. Software Engineering Education: A Roadmap. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 22., 2000, New York. Proceeding Internacional Conference on Software Engineering. New York, 2000, p. 371-380.

SCHWABER, Ken; SUTHERLAND, Jeff. Um guia definitivo para o Scrum: as regras do jogo. Disponível em: < http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf> Acesso em: 04 abr. 2013.

SEVERINO, Antônio Joaquim. Metodologia do trabalho científico. 23. ed. São Paulo: Cortez, 2013.

SHEHABUDDEEN, Noordin. et al. Representing and approaching complex management issues: role and definition. Cambridge: Institute for Manufacturing of University of Cambridge, 2000.

SOMMERVILLE, Ian. Engenharia de software. Tradução: Selma Shin Shinizu Melnikoff, Reginaldo Arakaki, Edilson de Andrade Barbosa. 8. ed. São Paulo: Prentice Hall, 2011.

SUTHERLAND, Jeff. Srum: a arte de fazer o dobro do trabalho na metade do tempo. Tradução: Natalie Gerhardt. São Paulo: Leya, 2014.

VYGOTSKI, Lev Semenovitch. A formação social da mente: o desenvolvimento dos processos psicológicos superiores. 6. ed. São Paulo, Martins Fontes, 1998.

WANGENHEIM, Christiane Gresse von; SAVI, Rafael; BORGATTO, Adriano Ferreti. Scrumia: an educational game for teaching SCRUM in computing courses. The Journal of Systems and Software, Amsterdã, n. 86 p, 2675–2687, 2013.

YIN, Robert K. Applications of case study research. 3. ed. Thousand Oaks: Sage Publications, 2012.

_____ Estudo de caso: planejamento e métodos. Tradução Cristhian Matheus Herrera. 5. ed. Porto Alegre: Bookman, 2014.

Análise da Relevância dos Tópicos e da Efetividade das Abordagens para o Ensino de Engenharia de Software

Carlos S. Portela[1], Alexandre M. L. Vasconcelos[1], Sandro R. B. Oliveira[1,2]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 – 50.740-560 – Recife – PE – Brasil

[2]Programa de Pós-graduação em Ciência da Computação (PPGCC) – Instituto de Ciências Exatas e Naturais (ICEN) – Universidade Federal do Pará (UFPA)
Belém – PA – Brasil

{csp3, amlv}@cin.ufpe.br, srbo@ufpa.br

Abstract. *This paper presents the results of a survey on the adoption and learning of Software Engineering (SE) topics recommended by ACM/IEEE and Brazilian Society of Computer (SBC) curriculum guidelines. In addition, we researched which teaching approaches are adopted by professors and what these the students consider effective for their learning. This survey was conducted with 47 students and 23 professors of Computer undergraduate courses in 20 public and private education institutions in Brazil. The data analysis of this survey suggests that 6 SE knowledge units are considered most relevant, while other 4 units could be considered less relevant in the SE discipline curricula. Moreover, we identified that the students consider practices teaching approaches more effective to their learning.*

Resumo. *Este artigo apresenta os resultados de um survey sobre a adoção e aprendizagem de tópicos de Engenharia de Software (ES) recomendados pelos currículos de referência da ACM/IEEE e SBC. Pesquisou-se, também, sobre quais abordagens de ensino são adotadas pelos professores e quais destas os alunos consideram efetivas para seu aprendizado. Este survey foi realizado com 47 alunos e 23 professores de cursos de graduação em Computação de 20 instituições de ensino públicas e privadas do Brasil. A análise dos dados do survey sugere que 6 unidades de conhecimento da ES são consideradas de maior relevância, enquanto outras 4 unidades poderiam ser consideradas menos relevantes nas ementas da disciplina de ES. Além disso, identificou-se que os alunos consideram abordagens de ensino práticas mais efetivas para o seu aprendizado.*

1. Introdução

A Engenharia de Software (ES) se constitui como uma das disciplinas de maior relevância nos cursos da área de Computação [ACM/IEEE, 2013]. Isto decorre tanto da importância do software em si quanto dos desafios relacionados com a formação completa de um profissional que irá atuar no mercado, resultando em uma demanda crescente por profissionais bem qualificados [Duley et al., 2003]. Tipicamente, estes profissionais de software são formados em cursos de graduação como modo de preparação para atuar na indústria [Nunes, Reis e Reis, 2010].

No entanto, a indústria de software se queixa de que os cursos de graduação não ensinam aos alunos as competências necessárias para que eles possam começar a executar o seu trabalho com eficiência [Wangenheim e Silva, 2009]. Desta forma, as empresas de software têm que complementar os conhecimentos dos recém-formados com treinamentos e prover habilidades relacionadas ao processo de desenvolvimento de software [Bessa, Cunha e Furtado, 2012].

Esta carência na formação de profissionais graduados na área de ES pode ser resultado de uma educação inadequada [Lethbridge et al., 2007]. Especialmente nos cursos de graduação, os tópicos de ES são normalmente ensinados de forma bastante superficial [Wangenheim e Silva, 2009]. Lethbridge (2000) realizou uma pesquisa com profissionais da área de software, constatando que alguns tópicos da graduação foram considerados menos úteis, enquanto se teve a impressão que para outros tópicos considerados importantes, não foi dada a devida atenção no ensino. Este estudo foi repetido por Wangenheim e Silva (2009) que reforçaram as críticas de que as competências de ES, necessárias aos profissionais de software, muitas vezes não estão sendo adequadamente abordadas nos cursos de Computação.

Um dos principais problemas apontados por estes estudos foi a dificuldade de cobrir todos os tópicos no tempo disponível durante a graduação. Diante deste contexto, esta pesquisa pretende analisar a relevância dos tópicos de ES constantes nos currículos de referências da ACM/IEEE (2013) e SBC (2005). Além disto, esta pesquisa objetiva obter dados quantitativos do ensino/aprendizagem de ES a partir da identificação das abordagens de ensino aplicadas nesta disciplina. Desta forma, poder-se-á identificar os tópicos mais relevantes e as abordagens mais adequadas para ministrá-los.

Diferentemente dos trabalhos relacionados, esta pesquisa considera as versões mais atuais dos currículos da ACM/IEEE e SBC (2013 e 2005, respectivamente). Desta forma, contemplam-se 83 tópicos e 10 unidades de conhecimento, ao invés de apenas 69 tópicos e 6 unidades de conhecimento das versões anteriores destes currículos. Além disto, esta pesquisa abrange participantes de todas as cinco regiões do Brasil.

Além desta seção introdutória, a Seção 2 apresenta o planejamento e aplicação do *survey*, além dos dados demográficos sobre os participantes. Os resultados da coleta de dados são descritos na Seção 3. As análises e discussões dos resultados do *survey* são relatadas na Seção 4. Por fim, as limitações e ameaças à validade, as conclusões e as próximas etapas deste trabalho são apresentadas na Seção 5.

2. Planejamento e Aplicação do *Survey*

Survey é um método de pesquisa abrangente para a coleta de informações que visam descrever, comparar ou explicar conhecimentos, atitudes e comportamentos [Kitchenham e Pfleeger, 2008]. A finalidade de um *survey* é produzir estatísticas, ou seja, descrições quantitativas ou numéricas de alguns aspectos da população de estudo.

Sendo assim, este *survey* pretende coletar informações sobre as opiniões de alunos e professores, representados por uma amostra, em relação ao ensino de ES. Este *survey* foi aplicado em cursos de graduação em Computação de universidades públicas e privadas do Brasil e segue os *guidelines* de Kitchenham e Pfleeger (2008). O protocolo do *survey* encontra-se disponível em *http://goo.gl/gqzMrP*.

2.1. Público-Alvo

Como público-alvo do *survey*, definiu-se:

I. Professore(a)s que lecionam/lecionaram a disciplina de ES a partir de 2005 (ano de publicação do currículo de referência da SBC); e

II. Aluno(a)s que concluíram a disciplina de ES entre 2005 e 2015 (período de vigência do currículo de referência da SBC).

Foram excluídos os participantes que não atendiam estes critérios ou que não estavam motivados a participar da pesquisa ou, ainda, aqueles que claramente não conseguiam responder as questões de pesquisa. Estes critérios de inclusão e exclusão foram divulgados no protocolo do *survey*.

2.2. Questionários

Foram utilizados como instrumentos para aplicação do *survey* 2 (dois) questionários auto administrados, com questões de pesquisa que tinham um conjunto de respostas pré-definidas. Estas questões de pesquisa foram fortemente baseadas nos *surveys* aplicados por Wangenheim e Silva (2009) e Lethbridge (2000). Revisando os currículos de referência da ACM/IEEE (2013) e da SBC (2005), identificaram-se 83 tópicos de ES e 125 aprendizagens esperadas, classificados e organizados em 10 unidades de conhecimento: 1) *Processos de Software;* 2) *Gerenciamento de Projetos de Software;* 3) *Ferramentas e Ambientes;* 4) *Engenharia de Requisitos;* 5) *Projetos de Software;* 6) *Construção de Software;* 7) *Verificação e Validação de Software;* 8) *Evolução de Software;* 9) *Confiabilidade de Software;* e 10) *Métodos Formais.*

No que diz respeito às abordagens de ensino de Engenharia de Software, identificou-se o trabalho de Prikladnicki et al. (2009), que destaca os principais métodos de ensino e abordagens de avaliação adotadas nas disciplinas de Engenharia de Software no Brasil. A partir desta pesquisa, foram selecionadas para análise os seguintes elementos de ensino: A) *Métodos de Ensino;* B) *Abordagens de Ensino;* e C) *Estratégias de Avaliação*. Não se objetivou listar todas os tópicos e abordagens de ensino de Engenharia de Software, mas sim ter uma ampla cobertura destes.

Além disso, foram incluídas questões demográficas a fim de caracterizar os participantes. Antes da coleta de dados, foi realizado um pré-teste de aplicação do *survey* com 2 professores e 4 alunos de instituições públicas e privadas a fim de identificar possíveis inconsistências e o tempo necessário para responder todas as questões de pesquisa dos questionários. Os participantes responderam aos questionários e indicaram que o tempo para responder todas as questões foi em média 32 minutos.

Estes questionários foram disponibilizados na web, utilizando a ferramenta *Google Forms*, que permite a coleta de informações através de uma pesquisa personalizada que é automaticamente ligada a uma planilha. Para reduzir a influência do cansaço nas respostas, as perguntas relacionadas as 125 aprendizagens esperadas e aos 83 tópicos de ES foram divididas em diferentes telas de acordo com as 10 unidades de conhecimento. Os questionários encontram-se disponíveis em *http://goo.gl/vn5jHS*. Sua divulgação foi realizada em listas de e-mails, grupos da área de ES em redes sociais, e *in loco* em universidades públicas e privadas de Belém-PA e Recife-PE.

A. Survey com Professores da Disciplina de Engenharia de Software

A Tabela 1 mostra as 3 (três) questões de pesquisa feitas para os professores da disciplina de Engenharia de Software a fim de identificar quais tópicos e quais abordagens de ensino estão sendo adotados.

Tabela 1 – Questões e Respostas para os Professores

Questão	**Opções de Respostas**
Q1. *Quais currículos de referências são adotados na definição da ementa da disciplina de Engenharia de Software?*	( ) *Curriculum Guidelines* da ACM/IEEE; ( ) Currículo de Referência da SBC; ( ) Outros; ( ) Nenhum.
Q2. *Quais tópicos de Engenharia de Software estão sendo contemplados na ementa destas disciplinas?*	Para cada um dos 83 tópicos de ES, o professor deveria responder: [ ] Contemplado [ ] Não Contemplado
Q3. *Quais abordagens de ensino são adotadas na disciplina de Engenharia de Software?*	A. Métodos de Ensino (quanto ao papel do professor, objetivos, etc.); B. Abordagens de Ensino (aulas expositivas, uso de jogos, etc.); C. Estratégias de Avaliação (provas individuais, trabalhos práticos, projeto de software, etc.).

B. Survey com Alunos Concluintes da Disciplina de Engenharia de Software

A Tabela 2 mostra as 3 (três) perguntas feitas para os alunos concluintes da disciplina de Engenharia de Software a fim de analisar a aprendizagem e sua preferência por abordagens de ensino. Para as questões Q1 e Q2 foi utilizada uma escala *Likert* de 0 até 5 para caracterizar os graus de utilidade e aprendizagem, respectivamente.

Tabela 2 – Questões e Respostas para os Alunos

Questão	**Opções de Respostas**
Q1. *O quão útil considera a disciplina de Engenharia de Software para a sua formação profissional?*	0.Completamente inútil 1.Quase nunca útil 2.Ocasionalmente útil 3.Moderadamente útil 4.Muito útil 5.Essencial
Q2. *O quanto aprendeu das Unidades de Conhecimento ministradas durante a disciplina de Engenharia de Software?*	Para cada uma das 125 aprendizagens de ES, o aluno deveria responder: 0.Não aprendi absolutamente nada 1.Aprendi vagamente 2.Aprendi o básico 3.Aprendi moderadamente 4.Aprendi muito 5.Aprendi em profundidade
Q3. *Quais abordagens de ensino de Engenharia de Software considera melhor para sua aprendizagem?*	A. Abordagens de Ensino (aulas expositivas, uso de jogos, etc.); B. Estratégias de Avaliação (provas individuais, trabalhos práticos, projeto de software, etc.).

2.3. Aplicação do *Survey*

Os dados foram coletados durante o período de 16 de Março a 29 de Maio de 2015. Durante esse período, recebeu-se respostas completas de 70 participantes, sendo 23 professores e 47 alunos. É uma baixa amostra, considerando que o *survey* foi divulgado para cerca de 50 professores e mais de 350 alunos.

Desta forma, a fim de reduzir este viés da amostragem, aplicou-se o *survey* em todas regiões do Brasil. Obtive-se respostas de 20 instituições de ensino públicas e privadas. A instituição que teve o maior número de participantes foi a Universidade Federal do Pará (UFPA), da qual participaram 3 professores e 20 estudantes. Quanto a região, 10 instituições do Nordeste participaram da pesquisa.

Os participantes representam 12 estados do Brasil, cerca de 44% do total de estados. Destes, 50% são de instituições da região Nordeste, 25% do Norte, 15% do Sul, 5% do Centro-Oeste e 5% do Sudeste. A maioria dos participantes, 80%, são de instituições públicas de ensino e 20% são de instituições privadas.

2.4. Os Participantes do *Survey*

A média de idade dos professores participantes é de 38 anos. A moda do ano da última formação destes professores foi 2011, sendo a formação mais recente em 2015 e a mais antiga em 1998 (há 17 anos). Quanto a titulação, 4% possui Especialização, 39% possui Mestrado assim como 39% possui Doutorado. 18% dos professores entrevistados possuem pós-doutorado. Em média, estes professores lecionam a disciplina de ES há 8 anos, sendo o maior tempo de atuação 25 anos e o menor 1 ano.

A média de idade dos alunos participantes é de 24 anos. A moda do ano em que estes concluíram a disciplina de ES foi 2014, sendo a formação mais recente em 2015 e a mais antiga em 2005 (há 10 anos).

3. Resultados do *Survey*

3.1. Professores e a Adoção dos Tópicos de Engenharia de Software

A. Currículos de Referências adotados na ementa de Engenharia de Software

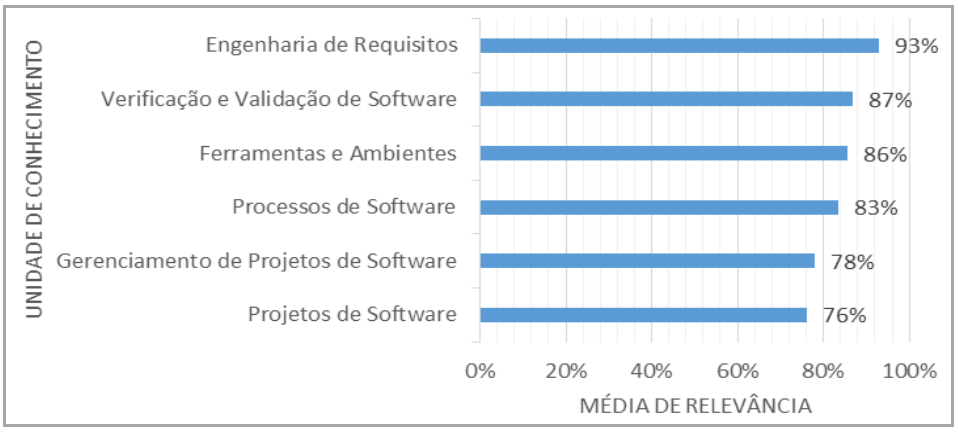
Em relação aos currículos de referência adotados pelos professores, observa-se que a maioria (24%) adota o currículo de referência da SBC. Há um equilíbrio quanto os demais, pois 16% utiliza o currículo da ACM/IEEE e outros 16% adota estes 2 currículos. 16% dos professores adota um currículo definido pela própria instituição e 12% considera outras abordagens, como SWEBOK e ENADE. Por fim, 16% dos professores não adota nenhum currículo de referência na definição de suas ementas.

B. Tópicos contemplados na ementa da disciplina de Engenharia de Software

Por restrições de espaço, não serão apresentados os resultados da adoção para cada um dos 83 tópicos. Serão apresentados os percentuais de adoção para cada unidade de conhecimento das quais estes tópicos fazem parte. O relatório completo, com todos os resultados apresentados e discutidos, encontra-se disponível em *http://goo.gl/CIzt4q*.

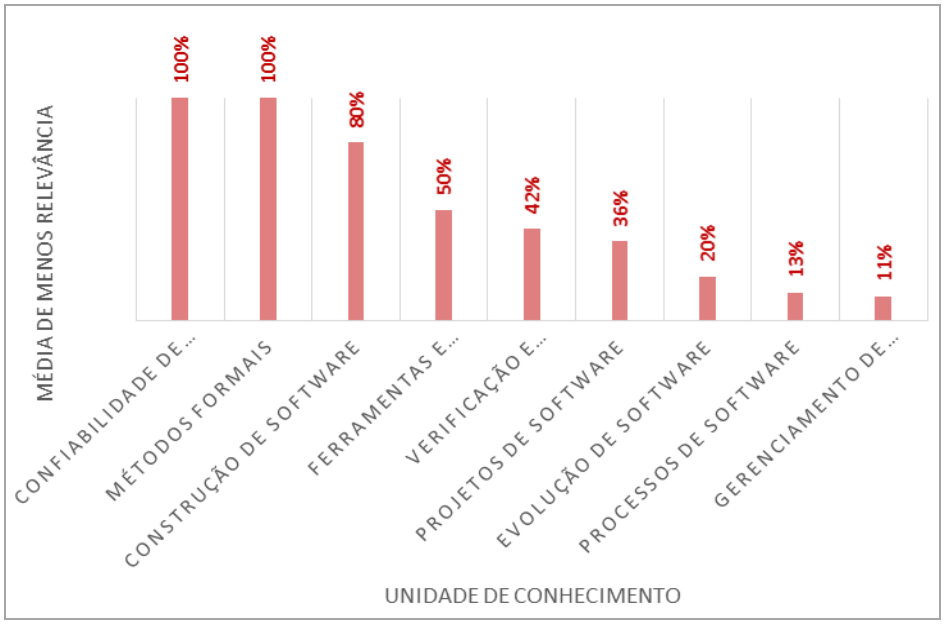
Há várias características que determinam a relevância de um tópico. Porém, neste *survey*, é tratada apenas a característica de adoção, considerando-se como relevantes os tópicos que são adotados por mais de 70% dos professores. Desta forma, é possível analisar a média de relevância obtida pelas unidades de conhecimento que são compostas por estes tópicos. No Gráfico 1, observa-se que a área de Engenharia de Requisitos é considerada a área de maior relevância.

Gráfico 1 - Média de Tópicos Relevantes por Unidade de Conhecimento



Em relação aos tópicos menos relevantes, consideraram-se aqueles que são adotados por menos de 40% dos professores. O percentual de unidades de conhecimento com tópicos menos relevantes é apresentado no Gráfico 2, onde observa-se que as áreas de Confiabilidade de Software e Métodos Formais possuem a maior quantidade de tópicos, ou seja, 100% de seus tópicos são adotados por menos de 40% dos professores.

Gráfico 2 - Percentual de Tópicos Menos Relevantes por Área



C. Abordagens de Ensino adotadas na disciplina de Engenharia de Software

Em relação aos métodos de ensino adotados pelos professores, há um certo equilíbrio, pois 42% dos professores adotam abordagens de ensino que focam nos alunos e 58% focam as aulas em si, ou seja, o professor é o principal fornecedor da informação.

Quanto as abordagens de ensino adotadas pelos professores, observou-se que todos adotam aulas expositivas (100%) e a maioria discute casos práticos com os alunos (90%), realiza projetos de software (86%) e ministra aulas de laboratório (71%).

Por fim, quanto as estratégias de avaliação adotadas pelos professores, observou-se que a grande maioria realiza trabalhos práticos (95%), provas individuais (90%) e avalia produtos de trabalhos gerados em projetos de software (90%).

3.2. Alunos e a Aprendizagem dos Tópicos de Engenharia de Software

A. Utilidade da disciplina de Engenharia de Software

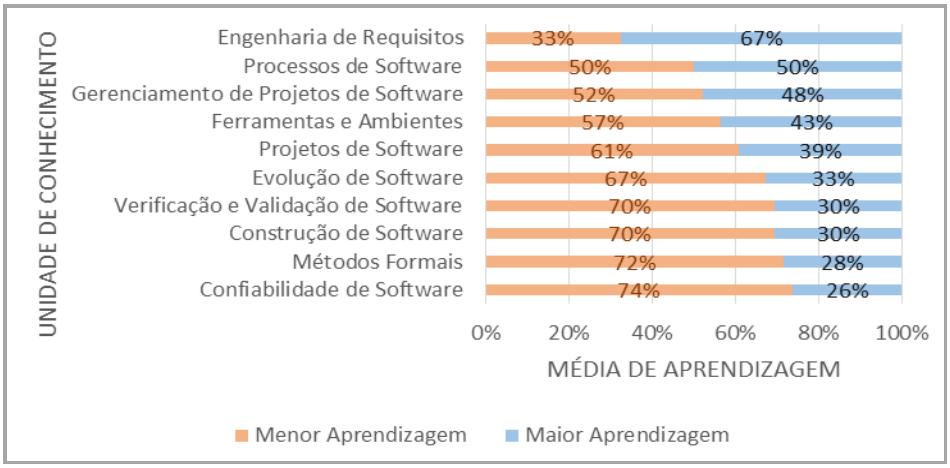
Em relação a percepção dos alunos quanto a utilidade da disciplina de Engenharia de Software para sua formação profissional, 63% considera a disciplina essencial, 20% muito útil e 7% moderadamente útil. Observa-se que 10% dos alunos entrevistados não considera a disciplina muito útil.

B. Aprendizagem dos Tópicos de Engenharia de Software

Observa-se que a Engenharia de Requisitos é a unidade de conhecimento que possui a maior porcentagem de aprendizagem, ou seja, 67% de seus tópicos são efetivamente aprendidos pelos alunos.

Conforme apresenta o Gráfico 3, observa-se que, além da Engenharia de Requisitos, as unidades Processos de Software, Gerenciamento de Projetos de Software e Ferramentas e Ambientes possuem grande aprendizagem, acima de 40%. Já as unidades Verificação e Validação de Software, Construção de Software, Métodos Formais e Confiabilidade de Software apresentam menor aprendizagem, abaixo de 30%.

Gráfico 3 - Média Percentual de Aprendizagem por Unidade de Conhecimento



C. Abordagens de Ensino Consideradas Efetivas

Quanto as abordagens de ensino consideradas efetivas pelos alunos, observou-se que a grande maioria considera a realização de Projetos de software (89%), Aulas de laboratório (72%) e Discussão de casos práticos (65%).

Por fim, quanto as estratégias de avaliação adotadas pelos professores, observou-se que a grande maioria dos alunos considera efetivo a realização de trabalhos práticos (89%) e a entrega de produtos de trabalhos gerados em projetos de software (85%).

4. Discussões sobre os Resultados

4.1. Sobre a Adoção de Currículos de Referência

Em relação a adoção de currículos de referência, observa-se que 56% dos professores adota um currículo de referência para definir suas ementas da disciplina de Engenharia de Software. É um baixo percentual, considerando que um total de 44% dos professores não adota estes currículos de referência.

É de suma importância a adoção destes currículos, pois estes são discutidos e elaborados por órgãos representativos da área, como a Sociedade Brasileira da Computação (SBC) e ACM/IEEE. Além disso, definem uma estrutura de conceitos inter-relacionados a fim de especificar diretrizes de ensino e formação profissional de acordo com as áreas da Computação, sendo uma delas a Engenharia de Software. Estas diretrizes definem o perfil profissional e acadêmico esperado para os estudantes da área, bem como apresenta a estruturação e detalhamento das matérias, como carga horária, tópicos a serem abordados e aprendizagens esperadas para cada um destes tópicos.

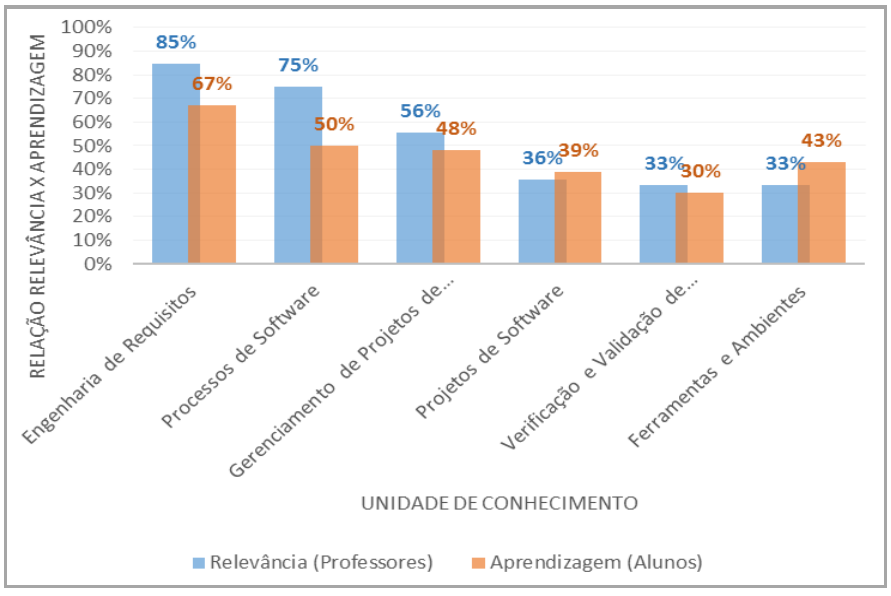
Sem a adoção destes currículos, os professores acabam por estabelecer ementas incompatíveis com diretrizes nacionais e internacionais de ensino e, possivelmente, não atendendo o que se espera de um curso de graduação em computação.

4.2. Sobre a Relevância dos Tópicos

A. Os Mais Relevantes

A partir da identificação das 6 unidades mais adotadas pelos professores nas ementas da disciplina de ES, é possível correlacionar o percentual de tópicos relevantes com o percentual de aprendizagem dos alunos, conforme Gráfico 4.

Gráfico 4 - Correlação entre Tópicos Mais Relevantes e Aprendizagem



Acredita-se que a relevância da unidade Engenharia de Requisitos deve-se ao fato de que a Engenharia de Software é uma disciplina preocupada com o desenvolvimento efetivo e eficiente de sistemas de software que satisfaçam os requisitos

dos usuários [ACM/IEEE, 2013]. Para satisfazer as necessidades dos usuários, é de extrema importância o conhecimento de técnicas de elicitação, modelagem e escrita de requisitos, tópicos estes abordados nesta unidade. Além disto, os profissionais da área devem ter a capacidade de entender o desenvolvimento de software como um processo a fim de assegurar prazos, custos e a qualidade do produto a ser desenvolvido. Neste contexto, insere-se a segunda unidade mais relevante, Processos de Software.

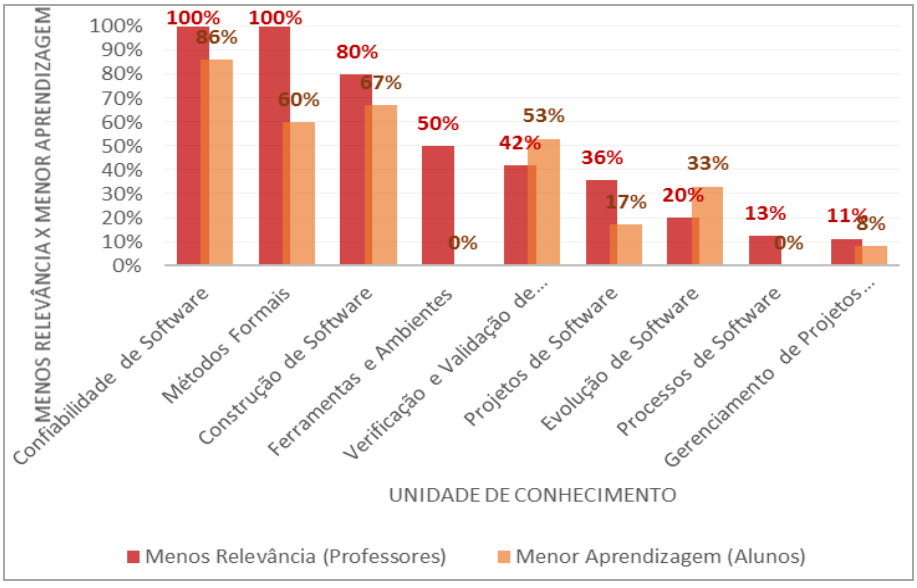
No entanto, não basta definir um processo e um plano de projeto, é necessário colocá-lo em prática e fazer o acompanhamento deste para realizar ajustes caso necessário. Neste contexto, se insere a terceira unidade mais relevante, Gerenciamento de Projetos de Software. A relevância desta unidade deve-se a importância e a dificuldade de realizar a gerência de um Projeto.

Por fim, quanto as três unidades restantes, observa-se uma complementariedade. A unidade de Projetos de Software consiste em adotar padrões de projetos, arquiteturas, interfaces, a fim de atender um conjunto específico de Requisitos, seguindo um Processo de Software. A unidade de Verificação e Validação complementa a unidade de Projeto na medida em que é responsável pelos testes e auditorias de qualidade do processo executado e dos produtos de trabalho gerados. Por fim, a unidade Ferramentas e Ambientes apoia a e modelagem de Processos, bem como a execução de Projetos na medida que permite automatizar atividades, como testes e controle de versão.

B. Os Menos Relevantes

Também é possível correlacionar o percentual de tópicos menos relevantes com o percentual de baixa aprendizagem dos alunos, conforme Gráfico 5.

Gráfico 5 - Correlação entre Tópicos Menos Relevantes e Aprendizagem



O fato das unidades Confiabilidade de Software, Métodos Formais e Construção de Software serem menos relevantes para os professores, impacta diretamente na aprendizagem dos alunos, que demonstraram baixa aprendizagem, 86%, 60% e 67% respectivamente, para os tópicos ministrados nestas unidades. A quarta unidade menos abordada pelos professores, Evolução de Software, possui 20% de tópicos considerados menos relevantes e 33% de baixa aprendizagem em relação a estes.

4.3. Sobre as Abordagens de Ensino

Em relação aos métodos de ensino, destaca-se que aulas centradas no professor geralmente são mais expositivas. Uma aula expositiva acaba sendo pouco eficiente, pois ativa apenas o sentido da audição. Por outro lado, quanto mais prática e dinâmica for a aula, mais ela será centrada no aluno. A aprendizagem mais prática acaba sendo caracterizada pelo envolvimento dos alunos em determinadas atividades, com resultados mais positivos [Prikladnicki et al., 2009].

A partir dos resultados deste *survey*, observa-se que os alunos estão mais interessados em realizar atividades práticas, como projetos de desenvolvimento em laboratórios que simulem situações próximas as que vão encontrar no mercado. Acredita-se que isso se deve ao fato da disciplina Engenharia de Software possuir muitos tópicos, 83 no total, o que acaba por torná-la menos atrativa para alunos que não possuem uma afinidade com a área. A abordagem prática permite a estes alunos fixarem melhor os conceitos a partir da sua aplicação.

Em relação as estratégias de ensino, houve convergência em relação a adoção pelos professores e a preferência dos alunos por Trabalhos práticos e expositivos, Entrega de produtos de trabalho, Participação e Aprendizagem. Esta preferência pelas estratégias de avaliação práticas reforça ainda mais o interesse dos alunos por abordagens de ensino práticas.

5. Considerações Finais

5.1. Limitações e Ameaças à Validade

Inicialmente, pretendia-se consultar, além de professores e alunos, profissionais da indústria em relação à relevância dos tópicos de ES. No entanto, não se obteve uma quantidade de respostas considerável. Isso deve-se ao tempo necessário para preenchimento do questionário, conforme relatado por alguns profissionais que foram convidados a responder o *survey*.

Além disto, cerca de 27 professores e 303 alunos foram excluídos do *survey*, pois não estavam motivados a responder o questionário devido o tempo estimado para preenchimento. A quantidade de participantes excluídos foi maior que a amostra obtida. Isto impactou diretamente a abrangência e o resultado do *survey*. Esse viés de amostragem foi difícil de tratar, pois o público-alvo reclamou muito da quantidade de tópicos e, consequentemente, do tempo necessário para responder ao questionário. Neste primeiro momento, não se poderia restringir a quantidade de tópicos analisados, pois isto comprometeria o objetivo da pesquisa. Para reduzir a influência do cansaço nas respostas, as perguntas relacionadas as 125 aprendizagens esperadas e aos 83 tópicos de ES foram divididas em diferentes telas de acordo com as 10 unidades de conhecimento.

O viés de amostragem causa problemas em generalizar os resultados da pesquisa, pois os entrevistados podem não ser uma amostra representativa da população-alvo [Kitchenham e Pfleeger, 2008]. A fim de tratar este viés, e obter uma amostra representativa, buscou-se diversificar a quantidade de instituições participantes das diversas regiões do Brasil.

A cobertura das abordagens de ensino foi baixa, baseando-se apenas no trabalho de Prikladnicki et al. (2009). No entanto, não é objetivo deste estudo listar todas os tópicos e abordagens de ensino de Engenharia de Software, mas sim ter uma ampla cobertura destes. Por fim, destaca-se que, a fim de atender seu objetivo principal que consiste em definir uma abordagem de ensino, este estudo considera apenas a característica de adoção para classificar os tópicos mais relevantes e menos relevantes.

5.2. Conclusões

O cenário atual do ensino demonstra que determinados tópicos são considerados de menor relevância, pelos professores e, consequentemente, apresentam baixo grau de aprendizagem pelos alunos. Acontece que estes tópicos consomem carga horária considerável da disciplina de Engenharia de Software, enquanto que alguns tópicos, considerados mais relevantes, apresentam baixa aprendizagem. Talvez isso deve-se ao fato de não haver tempo hábil para ministrar de maneira efetiva todos os tópicos destas unidades relevantes.

Analisando os resultados obtidos, acredita-se que as 6 unidades de conhecimento consideradas relevantes são totalmente complementares, sendo possível correlacionar seus tópicos. Essa correlação se apresenta como um trabalho futuro desta pesquisa, onde definir-se-á um plano de ensino baseado nestas unidades de conhecimento. Para as estratégias de ensino, deve-se ter uma atenção maior para os tópicos de Verificação e Validação de Software, Projetos de Software e Gerenciamento de Projetos de Software que apesar de considerados relevantes, apresentaram um baixo índice de aprendizagem pelos alunos entrevistados.

Analisando as abordagens de ensino, observa-se uma convergência entre alunos e professores no que diz respeito à adoção de métodos práticos de ensino e avaliação. O caráter da disciplina de Engenharia de Software, tradicionalmente teórico, faz com que estes prefiram aplicar os tópicos sugeridos pelos currículos de referência através de Projetos de software, onde o aluno é avaliado pela entrega de produtos de trabalho gerados (como Lista de Requisitos, Diagramas UML, código-fonte, dentre outros) e/ou Aulas de laboratório, onde os alunos apresentam trabalhos práticos e expositivos.

Por fim, além das justificativas apresentadas neste trabalho para a relevância destas unidades, destaca-se que os professores tendem a ministrar mais adequadamente os tópicos ligados a essas unidades porque estes são mais compreensíveis do que os tópicos de outras unidades (como Confiabilidade de Software e Métodos Formais). Desta forma, conclui-se que essa análise de relevância deve ser evolutiva, sendo refinada continuamente em trabalhos futuros.

5.3. Próximas Etapas

Para melhorar o cenário de ensino identificado neste *survey*, pretende-se, como próxima etapa desta pesquisa, definir um plano de ensino que foque nos tópicos considerados relevantes para a formação profissional dos alunos de Engenharia de Software. Além disto, pretende-se definir uma abordagem de ensino mais voltada para o desenvolvimento de habilidades e competências profissionais nos alunos, a fim de aumentar a aprendizagem destes tópicos e, consequentemente, formar profissionais mais preparados para atender às demandas da indústria de software.

Tal melhoria exige um trabalho mais colaborativo entre pesquisadores e profissionais da indústria com o intuito de se concentrar no seguimento de práticas de capacitação que realmente desenvolvam competências e habilidades profissionais e reflitam a realidade do contexto no qual os alunos atuarão após a graduação. Sendo assim, uma outra etapa desta pesquisa consistirá em um levantamento de práticas de capacitação adotadas por consultores em Melhoria do Processo de Software (MPS) relacionadas às unidades de conhecimento da ES. Espera-se, desta forma, atender as necessidades da indústria em relação à formação de profissionais de ES.

Agradecimentos

Os autores gostariam de agradecer à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro ao desenvolvimento desta pesquisa.

Referências

ACM/IEEE (2013) "Computer science curricula 2013 – Curriculum guidelines for undergraduate degree programs in Computer Science", https://www.acm.org, April.

Bessa, B. Cunha, M. e Furtado, F. (2012). ENGSOFT: Ferramenta para Simulação de Ambientes Reais para auxiliar o Aprendizado Baseado em Problemas (PBL) no Ensino de Engenharia de Software. Em *Anais do XX Workshop sobre Educação em Informática*. Curitiba, Brasil.

Duley, R. et al. (2003). Engineering an introductory software engineering curriculum. In *Proceedings of 16th Conference on Software Engineering Education and Training*, pages 99-106.

Kitchenham, B. e Pfleeger, S. (2008) "Personal Opinion Surveys", In: Guide to Advanced Empirical Software Engineering, Springer, ch. 3, pages 63-92.

Lethbridge, T. (2000) "What Knowledge Is Important to a Software Professional?" IEEE Computer, 33/5.

Lethbridge, T. et al. (2007). Improving software practice through education: Challenges and future trends. In *Proceedings of the Conference Future of Software Engineering*, pages 12-28.

Nunes, D. Reis, C. e Reis, R. (2010) "Educação em Engenharia de Software", Em: A carreira do pesquisador em Engenharia de Software: princípios, conceitos e direções. 1ª edição. Salvador, Brasil: UFBA, cap. 5, páginas 132-181.

Prikladnicki, R. et al. (2009). Ensino de Engenharia de Software: Desafios, Estratégias de Ensino e Lições Aprendidas. Em *Anais do II Fórum de Educação em Engenharia de Software*. Fortaleza, Brasil.

SBC (2005). Currículo de Referência da SBC para Cursos de Graduação em Bacharelado em Ciência da Computação e Engenharia de Computação, http://www.sbc.org.br, Abril.

Wangenheim, C. e Silva, D. (2009). Qual Conhecimento de Engenharia de Software é Importante para um Profissional de Software? Em *Anais do II Fórum de Educação em Engenharia de Software*. Fortaleza, Brasil.

Does Online Content Support UML Learning? An Empirical Study

Adriano Santos, Gustavo Vale, Eduardo Figueiredo

Software Engineering Laboratory (LabSoft), Department of Computer Science (DCC),
Federal University of Minas Gerais (UFMG)
Belo Horizonte, Brazil

{adriano, gustavovale, figueiredo}@dcc.ufmg.br

Abstract. *Open online courses are a method of online lecturing whose application in education is not bounded by space and location constraints. The successful implementation of open courses requires conceptual changes in how instructors and students behave in open unbounded education environment. Common features available in online courses are video lectures and online questionnaires. However, there is little knowledge on the efficacy of these features to support learning of software modeling in UML. To fill this gap, this paper presents an empirical study to evaluate whether online content supports learning of UML diagrams, such as Use Cases, Class and Sequence Diagrams. This study involved 193 students in three consecutive years of a Software Engineering course. All students had face-to-face lectures, but features of online teaching were gradually included: video lectures in the 2nd year and online questionnaires in the 3rd year. Student performance was assessed based on their grades in the course face-to-face exams. Our results indicate that students who answer online questionnaires have better grades in face-to-face exams. However, we could not always verify improvements in the performance of students due to video lectures watched.*

1. Introduction

Internet has become an important tool to modify education through open online courses. An open online course creates an educational environment not bounded by space and location constraints [Fox and Patterson, 2012]. The success of open online courses requires conceptual changes in the way as lectures and students behave in such an open unbounded environment [Fox and Patterson, 2012; Liyanagunawardena et al., 2012]. Supporters of open education claim that the online content, such as video lectures and online questionnaires, is an important tool in the learning process. In some sorts of open courses, students can join classes at any time, do exercise, read past discussions, and so on [Schimidt and McCormick, 2013]. Such innovation in terms of education allows discussion-oriented classrooms, i.e., face-to-face classes are dedicated for questions and activity solutions with closer interaction between students and teachers.

Several respectful universities around the world, mainly in North America and Europe, are providing open online courses based on to their face-to-face equivalent courses. These courses have attracted great attention of hundreds of thousands of worldwide students [Fox and Patterson, 2012; Schimidt and McCormick, 2013; SaaS, 2015]. For instance, Harvard University and Massachusetts Institute of Technology have

invested on the creation of various online courses by the edX[1] portal. In addition, more than 80 American and European universities, such as Princeton and Stanford, are involved on creation of hundreds of online courses in several areas by means of Coursera[2]. Many other courses have been successfully created in Udacity[3] and similar open learning portals, such as Udemy[4]. In addition to video lectures, students regularly registered in an online course can answer online questionnaires and have an assessment of their performance. In some courses, a student who successful completes the course obtains a certificate or statement of accomplishment.

There are a couple of online courses to teach subjects related to Software Engineering [Schimidt and McCormick, 2013; SaaS, 2015; Tilmann et al., 2013]. However, these courses focus either on Pattern-Oriented Software Architecture [Schimidt and McCormick, 2013], Software as a Service [SaaS, 2015], Introduction on Software Engineering [Pereira et al., 2013; Figueiredo et al., 2014] or on how to program [Tilmann et al., 2013]. For the best of our knowledge, there is no open online course to teach software modeling using the Unified Modeling Language (UML) [Booch, Rumbaugh, Jacobson, 2005]. More important, there is no systematic study to investigate whether this way of teaching is efficient and viable to teach the basic concepts of UML, such as Class Diagram, Use Case Diagram, and Sequence Diagram. Therefore, it is essential for us to investigate and evaluate the actual benefits and drawbacks of online courses in order to understand whether and how such courses can indeed improve the learning of UML concepts.

Every year at the Federal University of Minas Gerais (UFMG) in Brazil, more than 100 students take a face-to-face Software Engineering course. Since 2013, an educational online platform is used to complement classroom lessons. Actually, this online course provides access to 44 online video lectures and 16 online questionnaires [Figueiredo et al., 2014]. Moreover, conversations between students and the course instructors are possible due to the discussion forums and emails, which provide a highly interactive virtual Software Engineering learning community. As part of this course, it is presented an introduction to UML. This part is important to the course and, it is responsible for about 20% of the course content (7 videos and 3 online questionnaires).

In this context, this paper evaluates the UML part of the Software Engineering course by comparing the performance of students over three consecutive years. Two features of online learning were gradually introduced to the course: video lectures in the 2nd year and online questionnaires in the 3rd year. Online questionnaires are commonly used by the students to review for the face-to-face exam [Garcia et al., 2014]. We evaluate the grades of exam questions related to UML of 193 students divided into 6 different semesters. Our results show that videos and online questionnaires contribute to the improvement of up to 20% of student grades in UML questions when compared with students who did not use videos and online questionnaires. However, based on two

[1] http://www.edx.org/

[2] http://www.coursera.org/

[3] http://www.udacity.com

[4] http://www.udemy.com/

exam questions that repeated over four semesters, we verify that about 15% of grade improvement seems to be related to online questionnaires, rather than watched video lectures.

The rest of this paper is structured as follows. Section 2 presents some related work. Section 3 provides a background in some UML content and introduces the Software Engineering open online course. Section 4 presents the study settings, the evaluation of the UML content of the online course. Threats to validity are discussed in Section 5. Finally, Section 6 described our conclusions and future work.

2. Related Work

Many studies have been proposed to improve Software Engineering teaching [Liyanagunawardena et al., 2012; Dasarathy et al., 2014; Figueiredo et al., 2007; Figueiredo et al., 2014; Meirelles et al., 2011; Papaspyrou et al., 1999; Potter and Shots, 2011]. Most of these studies focus on software engineering and online courses, compated to traditional face-to-face courses in universities [Xie et al., 2013]. However, these studies do not analyze the impact of online environment on student learning of UML-related topics. We provide a different evaluation compared to related work because we focus on two common features of online support and their impact on grades of students regularly enrolled in a face-to-face course.

Ahmadi and Jazayeri (2014) discuss the process of student learning while the students are doing online activities. They analyze the results of online questionnaires and activities realized by students during an experiment. Nevertheless, the authors did not perform any comparison with the student results in traditional teaching methods.

Fox and Patterson (2012) offer a hybrid course (part online and part face-to-face) in Software Engineering, focused on test-driven development (TDD) and agile software development at Berkeley University. They report the use of agile development techniques to teach a face-to-face course with more than 100 junior and senior students at Berkeley University. Fox and Patterson (2012) also discuss their experience in providing the first part as an open course to 50,000 online students, most of who work in the IT industry. However, their paper does not present any assessment to verify whether the proposed online course has positive or negative aspects in the student learning. Unlike Fox and Patterson (2012), our work focuses on UML teaching.

Schmidt and McCormick (2013) propose a course to teach design and architecture patterns for development of concurrent and networked software systems. They focus on some specific topics in the long list of content covered by an introductory Software Engineering course. Similar to their work, we evaluate in this paper one facet of the Software Engineering discipline (UML diagrams) rather than the extensive course content. That is, we do not aim to assess learning of Software Engineering topics, such as software process, requirements engineering, and software maintenance.

Some studies [Ye et al. 2007] [Potter and Schots 2011] [Ahmadi and Jazayeri 2014] use open education as a complement to teaching Software Engineering. On the other hand, our work focuses on comparison of different levels of online support. We want to see what online supporting feature is more effective in helping students to learn UML diagrams.

3. Background

3.1. The Unified Modeling Language

The Unified Modeling Language (UML) is a general-purpose modeling language in the field of software engineering, which is designed to provide a standard way to visualize the design of a system [Booch et al., 2005]. UML is composed of two types of diagrams: behavioral and structural diagrams. Each diagram has elements that represent components, objects, users, modules, and states of a system. Figure 1 shows three elements of UML diagrams: a class `Account` of a Banking system, a `Manager` actor and an object `c` of class `Customer`. These elements are generally used in Class Diagram, Case Uses Diagram, and Sequence Diagram, respectively.

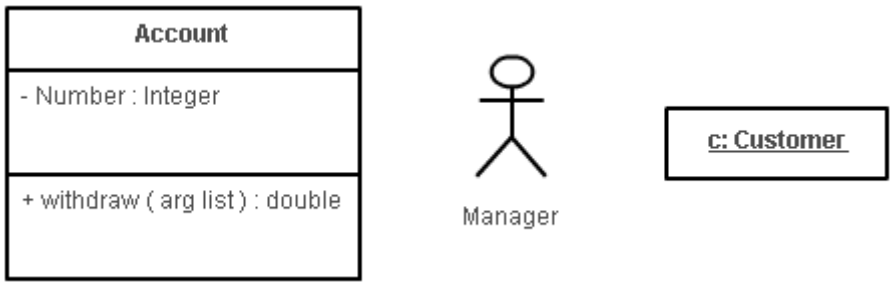


Figure 1. Elements of UML Diagrams

3.2. The Software Engineering Course and Our Preliminary Results

The online Software Engineering course was created in the first semester of 2013 [Pereira et al., 2013]. Initially, this course supported students of Federal University of Minas Gerais (UFMG) to learn concepts of Software Engineering. This course is free and based on two textbooks: *Software Engineering* [Sommerville, 2010] and *The Unified Modeling Language User Guide* [Booch et al., 2005]. It was make available online by using the Udemy platform. Udemy is a platform of online learning and management content that allows instructors create paid and free courses. By using Udemy, the instructors can provide video lectures, presentations, online questionnaires, and complementary files. The platform also allows that students discuss with others and with the instructors by means of discussion forums. Other features of Udemy include (i) annotations during the video lectures by students and (ii) statistics analyses for instructors.

Some results were presented to the community in previous work [Pereira et al., 2013; Figueiredo et al., 2014; Garcia et al., 2014]. The main results of our previous work can be summarized as follows: (i) students were engaged and motivated to participate in the online course, especially as a way to study for exams; (ii) exam grades of students in the face-to-face course with online support are significantly higher than the exam grades of students attending the same purely face-to-face course; (iii) frequency of students in the face-to-face course with online support was not a determinant factor for getting good grades; (iv) online questions with higher error rate are almost the same in more than one semester; (v) low correlation of correct answers in online questionnaires and the total number of videos watched; (vi) low to moderate

correlation of online questionnaires with frequency; (vii) low to moderate correlation of correct answers in online questionnaires and exams.

The preliminary results served as a motivation for us to continue the work in this paper. In this sense, this study aims to assess the progress in other dimensions, focusing mainly on the level of online support in a specific part of the course (UML content). Our online course has an audience of about 300 students currently registered, but we are analyzing in this paper data of only 193 students enrolled in the university face-to-face course in the last three years.

4. Evaluation of the Software Engineering Online Course

The online education has brought new challenges and potential benefits to traditional education. This section aims to evaluate the benefits and drawbacks of online education in leaning of UML diagrams. Section 4.1 presents the study settings and procedures we followed. Section 4.2 analyzes the student performance per year. Section 4.3 discusses performance focusing on two recurring questions in exams. Section 4.4 correlates the presence in UML face-to-face classes with the exam grades. Finally, Section 4.5 discusses and compares our results with previous sections.

4.1. Study Settings

We created three groups to investigate whether the use of online learning features contribute to improve student exam grades. Table 1 shows the three groups with the UML content covered during the course and how this content is made available to each group. Group A had only face-to-face classes; that is, with no online support. Group B had face-to-face classes and video lectures. Group C had face-to-face classes, video lectures, and online questionnaires. Additionally, it can be said that our study covers three consecutive years (2012, 2013, and 2014) and each group is composed by students from two different semesters of the same year. Therefore, Groups A, B, and C are composed by students from the first and second semester of 2012, 2013, and 2014, respectively. The number of students per semester and the total number of students can also be seen at the end of Table 1. For instance, Group A had 48 students in 2012-1 and 34 students in 2012-2, summing up to 82 students in 2012.

Table 1. The UML Content and the number of Students for each Group

UML Content	Group A	Group B	Group C
Introduction to UML	C1	C1,V1	C1,V1,Q1
Introduction to the main UML Diagrams	C2	C2,V2	C2,V2,Q1
Use Case Diagram	C2	C2,V3	C2,V3,Q1
Class Diagram	C3	C3,V4	C3,V4,Q2
Sequence Diagram	C3	C3,V5	C3,V5,Q2
Communication Diagram	C4	C4,V6	C4,V6,Q3
Activities Diagram	C4	C4,V7	C4,V7,Q3
Total students per semester	**48+34**	**37+22**	**20+32**
Total students per year	**82**	**59**	**52**

Considering all students from the three groups, we analyzed data from 193 students. These data were taken in the end of the semester and, only questions and lectures related to UML were analyzed in this paper. The same instructor was responsible by all face-to-face classes. It is also important to say that online

questionnaires are not equal to any exam questions. The UML content described in Table 1 was divided into four regular classes (C1, C2, C3, and C4), seven different video lectures (from V1 to V7), and three online questionnaires (Q1, Q2, and Q3).

4.2. Students Performance per Year

Figure 2 shows the average performance of students on exam questions in each semester. In 2012-1, the students got on an average grade of 55% and in 2012-2 they got an average of 64%, resulting in an average of 59% in 2012 (Group A). For Group B, the average of grades obtained was 52%. There was an intriguing low average grade in 2013-2: 35% (discussed below). Finally, for Group C, it was found an average of 80% and 78% for the semesters 2014-1 and 2014-2, resulting in an average of 79% for this group.

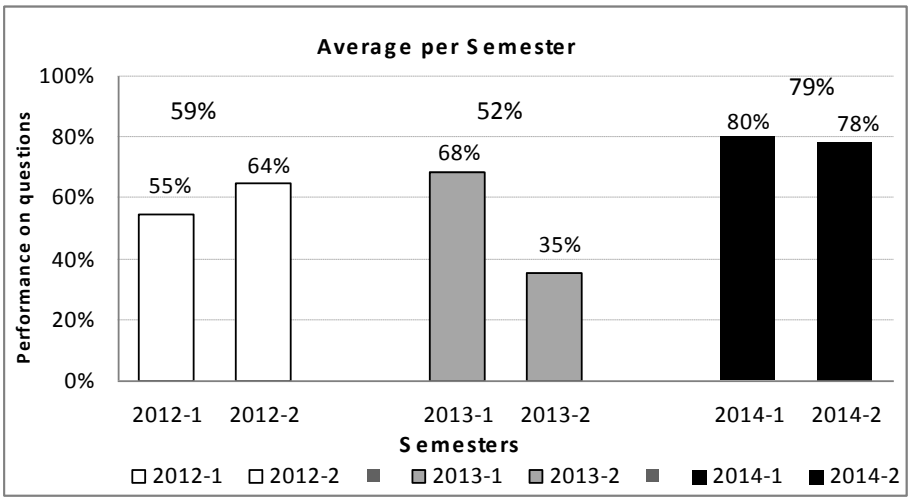


Figure 2. Student performance in questions relating to UML per Semester

Table 2 shows the number of exam questions applied per semester. A column with a percentage value means that the specific questions (represented by each line) appear in the corresponding semester. For instance, Questions 1 and 2 were included in the exams of 2012-2 and the average grade of the enrolled students was 73% and 56% for these questions, respectively. The unusual result in 2013-2 (see Figure 2) is probably due to the fact that only one question about UML was applied to students in this semester. After inspecting the 2013-2 exams, we observed that this UML question has a higher level of difficulty.

It is not trivial to measure how difficulty a question is, but note that the lowest average grades per semester occurred in 2012-1 and 2013-2 (Figure 2). In both semesters, exams had only one UML-related question. In spite of that, grades in 2012-1 were better than 2013-2. Additionally, we can note that Question 4 appeared also in 2014-1 and, despite students get an average of 54% in this question (20% higher than in 2013-2), this average is the lowest in the 2014-1. Therefore, we can conclude that Question 4 was more difficult than other UML questions. The last affirmation is also supported by Figure 2. Note that, the average grades in 2014-1 was 80%, yet the average grade for Question 4 was only 54%.

Table 2. UML questions applied in exams and total questions in each semester

Question	2012-1	2012-2	2013-1	2013-2	2014-1	2014-2
Question 1		73%	81%		92%	90%
Question 2		56%	57%		76%	75%
Question 3			83%		90%	
Question 4				35%	54%	
Question 5			52%			
Question 6					86%	
Question 7						74%
Question 8	55%					
Total Questions	**1**	**2**	**4**	**1**	**5**	**3**

By analyzing the different features of online support and still based on the results presented in Figure 2, it is clear that online questionnaires (Group C) help students to improve the exam grades. The improvement of Group C with respect to Groups A and B is an average of 20% and 27%, respectively. Related to support of video lectures, it is not clear an increased in student exam grades. On the contrary, we can observe a decreased of 7% in average grades from Group A to Group B. As previously discussed, this decrease can be due to uncontrolled confounding factors, such the difficulty of exam questions.

4.3. Analysis of Performance in Two Recurring Questions

In the previous analysis (Section 4.2), we said that it is hard to measure the difficulty of exam questions. To mitigate this threat, we compared only questions which repeat in more than 3 semesters. Therefore, this section evaluates the support of online content in two specific questions (1 and 2). These questions repeated in four semesters (2012-2, 2013-1, 2014-1, and 2014-2). Hence, these questions were applied in at least one semester of all analyzed groups.

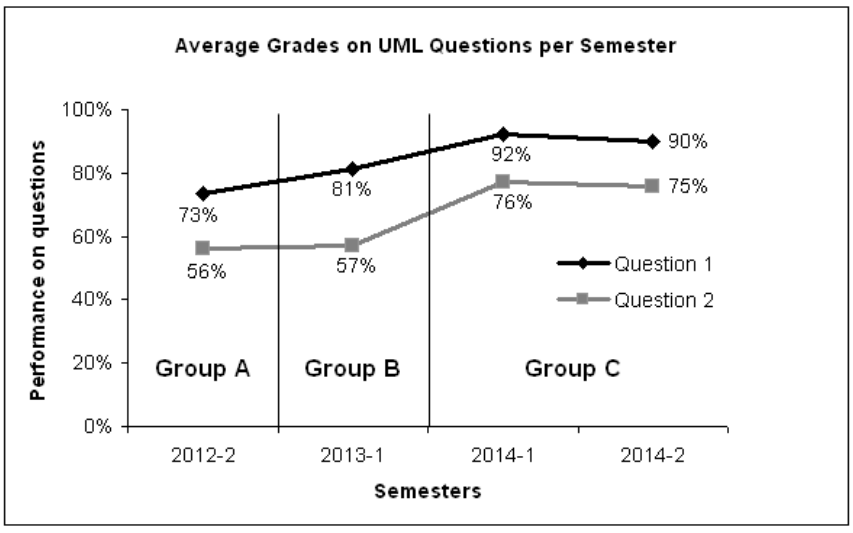


Figure 3. Average grades on two UML questions between semesters

Figure 3 shows that, for both Questions 1 and 2, there is an improvement in exam grades in semesters with some online support (since 2013-1). Exam grades had an improvement of 8% in average for Question 1 from Group B to Group A. Similarly, an improvement of 19% in exam grades can be observed from Group C to Group A. With

respect to Question 2, there is an improvement of 20% and 21 % in student grades of Group C compared to Groups B and A, respectively. These results show that online support have a positive impact in both levels (video lectures and online questionnaires). However, online questionnaires is mostly responsible for the improvement in the average exam grades of students.

4.4. Correlating Presence in Face-to-Face Classes with Exam Grades

We verified and analyzed the correlation (using Pearson coefficient) between the presence of students in face-to-face UML classes and exam grades considering only UML-related questions. Pearson Coefficient vary from -1 to 1, where -1 means a perfect negative correlation and 1 means a perfect positive correlation. Values close to 0 represents no linear correlation. With this analysis, we aim to see if online support is enough to teach students. In other words, considering Group A, if presence has a high positive correlation with student's exam grades it means that face-to-face classes are important to learn UML content. On the other hand, if the presence has non-correlation (close to 0) with student exam grades, it means that presence is not an important factor to learn UML content.

Table 3 shows the results of the correlation of presence in face-to-face classes and the student exam grades for all six semester considered in this study. For Group A, we obtained correlation values of 0.17 and 0.23 in 2012-1 and 2012-2, respectively. For Group B, we obtained 0.32 for 2013-1 and 0.45 for 2013-2. Finally, for Group C, we obtained correlation values of 0.65 and 0.60 for 2014-1 and 2014-2, respectively. The Pearson coefficient values were increasing from Group A to Group C. For Groups A and B, we considered a low correlation, close to no linear correlation (smaller than 0.5). These results suggest that face-to-face classes, regardless of video lectures, do not strongly support students to learn UML.

In the case of Group C, there is a moderate correlation between the presence of students in face-to-face and their exam grades. It is hard to explain such increase in the correlation. We believe that these results are related to how mature the online course becomes. That is, the online course in 2014 includes both video lectures and online questionnaires. As a result, students may have neglected face-to-face classes in favor of online video lectures and questionnaires. However, students who regularly attended face-to-face classes better learned UML than the ones who only relied on online content. As a general conclusion, we may say that face-to-face classes alone is not enough to support UML learning (Groups A and B), but student who skip these lectures cannot learn UML only based on online content (Group C).

Table 3. Correlation between presence in face-to-face classes and grades in exams

Semester	2012-1	2012-2	2013-1	2013-2	2014-1	2014-2
Correlation	0.17	0.23	0.32	0.45	0.65	0.60

4.5. Discussion

Observing the performance of students per year versus their performance for recurring questions, we note that the video lectures contributed to improving the exam grades of students in the analyzed questions. Certainly, there were questions with a high degree of difficulty that even with online content available, students could not obtain good

performance, as explained in Section 4.2. According to Figure 3, we can clearly see that for Question 1, there was an increase in performance of the Groups B and C compared to Group A. For Question 2, we see a small improvement from Group A to Group B, but it is not so clear. These two analyses lead us to conclude that student performance can be related to the type of question. In all analyzes, we have an improvement in the performance of Group C. Therefore, we conclude the online questionnaires in general support UML learning.

The clear improvement of online questionnaires and the questionable improvement of video lectures in UML learning probably are justified by the fact that students have an extra opportunity to practice their knowledge acquired in the face-to-face classes when they are doing the online questionnaires. They probably revisit other UML content and sources of information to resolve the questionnaires. This behavior contributes for an improvement in learning, because we are assuming that students who perform questionnaires and watches the video lectures spent more time studying than the ones who only attends face-to-face classes. Online questionnaires available in online learning environment are considered for evaluation. Hence, students earn points when they complete the questions, and this fact can motivate students to access the online platform and carry out the online activities. On the other hand, when only the video lectures are available, the students do not seem motivated because the access to video lectures does not count for their grades.

With our analysis of correlation, we can see that face-to-face classes are not so important to increase the exam grades of student in regular classes. This probably happen because in addition to online course, there are several sources of UML information, such as books, websites, and other online content. Therefore, students may consult other sources of information. We believe that other ways of teaching UML should be investigated, such as learn-by-doing and educational games. That is, face-to-face classes are important, but since UML is a visual content, it requires additional effort from the instructor to be taught.

The findings of our previous work [Pereira et al., 2013; Figueiredo et al., 2014; Garcia et al., 2014] has shown that the performance in online questionnaires have low or moderate correlation with performance in exams. In this paper, we are evaluating if the use of online activities, focusing on video lectures and online questionnaires, contribute to the increase of exam grades of students in UML content. Our results confirm that video lectures and online questionnaires available to students help to improve the UML learning. Previous work also shows that the error rate in online questionnaires is almost the same in more than one semester. In our study, exam questions with the highest error rate also occurred in more than one semester. This result suggests that students have similar profile and a difficult question for students in one semester probably also is difficult for students in other semesters.

5. Threats to Validity

Even with careful planning, this research can be affected by factors that could not be fully controlled, and may invalidate its main conclusions. Actions to mitigate their impact on the research results are described as follows.

The evolution of the course dynamics and students who failed. The evolution of the course dynamics can impact on the results. For example, changes in the order in which the contents are covered. In our work, the instructor followed the same general order of teaching in all semesters analyzed. Students who previously failed and attend the course a second time can also impact on the results because they may increase their exam notes. That is, these students already know the discipline and some exam questions. However, this threat is minimized because the course failure rate is of only 10% to 20%, and less than 5% of students in a class have attended it before.

The number of students assessed and exam questions for each semester. We consider that the number of students assessed in each semester is a minor threat because each class has 20 or more students and an average of 32 students per semester. Hence, in a class with 20 students, one or two high grades or low grades do not cause a big impact on the class average. The number of UML-related questions in exams is different between the semesters analyzed. However, some questions repeated in more than one semester allowing us mitigate the impact of question difficulty.

Online environment and student backgrounds. Online activities are not in a fully controlled environment. In other words, we cannot control whether students answer the questionnaires in groups or alone, or whether they cheat. However, we did not assess the impact of scores in online questionnaires on the UML learning. In fact, we are evaluating whether using online activities along with regular classes helps to improve UML learning (measured by grades in face-to-face exams). In addition, students are instructed to first attend the regular classes before accessing online content. The background of students is not controlled in this study, but empirical evidences show that students have similar average background knowledge over the years.

Scope limitation and threat to cause and effect - Our results are restricted to students from UFMG and only for UML content in a specific Software Engineering course. We cannot generalize our results to other universities, but students of the same course tend to have similar profiles. Related to threat to cause and effect, there is the possibility of actually students' knowledge had impact the results of the exams, and not because they answered an online questionnaire. That is, good students can solve the exams without difficulty because they are good students, and not because they solved questionnaires. This way, it is clearly hard to identify what action generates the actual effect on the other.

6. Conclusions and Future Work

This study evaluates whether online activities, such as video lectures and online questionnaires, contribute to improve student exam grades. We assessed the grades of 193 students of 6 different semesters and divided them into three groups that had different types of online support, but with the same UML content. The first group had only face-to-face classes, the second group had face-to-face and video lectures and, the third group had face-to-face classes, video lectures, and online questionnaires. The online content is available in the Udemy platform for a Software Engineering course [Pereira et al., 2013].

Two main analyses were performed. First, we analyzed the student grades per semester and, then, we analyze the student grades for the same two questions in four

semesters. Based on the results of this work, we conclude that the use of a platform to provide online content contributes to improving exam grades in questions related to UML. For the third group, that had access to online questionnaires and video lectures, the difference of student grades were about 20% compared to students that did not have access to online content.

Moreover, for two exam questions that repeated over four semesters, the grades increased by up to 20% for students who had access to video lectures and online questionnaires, compared with students who did not have access to online support. In the analysis per year, students who accessed only video lectures (Group B) had minor improvement in performance (1% to 8%) compared to the group who did not access any online content (Group A). On the other hand, students who accessed only video lectures (Group B) had a performance significantly lower (more than 10%) than students who accessed videos and online questionnaires (Group C). However, according to the analysis of repeated questions (Section 4.3), the use of video lectures contributes to increase of grades depending on the type of question. In general, this result shows that the use of other sources of study, such as video lectures and online questionnaires, help students as they study for a longer period of time than just in regular classes.

Our results also suggest that students perform better when exams have more than one question related to the same topic (UML, in our case). This fact helps students that know some specific content because with more questions they have more chance to answer one question related to a specific content they know. We believe that when the students have to do one activity in the online platform (such as questionnaires), the chance to access other content, such as video lectures, is higher than when they have only video lectures. Probably, they do not access the platform when they have only the video lectures, such as in the year of 2013. On the other hand, when online questions are available, students accessed the platform.

For future work, it is interesting to analyze online questionnaires in the context of online educational games and if games contribute to student learning of UML-related content. We also aim to further analysis whether other topics of Software Engineering can be supported from the use of online learning platforms. Furthermore, we can correlate other learning units, such as the performance in online questionnaires with performance on exam questions related to UML content.

Acknowledgements

This work was partially supported by CAPES, CNPq (grant 485907/2013-5), and FAPEMIG (grant PPM-00382-14).

References

Ahmadi, N. and Jazayeri, M. (2014) “Analyzing the Learning Process in Online Educational Game Design: A Case Study”. In 23rd Australian Software Engineering Conference (ASWEC), pp. 84-93.

Booch, G., Rumbaugh, J. and Jacobson, I. (2005) “The Unified Modeling Language User Guide”, 2 edition. Addison Wesley.

Dasarathy, B., Sullivan, K., Schmidt, D. C., Fisher, D. H. and Porter, A. (2014) “The Past, Present, and Future of MOOCs and their Relevance to Software Engineering”. In Proceedings of the on Future of Software Engineering, pp. 212-224.

Figueiredo, E., Lobato, C., Dias, K., Leite, J. and Lucena, C. (2007) “Um Jogo para o Ensino de Engenharia de Software centrado na Perspectiva de Evolução”. Workshop de Educação em Informática (WEI), pp. 37-46.

Figueiredo, E., Pereira, J., Garcia, L. and Lourdes, L. (2014) “On the Evaluation of an Open Software Engineering Course”. In proceedings of the 44th Annual Frontiers in Education Conference (FIE). Madrid, Spain.

Fox, A. and Patterson, D. (2012) “Crossing the Software Education Chasm”. Communications of the ACM, 55(5), 44–49.

Garcia, L., Martins, I., Ferreira, L. and Figueiredo, E. (2014) “Avaliação por Meio de Questionários de um Curso Online para Engenharia de Software”. In proceedings of the Fórum de Eduação em Engenharia de Software (FEES).

Liyanagunawardena, T. R., Adams, A. A. and Williams, S. A. (2012) “MOOCs: A Systematic Study of the Published Literature 2008-2012”. The International Review of Research in Open and Distance Learning.

Meirelles, L., Peixoto, D., Monsalve, E., Figueiredo, E., Werneck, V., Leite, J. C. and Pádua, C. (2011) “Uso de Jogos para o Ensino de Engenharia de Software”. In proceedings of Fórum de Educação em Engenharia de Software, São Paulo, Brasil.

Papaspyrou, N., Retalis, S., Efremidis, S., Barlas, G. and Skordalakis, E. (1999). “Web-based Teaching in Software Engineering”. Advances in Eng. Soft., 30(12), 901-906.

Pereira, J., Garcia, L. and Figueiredo E. (2013) “Proposta e Avaliação de Educação Aberta para Engenharia de Software”. In: Proceedings of Fórum de Educação em Engenharia de Software (FEES), Brasilia.

Potter, H. and Schots, M. (2011) “InspectorX: Um Jogo para o Aprendizado em Inspeção de Software”. In proceedings of Fórum de Educação em Engenharia de Software (FEES).

Schmidt, D. C. and McCormick, Z. (2013) “Producing and Delivering a Coursera MOOC on Pattern-Oriented Software Architecture for Concurrent and Networked Software”. In proceedings of the International Conference on Systems, Programming, Languages and Applications (SPLASH). Indianapolis.

Software as a Service – SaaS (2015). Accessed in 05/2015: http://www.edx.org/course/uc-berkeley/cs169-1x/software-service/691.

Sommerville, I. (2010) “Software Engineering”, 9 edition. Pearson.

Xie, T., Tillmann, N., Halleux, J. (2013) “Educational Software Engineering: Where Software Engineering, Education, and Gaming Meet”. In proceedings of the 1st International Workshop on Games and Software Engineering (GAS), pp. 36-39.

Ye, E., Liu, C., and Polack-Wahl, J. A. (2007). “Enhancing Software Engineering Education using Teaching Aids in 3-D Online Virtual Worlds”. Frontiers in Education (FIE).

Card Project Pro:
A Serious Card Game to Motivate Project Management

Willian Almeida Rodrigues[1], Windson Viana[1], Luís Fernando Maia Santos Silva[1], Fernando Antonio Mota Trinta[1], Jackson Gomes de Souza[2]

[1]Group of Computer Network, Software Enginer and Systems
Federal University of Ceará (UFC)
Av. Mister Hull Campus do Pici, Bloco 942-A CEP: 60455-760 - Fortaleza - CE - Brazil

[2]Information System and Computer Science
Centro Universitário Luterano de Palmas (CEULP/ULBRA)
Avenida Teotônio Segurado 1501 Sul CEP 77.019-900 - Palmas – TO – Brazil

[1]{willianrodrigues, windson, luissantos, fernandotrinta}
@great.ufc.br, [2] jgomes@ceulp.edu.br

Abstract. *Educational games are effective for the development of players' cognitive skills, especially, to improve abilities in solving complex problems. In this context, this paper presents the Card Project Pro (CPP), a card game that uses techniques of project management. Games structure, rules, and mechanics are based on the Guide of PMBoK, a guide to best practices in project management. The main goal of the CPP is to stimulate experience in simulated projects. The process of game development was done iteratively. A series of the game tests were implemented and the results were used to improve and change the game. User Evaluation indicates a better understanding and acceptance of the game, however, it also shows CPP is more suited as a complement for project management theory lectures.*

1. Introduction

Project Management is known as an activity that can be successfully performed through many approaches, such as PMBoK, Rational Unified Process, Extreme Programming, etc [Fitsilis 2008]. However, there are numerous common challenges faced by organizations during project management. Therefore, a compilation of best practices applied by great organizations is necessary.

The PMBoK (*Project Management Body of Knowledge*) [Project Management Institute 2014] emerged from the need for sharing such good practices, and is now composed by 47 processes divided among 10 areas of knowledge. The PMBoK works as tools to reduce the fuzziness of ongoing project developments, while improving their success rate on deadlines, cost reduction and quality.The use of project management techniques does not ensure the success of project development. Nevertheless, the PMBoK states, “a growing acceptance of project management indicates that it may have a notorious impact on the success of a project” [Project Management Institute 2014]. The majority of failures on software projects is due to misleading of the aspects presented in the PMBoK guide, and the lack of experience and expertise on project management can be seen as one of the main causes [Bomfin and Hastenreiter 2012] [Project Management Institute 2014] .

Learning project management demands practical experience. However, it is a risk for any organization to use ongoing projects as test cases for students and professionals without proper background. The previously mentioned situation raises the following question: How could students and professionals learn and practice project management techniques without offering risks to companies? Brom claims that games can be used as a motivational technique to ease teaching and boost the learning process hence being an outstanding approach for project management training [Brom et al. 2010].

Actually, there are games developed for numerous domains, such as public policy, defence, training, teaching, industry, and medical applications [Zyda 2005]. Games designed for purposes other than entertainment, enjoyment or fun are called "serious games" [Chen et al 2006]. Other authors present serious games as any form of game that has been developed with the intention to be more than entertainment [Ritterfeld et al. 2009]. Michael & Chen (2006) state games have the ability to grab full attention from players for a long period of time. This ability has started to draw the attention of researchers and teachers, since it has the potential to transcend entertainment, thus being used for educational purposes [Bittencourt 2005][Tarouco et al. 2004]. Ludic activities can be used as an approach to improve the learning process due to reducing the psychological pressure of the player through games. For instance, the player can assume the role of a project manager on a game that simulates ordinary situations to most project management companies, while making the player feel safer to determine the decisions for the project.

Games can also be used as ludic tools for improving learning experience, because of its ability to transfer concepts and knowledge. Salen & Zimmerman (2003) pointed out that the experience acquired on games, also known as meaningful ludic interaction, can be seen as "the way actions of a game result in its closure and meanings". Researchers propose the use of games for practicing project management techniques. Some works, such as Welcome to SCRUMIA [Wamgemheim et al.], [Rausis and Soares 2011] and Project Management Master [Software Quality Group], exploited analogical games for this purpose or used simple electronic games along with minor concepts of project management. A critical challenge when developing a game regards the design phase, since [Husain 2011] states that "too often we see examples of games produced by people in one particular field without even consulting the main end-user, seemingly missing the point."

Following this trend, in this paper, we present the Card Project Pro (CPP) serious game. CPP is a card game where the player has to plan, execute, monitor, control, and close projects, which uses PMBoK as basis to simulate the environment of a software development project. Our game was developed using an incremental design methodology, aiming to collect and evaluate feedback from end-users along the game development. Nevertheless, the CPP can be used to simulate other project management methodologies since its conception considered a generic software development project. Additionally, it can be adapted to be used by other areas, such as Civil Engineering and Production Engineering.

This paper is organized as follows: Section 2 shows related works. Section 3 presents the game, the development methodology, its mechanics was and context of use. Section 4 presents the study conducted to verify its usefulness, while Section 5 discusses the results of this work, and finally, Section 6 presents conclusions and future work.

2. Related Work

There are some other analogical and digital games that propose to aid the learning of project management. For instance, Welcome to SCRUMIA [Wamgemheim, C. G. V, 2013] is a game based on the SCRUM methodology where the player has to execute a sprint on a hypothetical project by applying concepts and competencies onto particular activities. SCRUMIA was designed to be played by an expert public that includes students from universities, project management courses and people with basic knowledge of project management.

Another game devoted to project management is the Detective Game [Rausis and Soares 2011]. It states that a group of players is hired to investigate the causes behind the failure of the project of a website. During the game, each player receives a set of documents that includes the state report during the project execution, hence the players must inspect the monitoring phase, the project control and evaluate time and cost consuming to find errors in the project. For each fault correctly calculated, the players are awarded with points, so the winner is the group of players that achieve a higher score. Likewise SCRUMIA, the Detective Game requires players with previous knowledge on project management.

Finally, the PM Master [Software Quality Group] is a quiz game that folds many knowledge areas, thus declaring the winner, the player who answers more questions correctly. The purpose of the game is to boost the knowledge of project management of players, thus demanding experienced players and students.

The main difference between the Card Project Pro and the previously mentioned games is that the CPP offers a wider investigation of project management while the other approaches are focused on specific areas or management techniques. An important aspect of exploiting specific areas is that the game can reinforce details on the area of knowledge, while the CPP method simulates the process from the beginning to the end, thus providing a simulated experience close to a real project management environment.

3. Card Project Pro - The Game

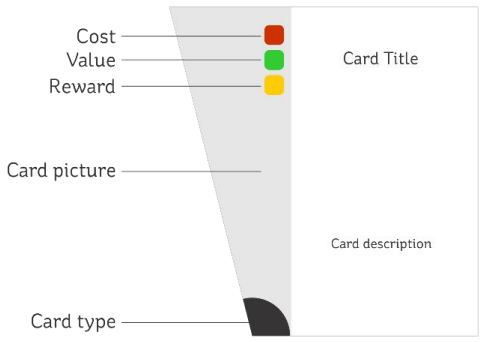


Figure 1. General outline of the cards

The game uses cards to represent resources in such a way that each card has always a value (currency in the game), and their management is done through card switching. There are also some cards representing costs, which are applied when the card is used (See Figure 1).

3.1. Mechanics

The game can be played by a single player in training mode, and by up to 4 players in production mode. In the last scenario, each player receives 1 mission card and 8 random cards each at the beginning of the game. The dynamics of the CPP happens in rounds that are finished when all players have made their decisions. By the end of the round, the results of actions are calculated and the time markers are set to measure the time consumed by the project and by player.

Round markers are used whenever projects are running on the game, to represent past rounds. Figure 2 shows that the project can last up to 2 rounds to be finished. Therefore, after adding 2 time markers, the project is labelled as delayed. In this case, the player is punished by losing his reward (See Project Card) and the personnel allocated to the project receive a round marker for each round played. The requirements worked on a project are calculated by multiplying the amount of round markers of Human Resource card and their skills. The project is considered completed if the requirements worked is equals or greater than the project requirements.

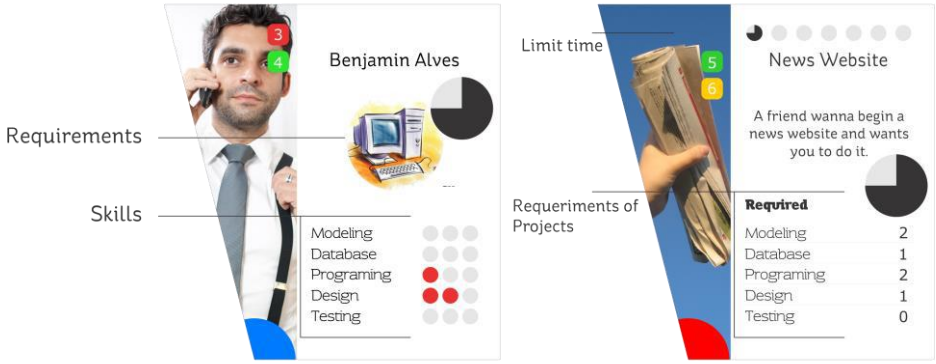


Figure 2. Round markers with Human Resource and Project card

We present the following example to illustrate a game round (as shown in Figure 3): A Human Resource card, in this case Benjamin Alves, is placed on the table together with a Resource card, exemplified by a Computer, so that the human uses the resource to work. Moreover, the player uses a Bonus card that increases 2 points on design skills, the Design Class. The Human Resource card will behave as a 4 points card if it has 2 points for design. The cost of the round is calculated by the sum of the costs from each card played, in this case, 3 points for the Human Resource card and 1 point for the Resource card are charged whenever the cards are on the table, while the cost of the Bonus card is billed when used. Finally, the ending cost of the round for this example is 6 points.

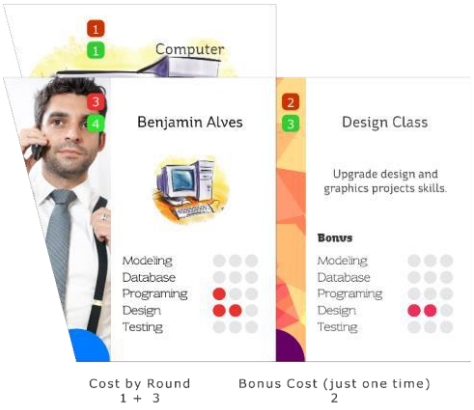


Figure 3. Human Resource on table

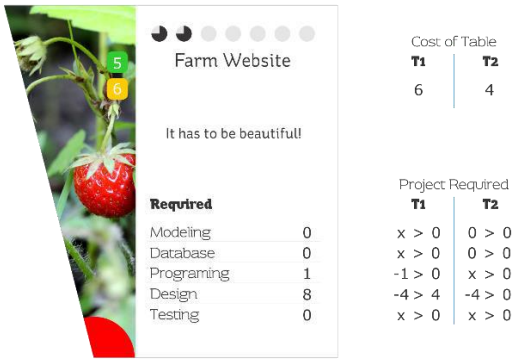


Figure 4. Projection by Round

In order to start a new project, a player has to put a Project card on the table, and every attribute of the card has to be fulfilled within 2 rounds. Hence, the project can be considered successfully finished. We exemplify this process using the Project card named Farm Website (See Figure 4). In this case, the project requires 1 point for programming and 8 points for design, so the costs and requirements have to be calculated by the end of the two next rounds (T1 and T2), as shown in Figure 4. By the end of the first round, the player has to pay 6 points for cost to receive the first round marker, and the project still needs 4 points for design which have to be completed in the next round. Whether the player cannot pay for the costs of the round, he has to discard the unaffordable cards on the table, thus possibly holding up the project. Each player is able to keep up to two projects on the table simultaneously, and whenever a project is completed on time, the player receives 6 points as a reward. Each point awarded has to be traded by a card from the stack. However, if a project exceeds the time it remains on the table, but it is not rewarded.

Additionally the players can perform the following actions during a round:

- Buy or trade cards once. The amount of cards that the player receives is equal to the sum of up to 2 cards he decides to dispose;
- Use Project, Human Resource, Resource, Action and/or Bonus cards;
- Manage resources on the table between Human Resource on the table;
- Manage personnel on table to projects;
- Before a new round is started the following items are checked:
    - All the costs on the table have to be paid, since human and other resources demand costs per round regardless their use. Additionally, bonuses are applied immediately and unaffordable cards have to be disposed.
    - Projects on table have their progress defined by the requirements accomplished on the round added to the overall progress of the project. Still, personnel and resources that were not afforded on the round will not score in progress.
    - Time markers have to be placed in each round for every personnel allocated to a project.

- Finally, projects that have all requirements completed are labelled as finished and whether there is no delay, the player receives a reward.

The game ends whenever a player achieves his mission - the entitled organizational goal. For instance, if a player has to finish 10 projects as his mission, he may proclaim himself the winner right after paying the costs for the last project.

The final performance of a player can be measured by the points he was rewarded and by the human, bonuses and other resources he used along the game, instead of only focusing on his final mission status. In conclusion, all the experience gained when playing can be used in the next matches. Despite of a change in the scenario, the management skills and knowledge acquired are the key for playing CPP.

3.2. CPP Cards

As the game simulates a software factory, the cards were defined to match this context, thus having characteristics that simulate situations commonly faced on such an environment. For instance, attributes like modelling, database, programming, design and testing were chosen as requirements to be fulfilled and resources, such as extra time and the computer were set as common supplies for performing activities. The main purpose of the game is to complete one of the 15 mission cards distributed to each player, using the other cards and their characteristics. Below, we present the cards and detail its attributes.

Human Resource Cards: They present different skills and attributes, which are used to achieve requirements described on a project card (See Figure 2 for details).

Project Cards: Since they represent the focus of the game to the player, their content shows each project requirement and goals to be pursued (Figure 2 shows an example of project card). Each project card has a set of requirements to be accomplished during the game and a time limit that has to be observed.

Bonus Card: This type of card (Shown in Figure 7) is used to improve Human Resource skills, and each card affects a particular set of attributes.

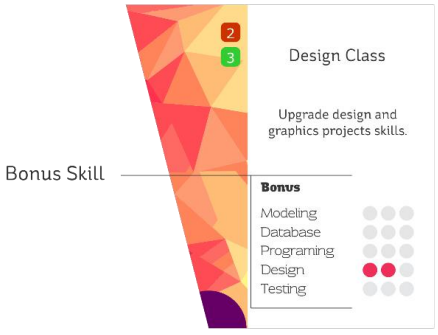


Figure 5. Bonus card

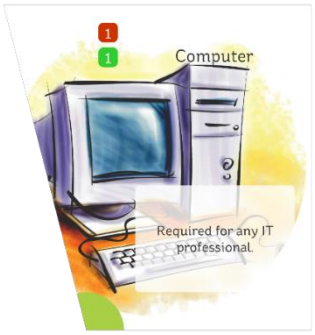


Figure 6. Resource Card

Resource Card: They are used to modify the role of other cards, and can be optional or mandatory. For example, the Human Resource card demands a Computer card to execute its job, thus being mandatory, while the extra time card is optional due to its ability to double production cards.

Action Cards: The game is unpredictable due to action cards, as they have the capacity to affect the game. For instance, the Vacation card showed in Figure 9 holds the cards on the table for a round, so the projects and personnel do not receive round markers and the round cost is set to zero.

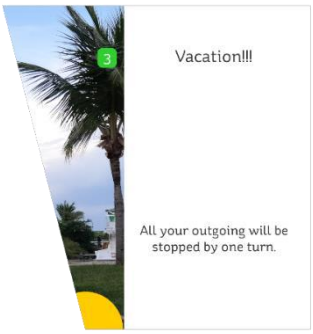


Figure 7. Action card

Figure 8. Mission card

Mission Cards: Those are cards that include the goals for the player, such as achieve a determinate amount of experience points on a specific attribute. It also finish a certain number of projects. Figure 10 depicts an example of mission card and its goals. In this card, the mission is to finalize 10 projects.

3.3. Design Principles

The game simulates a small company where the project manager is played by the user. During the game the player is in charge of the management of staff, resources, time, costs and projects. The game was designed applying concepts of project management, and the appliance of PMBoK techniques. The game behaviour can be presented by the following phases: *initialling*, *planning*, *monitoring* & *controlling* and *closing*.

On the **initialling phase** of the game, each player receives an *organizational goal*, which is represented by the Mission card, for instance, the user has to complete 3 projects successfully. An experienced player has a set knowledge from previous matches, so called *organizational process assets*. The player also has to acknowledge his competitors in a process known as *identify stakeholders* to later receive a new Project card and should be create a *develop project charter* on their minds, leading to the start of a **planning phase** for each card.

The **planning phase** is continuous on the game as real life. The players will always planning each action on the game: *scope*, *time*, *cost*, *quality*, *human resource*, *risk*, *procurement* and *stakeholder*. It has a great impact on project management, and demands the player to check and organize his resources aiming to fulfil requirements and complete the project. Only after performing self planning, the player is allowed to move to the **execution phase**.

In the **execution phase**, the players have to, one by one, present their actions, which means that according to their round, a player has to start his projects and execute his plans, thus trying to meet his goals. There are numerous actions that can be performed, for example, *perform quality control* of ongoing processes; *Acquire project team* through using a new Human Resource cards; *Develop project team* by improving their attributes

(using Bonus cards); *Manage project team*, that entails staff firing and hiring; and *conduct procurements*, that requires gathering resources according to the demand. And *Manage stakeholder engagement* by interesting with the other players in the game.

After the execution phase, the monitoring and controlling phase comes, in which the player evaluates the time consumed, the requirements fulfilled on each project, and has to manage acquisitions to unfinished projects according to their demands. Moreover, the schedule control, control cost, control risks and control scope should be checked, and so, the performance report should be presented to the other players. In addition, the players must perform a procurement administration whenever the project requires it.

Finally, the **closing phase** happens once the requirements of a project are completed, thus multiplying the round markers by the skills of personnel attached to the finished project.

The game uses a set of 132 ordinary cards, 15 Mission cards and coins for time marking. The cards are divided into: 30 Human Resource cards; 30 Project cards; 20 Bonus cards; 34 Resource cards; and 18 Action cards;

3.4. Context

The CPP context is placed in scenarios inspired by software factories, where each player represents a different corporation, with its own organizational challenges and goals. Within this context, the players act as project managers, hence being responsible for allocation and management of workforce and resources along with projects are executed and completed.

The game creates an environment that simulates experiences present in a real project management routine. For instance, by demanding the project manager to control teams, resources, costs and time of projects running inside the game, the player will be able to recognize the need for personnel changes, the skills needed by a project, the timing to start a new project considering available resources, among others.

4. Game Evaluation

4.1. Users

The game was tested with 8 graduate students (master and PHD students in Computer Science), divided in two groups, each performing two test sessions. Most users were not tabletop players, except one who claimed to play tabletop games regularly. Moreover, the students had little or no practical experience in project management, therefore, their learning was measured considering each knowledge area of PMBoK.

4.2. Evaluation Methodology

The tests were supervised, and lasted 60 minutes. The supervisor could intervene in the game at any moment, to present rules and clarify doubts. Before and after each session, players were asked to answer a set of questions about some aspects of the game:

- Pre evaluation: Evaluate the user profile as a player and project manager;
- Post evaluation: Measure the game motivation and user experience. Additionally, provides a new evaluation of user profile as a project manager. This approach is based on (Savi 2011), which has conducted a work on evaluation of educational games.

The Likert Scale was used in all questions, and presented the following options for answer: strongly disagree; disagree; neutral; agree; and strongly agree.

4.3. Feedback

The results were gathered through the feedback provided by players after answering the questions. A comparison regarding the project management skills before and after the tests was conducted to evaluate whether players had earned positive experience of project management. The post evaluation was divided into three groups: motivation, user experience, and learning. Motivation evaluated how players felt satisfied and confident about game results and actions. Additionally, the game relevance and the ability to hold player's attention were measured in this phase. User experience measures whether the game is fun, challenging, and immersive and promotes social interaction between players. The learning evaluation refers to the capacity of the game to teach and contribute to increase the learning skills of players.

5. Results

After the testing, most players reported that CPP encompasses the majority of concepts and activities presented in project management lessons, especially the time and cost management. Moreover, players also stated that, despite some confusion, they were capable of playing and understanding the game.

A comparison (See Figure 11) between manager's profiles showed that CPP is capable of maximizing the understanding regarding knowledge areas of PMBoK. Likerty Scale is 1 for "strongly disagree" to 5 for "strongly agree" for the item "After you had been played the game, how many you understand about demands of knowledge areas of Project Managements listed below?". After the trials, reports on this field changed from neutral (scale 3), in the best case, to agree (scale 4), in the worst case. The chart below **shows** this change in evaluation.

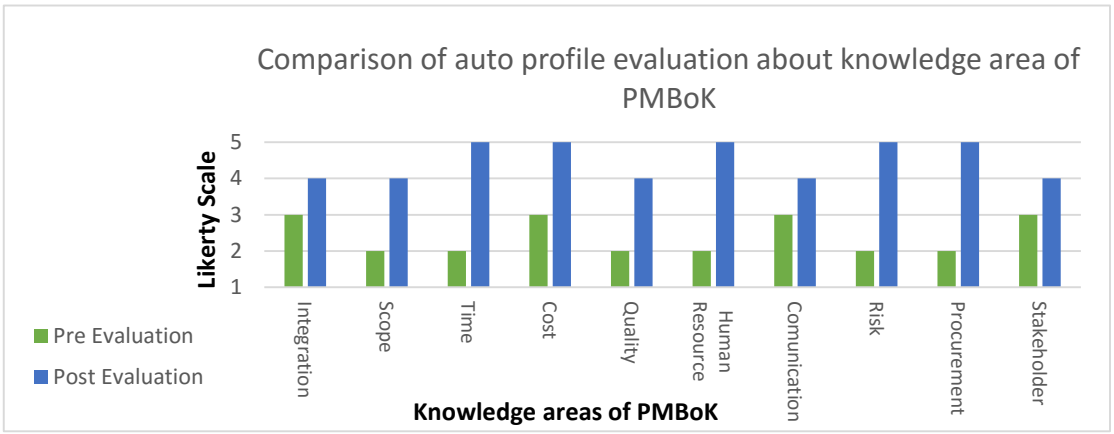


Figure 9. Comparison of auto profile pre and post evaluation

Reports also showed that most players felt motivated to play Card Project Pro, using their prior knowledge when playing (As shown in Figure 12). Thus, the game can be used as a tool to stimulate project management learning. However, it does not dim the relevance of theoretical studies of Project Management, like PMBoK.

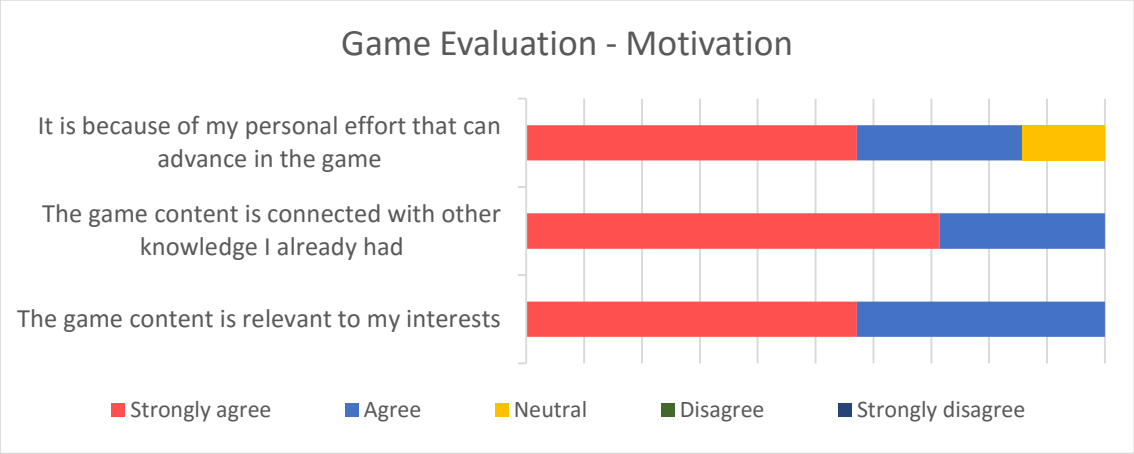


Figure 10. Motivation evaluation

User experience reports showed (In Figure 13) that eventually players win due to their skills in project management. However, it also presented that the game can promote fun and social interactions in the project management context.

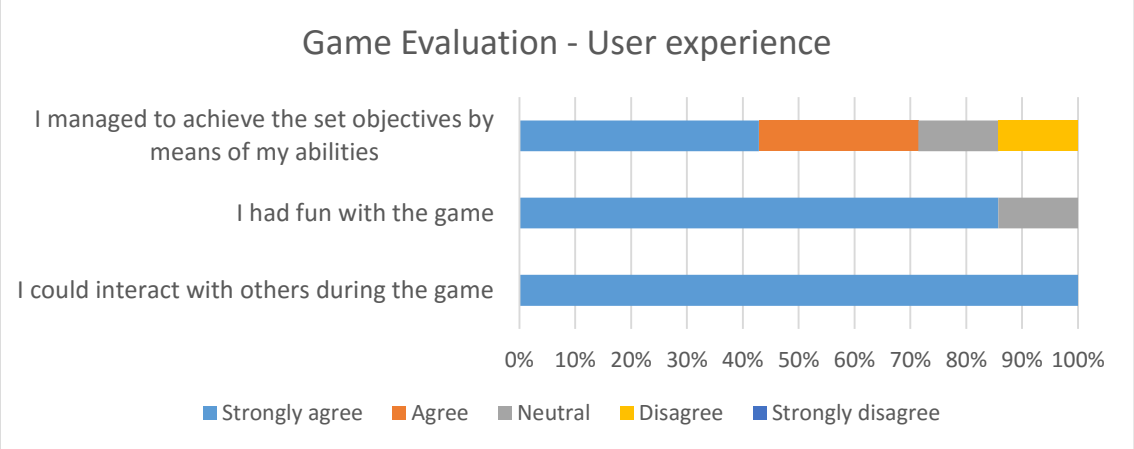


Figure 11. User experience evaluation

The learning report showed the game can be used to improve the practice of project management. The players acquired knowledge on areas of PMBoK whenever they were used in the game (As shown in Figure 14).

Among the threats to the validity, we can mention the lack of a test about real knowledge in Project Management of the volunteers. The relationship between the volunteers and the evaluator can have an impact on the results changing the evaluation results in the questionnaires. Also, further tests must be carried out for more accurate results.

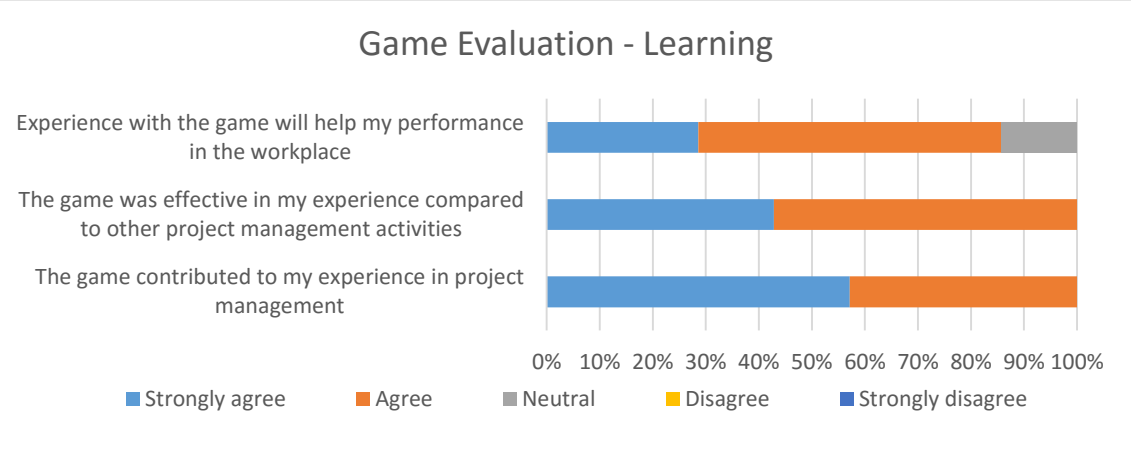


Figure 12. Learning evaluation

6. Conclusion and Future Work

Despite the importance of learning methodologies and techniques of project management, the practicing is fundamental to consolidate the concepts and provide the necessary experience on how to successfully manage projects. The lack of practical knowledge and maturity may lead to mistakes and uncertainty when conducting decisions, thus offering greater risks to projects. Besides, the experienced a project manager is, the greater his ability to foresee issues, overcome challenges and makes safer decisions. The CPP approach focuses on providing essential practical experience on project management through a simulation based on actual cases, thus offering an entertaining, accessible and risk free method to reach background on the field.

The game was designed to aid learners of project management techniques through linking the knowledge acquired during classes with training and simulations. In addition, companies may also use the CPP for coaching, to improve their management of workforce and during recruitment, to evaluate behaviour, skills and project management expertise of job applicants. The interactive design and initial user evaluation show a good acceptance by the players of the game, and indicate the viability to simulate a PMBOK experience by playing a card game.

The CPP game presents some drawbacks in its mechanics and project management techniques, and evaluation should be more explored. However, the ultimate goals of the game are to both easily present all the processes of PMBOK and provide players the necessary experience to cope with risks when managing projects.

As a future work, a mobile application of CPP is being planned to augment the experiences of playing a card game. We are also designing new features and contexts, which shall explore civil engineering and logistics domains. We will provide cards and resources that are suited to those environments. For further information on the Card Project Pro, a website is available on the link http://cpp.great.ufc.br/.

Acknowledge

Authors acknowledge the support of FUNCAP master student's scholarship (2014-2016).

References

Bittencourt, J.R. (2005) “Promovendo a Ludicidade Através de Jogos Livres”. In Anais do XVI Simpósio Brasileiro de Informática na Educação – Minicursos, p. 43 – 63.

Bomfin, D. F., Nunes, P. C., & Hastenreiter, F. (2012) “Gerenciamento de Projetos Segundo o Guia PMBOK: Desafios para os Gestores”. Revista de Gestão de Projetos, 58-87.

Brom, C., Sisler, V. and Slavík, R. (2010) “Implementing digital game-based learning in schools: Augmented learning environment of 'Europe 2045'”. Multimedia Systems 16(1), pages 23-41.

Chen, S., Michael¸ D. (2006) “Serious Games: Games that Educate Train and Inform”, Thomson Course Technology PTR.

Fitsilis, P. (2008) “Advances in Computer and Information Sciences and Engineering”. pp 378-383. Springer Netherlands.

Husain, L. (2011) “Getting Serious about Math: Serious Game Design Framework & an Example of a Math Educational Game”, Unpublished master's thesis, Lund University, Lund, Swedish.

Project Management Institute. (2014) “A Guide to the Project Management Body of Knowledge”, 5th ed.

Salen, K., & Zimmerman, E. (2003) “Rules of play: Game Design Fundamentals”. MIT Press.

Rausis, B.Z., Soares, G.M. (2011) “Detective Game – What killed the project?”, http://www.gqs.ufsc.br/detective-game-what-killed-the-project/, June.

Ritterfeld, U., Shen, C., Wang, H., Nocera, L. and Wong, W.L. (2009) “Multimodality and Interactivity: Connecting properties of serious games with educational outcomes”. Cyberpsychology & Behavior 12(6), pages 691-697.

Savi, R., Gresse Von Wangenheim, C.; Borgatto, A. Um Modelo de Avaliação de Jogos Educacionais na Engenharia de Software. 25th Brazilian Symposium on Software Engineering (SBES)/São Paulo/Brazil, 2011.

Software Quality Group. “Project Management Master”. http://www.gqs.ufsc.br/pm-master/, June.

Tarouco, L. M., Roland, L. C., Fabre, M.-C. J., & Konrath, M. P. (2004) “Jogos educacionais”.

Wamgemheim, C. G. V., Savi, R., Borgatto, A. F. “SCRUMIA – An Educational Game for Teaching SCRUM in Computing Courses”. Journal of Systems and Software, to appear 2013.

Zyda, M. (2005) "From Visual Simulation to Virtual Reality to Games", IEEE Computer, vol. 38, no.9. pp.25-32.

Uma Metodologia para o Ensino Teórico e Prático da Engenharia de Software

Rossana Maria de Castro Andrade[1], Ismayle de Sousa Santos[2], Italo Linhares Araújo[2], Rainara Maia Carvalho[2]

Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat)
Mestrado e Doutorado em Ciência da Computação (MDCC)
Universidade Federal do Ceará (UFC)
Fortaleza– Ceará – Brasil

{rossana,ismaylesantos,italoaraujo,rainaracarvalho}@great.ufc.br

Abstract. *The Software Engineering (SE) course is related to all aspects of software production. Due to the broad scope of this course, it is a challenge to cover all the necessary practical and theoretical content, and also keep the students constantly motivated. This paper presents a theoretical and practical teaching methodology adopted in SE course from the Department of Computing of the Federal University of Ceará (UFC) which seeks to overcome this challenge. An evaluation of this methodology was conducted through surveys and the results are promising. Thus, it is believed that the proposed methodology facilitates the understanding of practical and theoretical concepts while also motivates students to learn about SE.*

Resumo. *A disciplina de Engenharia de Software (ES) está relacionada com todos os aspectos da produção do software. Dada a abrangência dessa disciplina, é um desafio englobar toda a teoria necessária, exercitar na prática e ao mesmo tempo manter os alunos motivados. Este artigo apresenta uma metodologia de ensino teórico-prática adotada na disciplina de ES do Departamento de Computação da Universidade Federal do Ceará (UFC) que procura lidar com esse desafio. Uma avaliação da metodologia foi conduzida por meio de surveys e os resultados são promissores. Sendo assim, acredita-se que a metodologia proposta facilita o entendimento dos conceitos teóricos ao mesmo tempo que motiva os alunos a querer aprender mais sobre ES.*

1. Introdução

Segundo [Sommerville 2011], a Engenharia de Software é uma disciplina cujo foco está em todos os aspectos da produção de software, desde a especificação dos requisitos até a sua manutenção. Tal disciplina busca tornar sistemático o desenvolvimento do software através do ensino de métodos e técnicas que objetivam entregar o produto com qualidade, dentro do prazo e orçamento.

Sabendo que cada vez mais a sociedade depende de softwares e que a era dos *smartphones* tornou essa dependência ainda maior, a demanda por profissionais que

[1] Bolsista do CNPq de Produtividade DT-2 com o número de processo 314021/2009-4
[2] Bolsista de Doutorado da Capes

exercitem os métodos e técnicas de ES também aumentou. Por isso, o ensino de ES é essencial nos cursos de computação. A importância da ES é tanta que já existe um curso de graduação em Engenharia de Software [ACM 2004].

Já é conhecido que o ensino da teoria sozinha não estimula um aprendizado profundo [Silva e Vasconcelos 2014], e que esta deve estar alinhada com a prática. Isso é ainda mais importante no ensino de ES, pois essa é uma disciplina extremamente prática e os alunos precisam estar preparados para a realidade do mercado, que é dinâmico e envolve difíceis tomadas de decisão.

Em um trabalho anterior [Andrade et al. 2008] foi apresentado uma proposta de uma metodologia de ensino teórico-prático de Engenharia de Software, bem como uma avaliação desta proposta por meio das notas dos alunos. Essa proposta vem sendo aplicada e aprimorada na disciplina de ES do Departamento de Computação da UFC desde 2005.

Dessa forma, o objetivo deste artigo é comparar a metodologia de ensino de ES consolidada ao longo dos anos, e que foi aplicada no semestre 2015.1, com a metodologia apresentada anteriormente [Andrade et al. 2008]. Além disso, são apresentados os resultados de *surveys* [Wohlin et al. 2000] aplicados com os alunos com o objetivo de avaliar o impacto das novas mudanças.

O restante deste artigo está organizado como se segue. Na Seção 2 são apresentados os trabalhos relacionados. Na Seção 3 descrevemos e justificamos as mudanças feitas na metodologia apresentada por Andrade et al. (2008). Na Seção 4 apresentamos os resultados das avaliações conduzidas durante o semestre letivo de 2015.1. Por fim, na Seção 5 são apresentadas as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Silva e Vasconcelos (2014) propõem a utilização de um ambiente integrado desenvolvido para automatizar os processos de ES, como um apoio ao ensino-aprendizagem da área. A ênfase do artigo está no uso de duas ferramentas do ambiente relacionadas aos processos de Gerência de Projetos e de Gerência de Requisitos. No entanto, o trabalho não apresentou dados que mostrem a melhoria do desempenho dos alunos. O diferencial deste artigo está em apresentar toda a metodologia aplicada no ensino da ES na UFC.

O trabalho de Garcia et al (2014) apresenta uma avaliação do desempenho de alunos em um curso on-line de Engenharia de software da Universidade Federal de Minas Gerais que apoia um curso presencial de ementa equivalente. Os resultados mostraram que assistir às aulas em vídeo do curso on-line não foi um fator determinante para o desempenho dos alunos. Diferentemente do trabalho de Garcia et al., o presente trabalho avalia não somente o desempenho do aluno, mas também as práticas utilizadas na metodologia de ensino.

Billa e Cera (2012) propõem o ensino da teoria em um curso de Engenharia de Software por meio de uma abordagem de ensino-aprendizagem baseada em Problemas. Nessa abordagem equipes de alunos propõem um sistema computacional que solucione um problema real praticando assim os conceitos de ES e a organização do trabalho em equipe. Na metodologia apresentada neste artigo os alunos também desenvolvem um

aplicativo. Contudo, ela também prescreve outras práticas, como dinâmicas e apresentações sobre a carreira. Além disso, o foco deste artigo é uma disciplina de ES e não um curso de Engenharia de Software, como é o caso do trabalho de Billa e Cera.

3. Metodologia de Ensino de ES

Nesta Seção são apresentados os detalhes da metodologia aplicada na disciplina de ES do Departamento de Computação da UFC. A disciplina de ES é obrigatória, possui quatro créditos, totalizando sessenta horas-aula por semestre e as turmas têm em média quarenta alunos.

3.1. Tipos de Atividades

Na proposta inicial da metodologia, apresentada por Andrade et al. (2008), as atividades eram divididas em:

- *Atividades intra-classe*: atividades que são realizadas em sala de aula e englobam aulas expositivas, exercícios e discussões em grupo.
- *Atividades extra-classe*: atividades desenvolvidas fora do espaço físico da sala de aula. Elas têm como objetivo o desenvolvimento de um projeto de software, obtendo como produto final uma aplicação para dispositivos móveis.

Com o tempo percebeu-se, no entanto, a necessidade de atividades que iniciem em sala de aula, mas que pudessem ser concluídas fora dela. Logo, além de usar atividades *intra e extra-classe*, a metodologia atual aplica *atividades híbridas,* que iniciam em classe, mas que podem ser finalizadas fora da sala de aula, respeitando um prazo pré-estabelecido. Por exemplo, em uma aula de UML, os alunos iniciaram no laboratório a especificação de diagramas UML e tinham até a meia-noite do mesmo dia para entregar as versões finais.

3.2. Ensino da Teoria

Na metodologia apresentada por Andrade et al. (2008) eram previstas apenas aulas expositivas para o ensino da teoria relacionada a disciplina de ES. Contudo, dado a quantidade de informações repassadas durantes as aulas, percebeu-se que somente aulas expositivas eram ineficientes para motivar o aluno a ficar atento durante a aula.

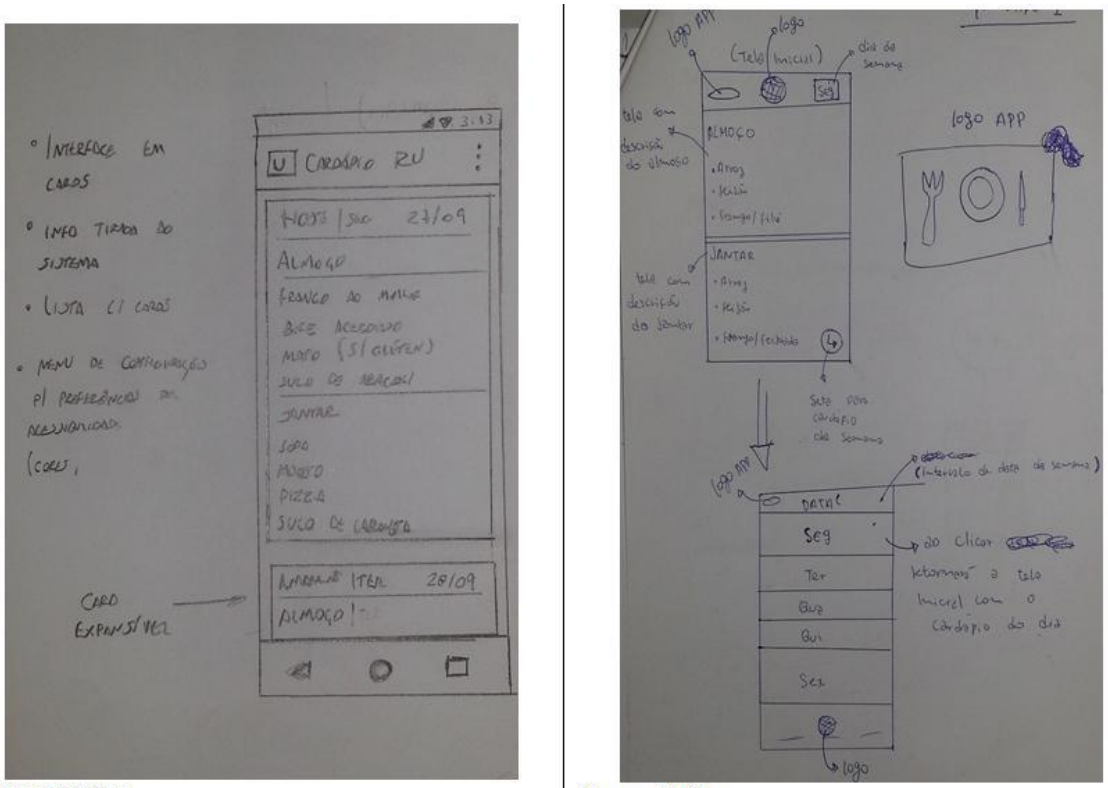
Sendo assim, na metodologia atual, além de uma maior quantidade de exercícios *intra-classe*, *extra-classe* e *híbridos* (iniciados em sala e terminados fora dela), passou-se a conduzir dinâmicas em sala de aula com intuito de despertar o interesse do aluno na disciplina.

No semestre 2015.1 foi aplicada a dinâmica “O Cliente Mandou”, criada pelos autores com base em uma brincadeira infantil bastante conhecida chamada “O Mestre Mandou”. A seguir é discriminado o passo-a-passo desta dinâmica:

- **Passo 01**: As equipes devem ser montadas (sugestão: 6 pessoas no máximo);
- **Passo 02**: Cada equipe deve nomear o gerente, o analista de requisitos, o analista de teste e os desenvolvedores;

- **Passo 03**: Definir uma pessoa fora das equipes como cliente (e.g. professora ou monitor);
- **Passo 04**: Cada analista de requisitos das equipes tem dois minutos para se reunir com o cliente e coletar os requisitos;
- **Passo 05**: Os analistas de requisitos devem então repassar para a equipe os requisitos coletados;
- **Passo 06**: Os desenvolvedores devem desenhar o produto do cliente; os analistas de testes devem especificar em alto nível os testes que serão feitos; e o gerente deve estimar o tempo e o custo de desenvolvimento. Tudo isso dentro de um limite de tempo de 10 minutos;
- **Passo 07**: Confrontar os resultados de cada equipe com o produto esperado.

A Figura 1 apresenta o resultado de duas equipes com essa dinâmica. Com essa atividade foi possível então observar os diferentes pontos de vista dos alunos para um mesmo problema em termos de funcionalidades necessárias, tempo de desenvolvimento, preço e testes a serem executados. A comparação entre os diferentes resultados dos alunos durante a aula também proporcionou aos alunos perceberam o quão é importante a Engenharia de Software para o desenvolvimento de software.



Tempo: 39 dias
Preço: R$ 11.500,00
Testes: estresse, unitário, sistema, usabilidade, acessibilidade
(A)

Tempo: 20 dias
Preço: R$ 2.800,00
Testes: aceitação e sistema
(B)

Figura 1. Resultado da dinâmica o "Cliente Mandou" de duas equipes diferentes

3.3 Ensino na Prática

Na proposta de Andrade et al. (2008) para realizar as atividades do projeto de software os alunos eram organizados em equipes de no máximo quatro alunos constituindo uma equipe de projeto. Dentro de cada equipe, um dos alunos exercia o papel de gerente de projeto e dos demais integrantes representavam os demais papéis envolvidos no desenvolvimento de software (e.g. engenheiro de requisitos, analista de testes). Desta forma, os alunos exercitavam habilidades como liderança, cooperação e articulação.

Para auxiliar as atividades, cada equipe contava com o auxílio de um monitor, sendo que para cada monitor eram alocadas três ou quatro equipes. Além disso, a escolha do modelo de processo de desenvolvimento e da ferramenta de desenvolvimento era a critério da equipe. Para facilitar o uso das tecnologias envolvidas, aulas extras eram ministradas.

Por fim, cabe ressaltar que independente do modelo de processo escolhido, cada equipe tinha que entregar um conjunto de artefatos básicos com marcos pré-estabelecidos pelo professor. Tais artefatos foram apresentados e explicados durante as aulas teóricas. A Tabela 1 sumariza este conjunto de artefatos.

Tabela 1. Artefatos a ser entregues por cada equipe de projeto (adaptado de Andrade et al. (2008))

Plano de Projeto	Estabelece os recursos disponíveis, a estrutura analítica do projeto, os riscos e os mecanismos de acompanhamento
Cronograma	Define o prazo estimado para atingir cada marco e a alocação de recursos nas atividades
Estudo de Viabilidade	Define a viabilidade de implementação considerando restrições de tecnologia, custo, prazo e integração com outros sistemas
Especificação de Requisitos	Estabelece o que a aplicação deve implementar
Especificação de Casos de Uso	Detalha os requisitos da aplicação por meio das descrições dos casos de uso
Especificação de Arquitetura	Apresenta uma visão geral de alto nível da arquitetura prevista do sistema
Projeto da Aplicação	Envolve os diagramas de classes, sequência e de estados
Relatório de Padrões de Software	Identifica os padrões de projeto utilizados durante o desenvolvimento da aplicação
Código	Código-fonte da aplicação
Plano de Testes	Define o que e como vai ser testado, os tipos de testes a serem aplicados, prazos e os recursos alocados
Relatório de Testes	Registra os defeitos encontrados na aplicação
Apresentação	Apresentação aberta para os demais alunos resumindo os resultados obtidos com o desenvolvimento do projeto

Assim como na proposta inicial, a exigência de que o domínio da aplicação fosse o de aplicações móveis foi mantida neste semestre de 2015.1. Além disso, as aulas extras continuaram a ser ministradas. Contudo, ao longo dos anos percebeu-se que muitas equipes não concluíam os aplicativos e a principal razão eram dificuldades técnicas. Para resolver esse problema, as seguintes mudanças foram aplicadas:

- Passou-se a exigir que a linguagem de programação fosse Java na plataforma Android[3]. Essa decisão foi tomada devido ao conhecimento dos monitores nessa linguagem e plataforma, facilitando assim o auxílio dos monitores nas atividades do projeto;
- A ferramenta de gerência de projetos tinha que ser a Gantter[4], pois permitia o compartilhamento do planejamento definido pelos gerentes das equipes com os tutores responsáveis pelas mesmas;
- O tamanho da equipe passou a ser 6 com o intuito de simular uma empresa em ambiente real e melhor distribuir as atividades dentro da equipe.

Em relação aos artefatos produzidos pelas equipes, neste primeiro semestre de 2015 foi adicionado o termo de abertura contendo a assinatura da professora e dos monitores responsáveis por cada uma das empresas, os quais desempenhavam o papel de *stakeholders*. Dessa forma, o processo de desenvolvimento pôde ser simulado de maneira mais próxima da realidade encontrada no mercado.

3.4 Interação com os alunos

A interação com os alunos da disciplina também evoluiu com o passar do tempo. Na proposta inicial [Andrade et al. 2008] a interação entre alunos, monitores e professor era feita por meio de uma lista de discussão ou por reuniões presenciais. Além disso, todo o material da disciplina era disponibilizado em um site.

Durante o semestre 2015.1, além das interações apresentadas acima, duas mudanças foram feitas: (i) os alunos passaram a interagir também com profissionais graduados (analistas de testes, arquitetos de software e gerentes de projeto) que trabalhavam em projetos de desenvolvimento, pesquisa e extensão do laboratório GREat[5]; e (ii) os monitores passaram a se comunicar com os alunos também pelo *Facebook*[6], mantendo um contato mais próximo com os mesmos.

Com essas mudanças observou-se que alguns alunos se sentiram mais motivados a tirar dúvidas de implementação, pois eles gostavam de interagir com os profissionais já graduados para trocar experiência. Além disso, percebeu-se que a interação por meio de uma mídia social facilitou a comunicação com os alunos, visto que a maioria deles usava o *Facebook* com frequência.

3.5 Palestras sobre Carreira

Com o intuito de aproximar ainda mais os alunos da realidade fora da sala de aula, uma inovação neste semestre 2015.1 na metodologia de ensino foi que uma das primeiras aulas foi para apresentar os alunos aos cargos relacionados a Engenharia de Software. Nesta aula, os alunos visitaram um ambiente real de desenvolvimento de software e ouviram os relatos de profissionais graduados que exercitavam os seguintes papeis: gerente de projetos, analista de requisitos, analista de qualidade, analista de testes, arquiteto de software, design e desenvolvedor.

[3] https://www.android.com/
[4] http://www.gantter.com/
[5] http://www.great.ufc.br
[6] http://facebook.com/

3.6 Avaliações

O método de avaliação também foi alterado quando comparado com a metodologia proposta por Andrade et al. (2008). Neste semestre de 2015.1, a avaliação dos alunos foi feita por meio de uma média aritmética entre as notas de duas provas, listas de exercícios e de um projeto de desenvolvimento.

As provas valem de zero (0) a dez (10) e avaliam o conhecimento dos alunos quanto aos conteúdos ministrados na disciplina. Cabe ressaltar que o conteúdo ministrado nas aulas extras não era cobrado nas provas.

As listas de exercícios possuíam valores diferentes que somados poderiam valer até dez (10) pontos. Neste semestre, foram passadas cinco listas, sendo 4 com pontuação um (1) e uma valendo seis (6) pontos. O intuito dessas listas é abordar temas que não foram explorados nas provas.

Com relação a nota do projeto, esta é composta pela soma das notas de cada artefato. Tais notas são definidas em primeira instância pelo monitor que acompanhou de perto a equipe e depois são revisadas pelo professor.

3.7 Visão Geral das Mudanças

Com base no que foi apresentado nas subseções anteriores, a Tabela 2 apresenta uma visão geral das diferenças entre a metodologia inicialmente proposta [Andrade et al. 2008] e a metodologia atual.

Tabela 2. Evolução da metodologia de ensino de ES

Critério		**Metodologia apresentada em Andrade et al. 2008**	**Metodologia atual (aplicada em 2015.1)**
Tipos de Atividades	**Aulas Expositivas**	Intra-classe	Intra-classe
	Exercícios	Intra-classe e Extra-classe	Intra-classe, Extra-classe e Híbridas
	Relacionadas ao projeto de Software	Somente Extra-classe	Extra-classe e Híbridas
Ensino da Teoria	**Dinâmicas**	Nenhuma	"O Cliente Mandou"
	Aulas Expositivas	Aulas em sala de aula	Aulas em sala de aula
Ensino na Prática	**Domínio da aplicação**	Software Móvel	Software Móvel
	Ferramenta de gerência	Livre para a equipe escolher	Gantter
	Linguagem de Programação	Livre para a equipe escolher	Java (Android)
	Tamanho da equipe	Quatro *Equipe de Projeto*	Seis *Empresa de Software*
	Aulas extras	Sobre a linguagem, ferramentas e outros	Sobre a linguagem, ferramentas e outros

Interação	**Entre alunos, monitores e professor**	Reuniões presenciais, Lista de e-mail e site	Reuniões presenciais, Lista de e-mail, site e *Facebook*
	Entre alunos e profissionais graduados	Nenhuma	Sempre que solicitado pelo aluno
Palestras sobre a carreira	**Apresentação de um ambiente de desenvolvimento de software**	Nenhuma	Apresentação do ambiente de trabalho de uma equipe de desenvolvimento real
	Apresentação de cargos relacionados a ES	Nenhuma	Apresentação sobre os cargos relacionados a ES por profissionais graduados
Avaliações		Duas provas e um projeto final	Duas provas, listas de exercícios e um projeto final

4. Avaliação da Metodologia

Nesta seção são apresentados os resultados das avaliações feitas durante a disciplina por meio de questionários, utilizando formulários on-line para coleta de dados, e o desempenho dos alunos nos projetos de desenvolvimento de software.

4.1 Avaliação das palestras sobre carreira

Uma das mudanças na metodologia aplicada neste semestre 2015.1 foi uma aula destinada para apresentação dos cargos e palestras sobre a carreira relacionada à Engenharia de Software (ver Seção 3.5). O objetivo dessa aula era motivar o aluno a estudar o conteúdo que seria repassado na disciplina de ES.

As Figura 2 apresenta a opinião coletada dos alunos quanto a esta aula. De maneira geral, todos os alunos acharam que a aula foi interessante. Além disso, 89% dos alunos acharam que a aula foi motivadora. Logo, pelos resultados apresentados acredita-se que a aula atingiu o objetivo de motivar os alunos a se interessarem pela disciplina de Engenharia de Software. Por fim, as perguntas feitas pelos alunos durante esta aula corroboram com os indícios mostrados pelo questionário de que a aula foi interessante e motivadora para os alunos.

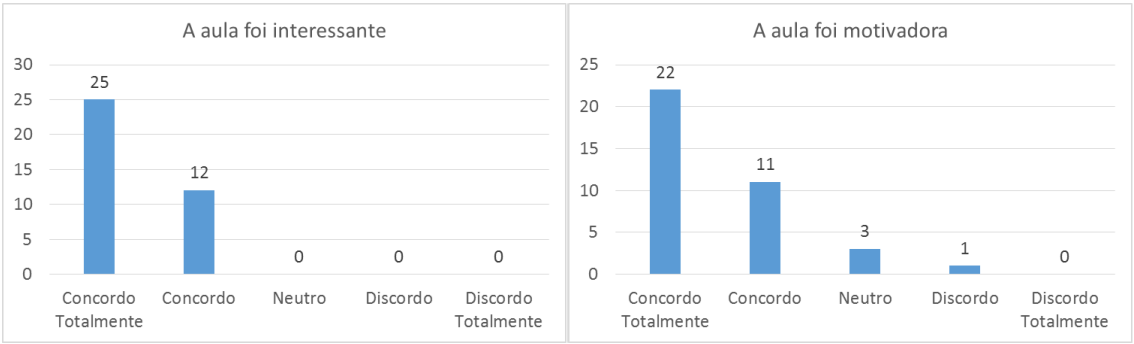


Figura 2. *Feedback* dos alunos sobre a aula de cargos e carreira

Ao fim da aula os alunos foram questionados sobre qual cargo eles se identificaram. A Figura 3 apresenta os resultados. Com base nesta figura foi possível constatar que os alunos tinham interesse nos diversos cargos apresentados na aula. Destacou-se então durante a aula, que o conhecimento necessário para exercer tais cargos provinha em parte da disciplina de Engenharia de Software, reforçando assim a importância da mesma. Além disso, a maioria dos alunos se identificou com o cargo de "desenvolvedor". Logo, o desenvolvimento de um software durante a disciplina vem de acordo com os interesses dos alunos.

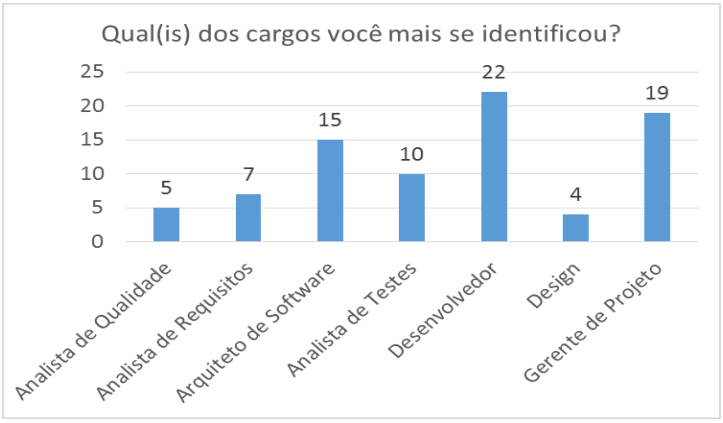


Figura 3. Resultado do questionário sobre o cargo que eles se identificaram

4.2 Avaliação da dinâmica "O Cliente Mandou"

Também foi coletado o *feedback* dos alunos quanto a dinâmica executada em sala. De acordo com os resultados apresentados na Figura 4 é possível notar que os alunos respondentes puderam com a dinâmica entender o quão é complexo a atividade de desenvolvimento de um software. Ela foi importante para que fosse ressaltada a necessidade dos documentos, modelos e processos apresentados durante a disciplina.

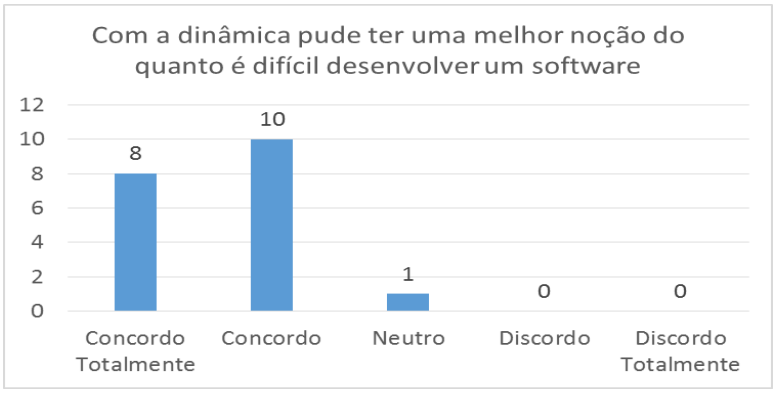


Figura 4. Visão dos alunos quanto a dificuldade do desenvolvimento de software após a dinâmica "O Cliente Mandou"

Além disso, os alunos foram questionados sobre a principal dificuldade deles na execução da dinâmica (ver Figura 5). As principais dificuldades foram estimar o custo e definir as funcionalidades. Essa coleta foi interessante porque foi possível identificar quais assuntos os alunos perceberam que tinham mais carência, e assim os próprios alunos mostraram interesse em saber como definir e gerenciar melhor os requisitos, bem

como melhor estimar o custo. Também se observou o interesse dos alunos em aprender como superar as demais dificuldades apresentadas na Figura 5.

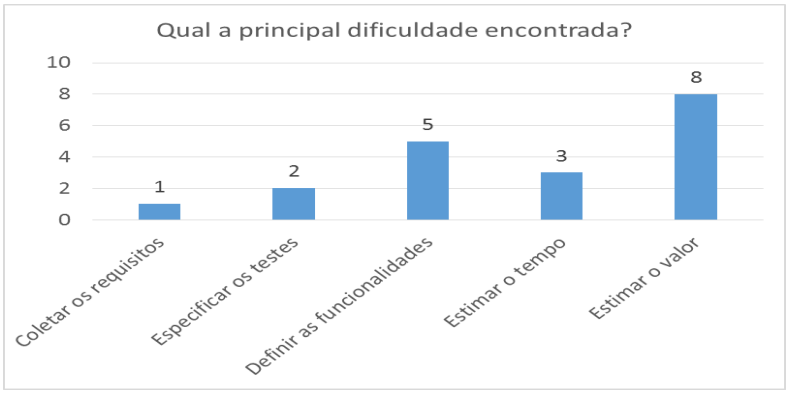


Figura 5. Principais dificuldades encontradas durante a dinâmica "O Cliente Mandou"

4.3 Avaliação da disciplina e atividades intra-classe

As últimas avaliações feitas na disciplina foram sobre se os alunos estavam gostando da disciplina e se eles estavam satisfeitos com as dinâmicas e atividades intra-classe.

A Figura 6 apresenta o *feedback* dos alunos quanto a disciplina. Esse questionário foi aplicado no meio do período letivo. Conforme os dados apresentados nessa figura, percebe-se que de maneira geral os alunos gostaram da metodologia que estava sendo aplicada. Apenas dois alunos não estavam gostando da disciplina, mas eles não deixaram sugestões ou críticas no formulário de coleta da opinão.

Com relação a avaliação das dinâmicas e atividades intra-classe, apresentada na Figura 7, a maioria dos alunos afirmou ter gostado das disciplinas e atividades intra-classe. Apenas 3 alunos marcaram como "neutro" a avaliação das dinâmicas e atividades intra-classe, mas apenas um deles justificou a escolha da opção "neutro" afirmando que essas atividades ocupavam muito tempo da aula e exigiam bastante esforço dos alunos, se tornando assim cansativas.

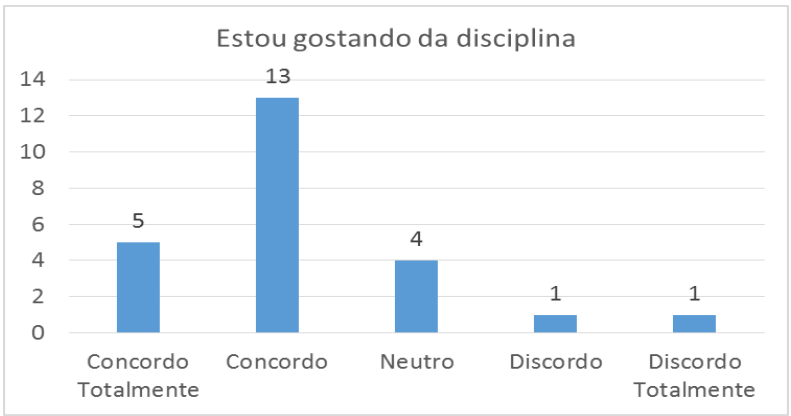


Figura 6. Avaliação geral da disciplina

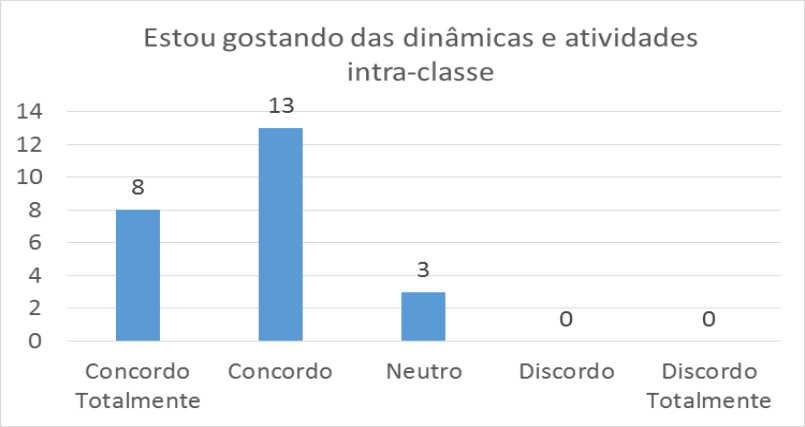


Figura 7. Avaliação das dinâmicas e atividades intra-classe

4.4 Desempenho nos projetos de desenvolvimento

A média geral das notas dos projetos neste semestre 2015.1 foi de 8.05, indicando que os alunos tiveram em geral um desempenho satisfatório no projeto. Para comparação, as médias das notas das provas dos alunos foram de 6,4 para a primeira prova e 5,4 para a segunda prova. Com isso, pode-se notar que os alunos se saíram melhor no projeto do que nas provas.

Com relação aos artefatos solicitados, todos foram entregues no prazo por todas as equipes. É interessante destacar que para aumentar o aprendizado em cima dos documentos exigidos no trabalho final (ver Seção 3), uma vez que eles eram entregues, tais documentos eram corrigidos e os alunos tinham a opção de devolver o documento corrigindo os problemas identificados. Para incentivar tal prática, caso os alunos corrigissem os documentos eles recebiam bônus adicional. Todas as equipes corrigiram todos os documentos.

Por fim, com relação a entrega final do produto, todas as equipes conseguiram entregar o produto pelo menos parcialmente, isto é, com alguns requisitos implementados. Algumas equipes inclusive surpreenderam pela competência na execução das atividades e pelo profissionalismo empregado. Acredita-se que as aulas extras, a presença constante dos monitores e a interação com os profissionais graduados tenha contribuído para esse resultado positivo. O aplicativo HelpMeCook [Novais et al. 2015] é um exemplo de um aplicativo desenvolvido durante o semestre letivo 2015.1 na disciplina de ES do Departamento de Computação da UFC.

5. Conclusões e Trabalhos Futuros

A Engenharia de Software é uma disciplina fundamental para a carreira do estudante de graduação em Computação, pois ela está relacionada com todos os aspectos da produção de software. Dado a quantidade de teoria envolvida, ensinar ES apenas com aulas expositivas tradicionais pode dificultar o aprendizado dos alunos.

Nesse cenário, uma proposta de metodologia foi apresentada em um trabalho anterior e melhorada ao longo dos anos. Esse artigo apresentou então a metodologia de ES aplicada no semestre de 2015.1 no Departamento de Computação da Universidade

Federal do Ceará. O diferencial desta metodologia é procurar passar a teoria (por meio de aulas expositivas, atividades e dinâmicas), exercitar na prática (por meio das atividades e do projeto de desenvolvimento de software) e motivar os alunos a se interessar por ES (por meio das dinâmicas, aula sobre carreira e palestra dos profissionais).

Para avaliar a metodologia aplicada, vários *questionárioss* foram passados aos alunos ao longo da disciplina. De modo geral, os resultados foram positivos e indicaram indícios de que a metodologia está adequada e condizente com o objetivo de envolver a teoria e prática no ensino de ES, ao mesmo tempo em que procura motivar os alunos a querer aprender o conteúdo passado durante a disciplina.

Como perspectivas de trabalhos futuros, pretende-se coletar diferentes medidas (e.g. frequência nas aulas e notas) para avaliar a metodologia aplicada sob outros aspectos. Além disso, alterações devem ser conduzidas (e.g. introdução de jogos educativos) e avaliadas em busca de uma melhoria continua da metodologia.

6. Agradecimentos

Agradecemos a colaboração do aluno de graduação Aldy Colares, da graduada Paula Caldas e da aluna de mestrado Andressa Bezerra pelo suporte nas atividades da monitoria da disciplina. Agradecemos também aos profissionais que fizeram apresentações na disciplina e tiraram dúvidas dos alunos.

Referências

ACM (2004). Curriculum guidelines for undergraduate degree programs in software engineering. Disponível em http:// http://sites.computer.org/ccse/.

Andrade, R. M. C.; Marinho, F. G.; Lelli, V.; Rocha, L. S. (2008) Uma proposta de Metodologia para o Ensino de Engenharia de Software. In: Fórum de Educação em Engenharia de Software – FEES.

Billa, C. Z.; Cera, M. C. (2012) Utilizando Resolução de Problemas para aproximar Teoria e prática na Engenharia de Software. In: Fórum de Educação em Engenharia de Software - FEES.

Garcia, L.; Martins, I.; Ferreira, L.; Figueiredo E. (2014) Avaliação por Meio de Questionários de um Curso Online para Engenharia de Software. In: Fórum de Educação em Engenharia de Software - FEES.

Novais, A. Herbster, C.; Melo, F.; Galas, K.; Fontenele, M.; Torres, T. (2015) HelpMeCook - Aplicativo desenvolvido durante a disciplina. Disponível em: http://www.helpmecook.com.br/

Silva, S. V.; Vasconcelos, A. P. P. (2014) Ambiente Integrado como Apoio ao Ensino da Engenharia de Software. In: Fórum de Educação em Engenharia de Software – FEES.

Sommerville, Ian. Engenharia de software. 9ª Edição. Pearson Education, 2011.

Wohlin, C.; Runeson, P.; Host, M.; Ohlsson, M.; Regnell, B.; Wesslen, (2000) A. Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers

Aplicação de *Design Thinking* em Disciplinas de Oficina de Desenvolvimento de Sistemas

Emanuel F. Coutinho, George Allan M. Gomes, Antonio José M. L. Júnior

Instituto Universidade Virtual – Universidade Federal do Ceará – Fortaleza – CE, Brasil

{emanuel,george,melojr}@virtual.ufc.br

Abstract. *Apply theory and practice is usually a difficult task to lead in undergraduate courses. Generally disciplines involving development projects are used as an attempt to minimize this difficulty. One way of conducting this kind of discipline is defining a development methodology to be applied. This paper aims to present an approach based on design thinking to develop products and services that involve software, aligned with the best practices of Software Engineering. An assessment of the approach was carried out with students of the discipline, concluding that its use contributed to improve communication, documentation and projects monitoring.*

Resumo. *Aplicar teoria e prática normalmente é uma tarefa difícil de se conduzir em cursos de graduação. Geralmente disciplinas que envolvem projetos de desenvolvimento são empregadas na tentativa de minimizar esta dificuldade. Uma maneira de se conduzir esse tipo de disciplina é definindo uma metodologia de desenvolvimento a ser aplicada. Este trabalho tem como objetivo apresentar uma abordagem baseada em design thinking para o desenvolvimento de produtos e serviços que envolvem software, alinhado às boas práticas da Engenharia de Software. Uma avaliação da abordagem foi realizada com os alunos da disciplina, concluindo que sua utilização melhorou a comunicação, documentação e acompanhamento dos projetos.*

1. Introdução

Disciplinas que alinham a teoria à prática são normalmente difíceis de se conduzir devido à necessidade de se tratar fatores técnicos (linguagens de programação, ferramentas, componentes, etc) e humanos (comunicação, gestão, disponibilidade, etc) ao mesmo tempo. Além disso, motivar os alunos e professores não é uma tarefa fácil, pois muitos imprevistos ocorrem durante o semestre. Normalmente disciplinas que envolvem projetos de desenvolvimento de produtos ou serviços (software ou hardware) são utilizadas em grades curriculares para tentar resolver esta dificuldade, e possibilitar uma experiência de desenvolvimento de aplicações/sistemas para os alunos.

O *design thinking* [Ambrose e Harris, 2011] tem se mostrado uma alternativa para o desenvolvimento de projetos de software bastante atraente. Suas atividades muitas vezes são encontradas em processos de desenvolvimento de software, e facilmente são entendidas e aplicadas por clientes e desenvolvedores.

O curso de graduação Sistemas e Mídias Digitais (SMD)[1], da Universidade Federal do Ceará, tem como objetivo formar profissionais com conhecimentos especializados em duas grandes áreas principais: Sistemas Multimídia e Mídias Digitais. Desta forma, contribui-se para o desenvolvimento de novos perfis profissionais que possam sustentar o desenvolvimento, viabilizando atividades produtivas nas áreas de geração de mídias digitais e desenvolvimento de sistemas multimídia, tais como: sistemas *web*, dispositivos móveis, jogos digitais e animações gráficas. Sua grade curricular possui um conjunto de disciplinas, chamadas oficinas, com o objetivo de possibilitar aos alunos uma experiência de desenvolvimento de sistemas. Essas disciplinas são: Oficina de Desenvolvimento de Sistemas Multimídia, Oficina de Desenvolvimento de Jogos Digitais e Oficina de Produção de Mídias Digitais.

Este trabalho tem como objetivo apresentar uma abordagem baseada em *design thinking* para o desenvolvimento de produtos e serviços que envolvam software. Sua intenção é aplicar conceitos de Engenharia de Software juntamente com as ideias de *design thinking*. Para a obtenção de um *feedback* sobre a abordagem, uma avaliação da abordagem foi realizada com os alunos da disciplina, onde concluiu-se que a utilização da abordagem colaborou para a melhoria da comunicação, documentação e acompanhamento dos projetos.

2. Abordagem Proposta

A abordagem proposta possui os seguintes objetivos: (1) desenvolvimento de produtos/serviços; (2) vivência de processos; (3) estabelecimento de relacionamentos; e (4) base para trabalhos de conclusão de curso. A metodologia baseou-se no *design thinking*, com algumas adaptações. Adicionou-se uma fase para planejamento inicial e acompanhamento dos projetos. A Figura 1 exibe as atividades e produtos da abordagem.

O acompanhamento das atividades ocorre pela análise de três elementos: *blog* de cada equipe, pontualidade na entrega das atividades, e presença da equipe nas aulas. Todas as atividades são entregues por meio de postagens em *blog*. Cada equipe deve elaborar e manter um *blog*[2] a ser utilizado durante toda a disciplina, sendo utilizado para a documentação de tudo o que acontece no projeto: decisões, diagramas, produtos, tarefas, experiências, processos de desenvolvimento, etc. Nele é onde os produtos que valerão nota serão armazenados. O produto final é acompanhado ao longo do semestre, desde suas etapas de concepção, até suas etapas finais de testes.

A avaliação da disciplina foi definida pela pontuação das atividades em equipe e individuais. As atividades em equipe são a elaboração do documento do projeto e dos dois vídeos, enquanto que as atividades individuais consistem no resumo e pôster. Há uma penalidade aplicada no caso em que menos de 50% dos componentes de cada equipe estejam ausentes nas aulas (não é necessária a presença de todos os alunos por aula, mas também é importante que não sejam sempre os mesmos alunos presentes, a exceção de toda a equipe presente), que ao final do semestre seria aplicada à nota final da equipe. Também existem penalidades pontuais em caso de atraso na entrega das

[1] Sistemas e Mídias Digitais (SMD) - http://smd.virtual.ufc.br/

[2] Lista dos *blogs* das disciplinas: http://goo.gl/VGQzkU

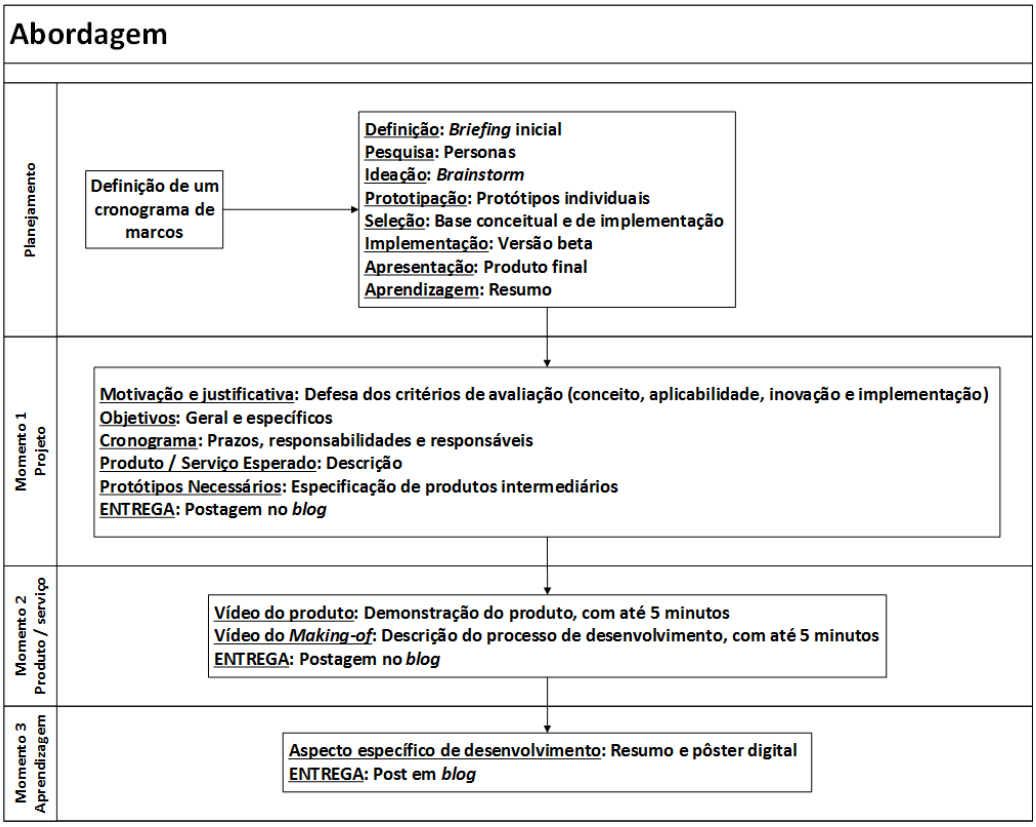


Figura 1. Visão geral da abordagem para o desenvolvimento de produtos/serviços baseada em práticas do design thinking.

atividades, influenciando na nota dos produtos. A nota final da equipe é composta pela soma das notas de cada atividade em equipe, e multiplicada pela quantidade de alunos da equipe. Esse valor é repassado à equipe que divide a nota entre si. Após a devolução das notas distribuídas entre os membros da equipe, cada aluno tem adicionada sua nota individual, compondo a nota final da disciplina. Assim incentiva-se a divisão justa e honesta dos pontos entre os membros da equipe.

3. Aplicação da Abordagem

Esta seção relata a aplicação da abordagem nas três disciplinas de oficina, no semestre 2015.1. Ao final da seção é descrita uma análise da visão dos alunos sobre a utilização da abordagem, por meio de um questionário aplicado com os alunos.

3.1. Relato das Atividades

Sete projetos foram desenvolvidos durante no semestre letivo, bastante diversificados: visualização de dados de *tweets* sobre o mapa do Brasil; *party game* que utiliza *smartphones* juntamente com cartas onde os jogadores tentam identificar os nomes de fontes tipográficas; gerador de personagens virtuais em 3D; instalação multimídia interativa, onde o usuário interage com a aplicação e esta responde em uma narrativa em torno do ciclo de vida de vagalumes; sistema de oferta e aquisição de bolsas de pesquisa ou de projetos; jogo pervasivo para um dispositivo móvel para auxiliar os alunos novatos do SMD no processo de adaptação à universidade; visualização da evolução dos usuários de uma rede social e suas interações. A Figura 2 exibe algumas telas dos

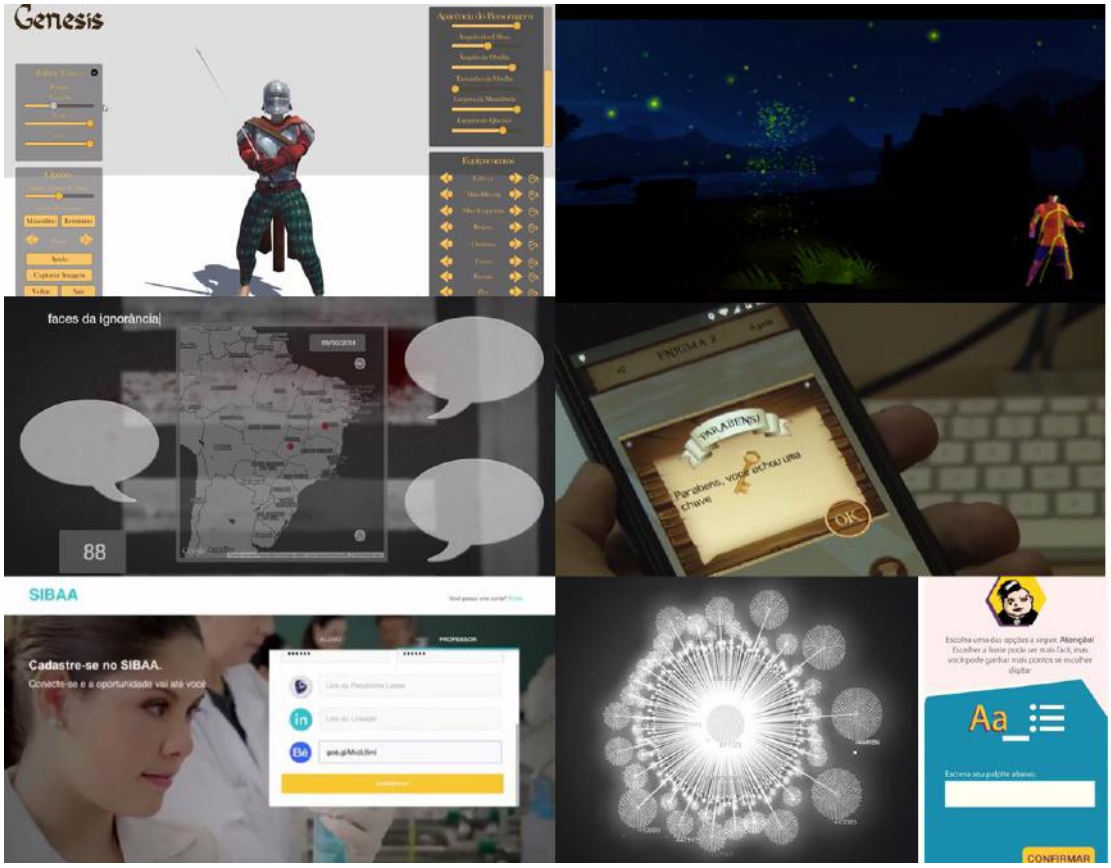


Figura 2. Aplicações desenvolvidas ao final das disciplinas.

produtos produzidos pelas equipes durante o semestre. Alguns aspectos se destacaram durante o semestre, sendo que eles faziam parte de atividades da abordagem proposta:

- Evolução dos trabalhos: percebeu-se uma evolução no nível dos projetos, tanto no nível técnico quanto no conteúdo, assim como a qualidade dos produtos finais. A complexidade dos trabalhos também foi maior, envolvendo em alguns casos uma mistura entre software e hardware, e entre o analógico e o digital;
- Foco no processo e projeto: em relação ao processo de desenvolvimento, houve um destaque para a comunicação/atendimento entre professores e alunos. O acompanhamento teve algumas deficiências, mais em relação ao planejamento; mas em relação às turmas anteriores foi bem melhor. Em relação aos produtos intermediários entregues, os alunos tinham a liberdade de escolher como representar o produto. Entretanto surgiram muitos diagramas de classe, requisitos funcionais, diagramas de atividades e *mockups* de tela;
- Foco na revisão: praticamente todos os produtos intermediários sofreram algum tipo de revisão por parte dos professores. Desde o momento inicial, protótipos, projeto, versões beta e resumos, todos foram lidos, algumas vezes de maneira individual, outras para toda a turma, de maneira que a sugestão de uma equipe muitas vezes acabou sendo útil para as demais. Esta atividade promoveu um acréscimo na qualidade dos trabalhos e minimizou falhas na comunicação;

- Utilização de *blogs* e vídeos para documentação: a estratégia de utilização de *blogs* para a documentação (não foi utilizada nenhum papel na disciplina) foi um ótimo mecanismo de comunicação, tanto para a documentação de decisões, requisitos, diagramas elaborados, quanto para armazenamento do material produzido, como os vídeos. Além disso, essa estratégia serve como divulgação dos projetos, da metodologia, disponibilização de exemplos e documentos para serem utilizados pelas próximas turmas e por outras disciplinas.

3.2. Avaliação e Análise dos Resultados

Para a avaliação da abordagem, um questionário foi aplicado aos alunos nas turmas das disciplinas (semestre 2015.1). O questionário utilizado apresentou questões objetivas (múltipla escolha) e discursivas. As questões objetivas foram formuladas conforme a escala de Likert [Likert, 32] onde o aluno deveria selecionar um valor em uma escala de 1 a 5, onde 1 corresponde a "discordo totalmente" e 5 corresponde a "concordo totalmente". Além disso, solicitou-se aos alunos que citassem os pontos fortes e pontos fracos da metodologia. O questionário foi composto pelas questões descritas na Tabela 1, e o gráfico de barras da Figura 3 foi gerado. Apenas 18 alunos responderam às questões. De maneira geral, todas as questões obtiveram notas altas. Todas as questões também tiveram mediana e moda próximas a 5, indicando que a maioria das respostas foram do tipo "concordam parcialmente" e "concordam fortemente", indicando que a abordagem sob o ponto de vista dos alunos beneficiou a disciplina.

Tabela 1. Questões objetivas aplicadas aos alunos nos questionários

Q1	Você conhecia a metodologia utilizada na disciplina?
Q2	Você acredita que a metodologia utilizada ajudou no desenvolvimento dos projetos?
Q3	O planejamento das atividades pela equipe de projeto foi facilitado devido à metodologia utilizada?
Q4	A comunicação / acompanhamento entre alunos e professores foi melhor devido à metodologia utilizada?
Q5	A comunicação entre os alunos da própria equipe foi melhor devido à metodologia utilizada?
Q6	A divisão de tarefas foi melhor devido à metodologia utilizada?
Q7	A equipe conduziu melhor as atividades devido à metodologia utilizada?

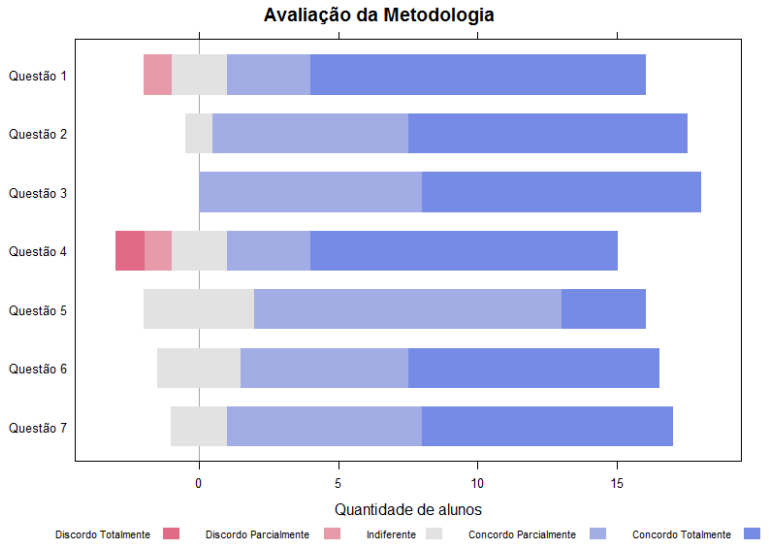


Figura 3. Gráfico de barras resultante da aplicação do questionário.

Na Figura 3 destacou-se um alto grau de concordância em todas as questões. Baseado nas respostas, identificou-se que os alunos se beneficiaram de um melhor planejamento nos projetos, e consequente melhor condução do desenvolvimento entre os membros da equipe. As questões 1 e 4 obtiveram respostas de discordância. A questão 1 era relacionada se o aluno conhecia o *design thinking*. Esta observação ficou sem explicação, pois em diversas outras disciplinas os alunos trabalharam com *design thinking*, esperando-se que fosse uma questão com notas altas. A questão 4 foi relacionada à comunicação entre aluno e professor. Nesta questão, os professores reconheceram que houve momentos de falta de clareza nas orientações e prazos, mais especificamente no que os produtos intermediários deveriam apresentar, e isso pode ter levado a erros de interpretação nas atividades e nos prazos.

Os pontos fortes mais recorrentes foram: acompanhamento (utilização de *checkpoints* para as atividades), comunicação (entre professores e equipes), utilização do *design thinking* (atividades e etapas), liberdade para as equipes (ideias, atividades, projeto, criação do produto), professores (facilidade em conversar, confiança nos alunos e criatividade) e tarefas (planejamento, divisão e *feedback*). Alguns aspectos foram mais específicos: exploração das diversas áreas do curso (disciplinas), flexibilidade nas decisões, fortalecimento da criatividade, protótipos para testes e seleção, e seleção do cliente. Os pontos fracos identificados foram: cronograma longo no início e curto no final, implicando em um esforço maior de desenvolvimento no final da disciplina; discordância entre professores, muitas vezes em relação a como proceder sobre produtos; foco mais no vídeo que no produto, produto final entregue na disciplina; falta de clareza em explicar quais produtos deveriam ser entregues; e certa desorganização nas tarefas da disciplina.

4. Considerações Finais

Este artigo apresentou uma abordagem baseada em *design thinking* para o desenvolvimento de produtos e serviços em uma disciplina do curso de graduação Sistemas e Mídias Digitais. Muitos conceitos de Engenharia de Software foram utilizados nessa disciplina como apoio à abordagem, principalmente relacionados à requisitos e revisões dos produtos. As principais contribuições deste trabalho foram: abordagem proposta e sua aplicação, avaliação da abordagem realizada pelos alunos, e melhoria da comunicação, documentação e acompanhamento dos projetos devido à utilização da abordagem. Entretanto algumas deficiências foram identificadas e serão ajustadas nas próximas edições da disciplina. Como trabalhos futuros pretende-se evoluir o mecanismo de avaliação dos alunos e dos projetos, e definição de um acompanhamento mais efetivo. A definição de quais produtos e o que eles devem conter também deve ser melhor documentada para proporcionar uma aplicação mais eficiente da abordagem. Por fim, aplicar novamente a abordagem nas disciplinas.

Referências

Ambrose, G., Harris, P. (2011) “Design Thinking”, ISBN 9788577807543, AVA Publishing SA.

Likert, R (1932) “Technique for the Measurement of Attitudes”, Archives of Psychology 140: pp. 1-55.

(Re)Evaluating the Influence of Contextualized Examples in Teaching an Introductory Software Engineering Course in Brazil

Arilo C. Dias-Neto[1], Rasha Osman[2]

[1]Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Manaus, AM, Brasil

[2]Department of Computing – Imperial College London
London SW7 2AZ, UK

arilo@icomp.ufam.edu.br, rosman@imperial.ac.uk

Abstract. *Examples and experiences used in international textbooks are derived from within the corresponding international software industries. This information is not available for non-international scenarios. In previous work, we have shown that student motivation and perception of an introductory software engineering course improved when using examples familiar to the students' environment. In this paper, we re-evaluate our work by conducting a study on a larger cohort of students. The results of this study confirm that of previous results and empirically show that using local examples familiar to the students' normal lives and environment will increase students' positive attitude towards teaching and learning Software Engineering.*

1. Introduction

Software engineering educators in emerging and developing countries mostly depend on internationally available textbooks and teaching material. Examples and experiences used in international textbooks are derived from within the corresponding international software industries. This information is not available for non-international scenarios, where the software engineering processes, realities and products differ due to environmental, cultural, and economic factors and need to be supported with realistic local examples. As an example, the reality of software engineering in Brazil is very different from that of developed countries. In general, the processes, techniques and methods used in software engineering are similar; however, the context and application areas are completely different.

There has been limited work addressing the contextualization of the computer science (e.g. [Vesisenaho et al. 2006]) and software engineering (e.g. [Fendler and Winschiers-Theophilus 2010] and [Osman 2012]) curriculum within the African context. Recent research into software engineering education in Brazil focused on developing pedagogy to close the gap between academia and the expanding software industry [Lucena et al. 2006; Lustosa-Neto et al. 2013; Santos et al. 2001; Santos and Soares 2013; Silveira-Neto et al. 2013].

This paper is an initial step towards research into contextualization and localization of the software engineering curriculum in emerging and developing countries. The primary hypothesis of this study is that using local examples familiar to the students' normal lives and environment will increase students' positive attitude towards teaching and learning Software Engineering and will enhance their appreciation of software engineering in general and in their local environment (in this case, Brazil).

A secondary objective of this study is to confirm the results presented in previous work [Osman and Dias-Neto 2014] using a larger cohort of students.

2. Study Design and Execution

2.1. Course Description and Group Distribution

This study was conducted between Sept and Dec'2014 at the Federal University of Amazonas, for the *Introduction to Software Engineering* module of the Information System Undergraduate Course, similar to the course described in [Osman and Dias-Neto 2014]. A total of 34 students started this course: 26 fulltime and 8 from industry, with only 32 completely the course. The total number of number of students in involved in this study is 68% more than our previous study. The course is organized in two weekly *lectures* based on two international Software Engineering textbooks: [Pressman 2010] and [Somerville 2011]. A weekly *tutorial* covers the same topics studied during the week using example software systems to explain the concepts. In addition, the course contained a practical group project based on actual systems, in which students are organized in groups of three or four to complete a selected software project. The final module grade is the mean of (1) the result of three written exams and (2) the score of the project work.

Similar to the previous study [Osman and Dias-Neto 2014], the students were randomly divided into two groups, the **control** and **experimental** groups. Students with industrial experience are split evenly between the two groups. Both groups attended the same lectures; however, they attended different tutorial sessions. The **Experimental group (E)** were given examples of software projects that can be applied to domains similar to the Brazilians' everyday activities. The **Control group (C)** used examples of software projects that are applied to scenarios not usual for Brazilians, but that are described in the main international software engineering books.

2.2. Examples used in the Tutorials

In the tutorials, the students systematically apply the material presented in the lectures to two example software system specifications. The example systems used are: (1) *Personal Electronic Organizer (PEO)*: an online web based personal contact list, calendar and notification system for desktops and mobile phones (familiar situation for the Brazilian students). (2) *Library*: web-based information system that manages the services offered by a university library, such as searching, borrowing and returning books (usual scenario for the Brazilian students). (3) *Parking*: embedded system to manage parking in a shopping mall. The system assists drivers in parking their vehicle and controls the free spaces to park. Several Software Engineering textbooks use this as an example, however, it represents an unconventional scenario in Brazil and therefore, students have not had access to such an application's features and specifics.

The tutorial sessions for each group used a different set of examples. The experimental group used the *PEO* and *Library* examples and the control group used the *Parking* and the *PEO* examples (the same examples used in the original course). This distribution gave the experimental group the advantage of being more familiar with the content and context of the examples used than the students in the control group.

2.3. Study Execution

To assess the hypothesis of this study, two questionnaires were used to analyze the attitudes of the students. A *pre-course* questionnaire that was filled out by each student

before teaching started with the objective of evaluating the student's general knowledge and perception of software engineering. A *post-course* questionnaire that was filled out *after* the course completed. This was a more extensive questionnaire that repeated some questions from the pre-course questionnaire, in addition to questions evaluating a student's perception of how each type of activity (lectures, tutorials, project and use of examples) supported their learning. The total number of questionnaires answered during this study is summarized in Table 1.

Table 1. Demographics of the student groups for both questionnaires.

Groups	pre-course (*)			post-course (**)		
	M	*F*	***Total***	*M*	*F*	***Total***
Experimental	14	4	**18**	13	4	**17**
Control	13	3	**16**	12	3	**15**
Total	27	7	**34**	25	7	**32**

* 1 student did not participate in the lecture when the pre-course questionnaire was administered.
** 1 male student joined the control group after the first lecture. In addition, 1 male from the experimental group and 2 males from the control group dropped the course mid-semester.

The details of the questions of the post-course questionnaire can be found in [Osman and Dias-Neto 2014]. Answers for all questions were Likert-type questions on the ordinal scale (1: Strongly Disagree, 2: Disagree, 3: Neutral, 4: Agree, 5: Strongly Agree). The pre-course questionnaire contained nine questions, the questions (Q1, Q2, Q3, Q5, Q6, Q7, Q8, and Q22) and another question (*I have applied software engineering techniques before*) which was not repeated in the post-course questionnaire.

3. Study Analysis

This Section details the results of the analysis of the attitudes of the experimental and control groups using descriptive statistics. Further, a statistical analysis is presented applying *one-tailed*, *two tailed and paired t-tests* at 0.05 level ($\alpha = 0.05$) and 0.10 level ($\alpha = 0.10$) to compare the means of the attitudes of both groups. In the following, we use the term *positive* attitude to mean answers of strongly agree and agree on the ordinal scale of both the pre-and post-course questionnaires. For all the analysis, *negative* stated questions were reverse scored.

3.1. Analysis of Pre-Course Attitudes

Initially, a *two-tailed t-test* was used to compare the mean attitudes of the pre-course questionnaire for both groups. No statistically significant difference between the mean attitudes of the experimental and control groups was found and hence both groups were equivalent in attitude before the course started.

The pre-course questions reflect the attitude and knowledge of students on software engineering in general and in Brazil. To analyze the effect of the different teaching methods on the students' attitudes, we first analyzed the changes in attitude for the questions of the pre-course questionnaire. We used the questions that appeared on both the *pre* and *post-course* questionnaires (only students that filled the pre-course questionnaire were included). Table 2 details the percentages, means, and standard deviations of pre and post- course attitudes for both groups. From Table 2, it is evident that the participants in the control group had a more positive attitude (71%) than the experimental group (61%) for the pre-course questionnaire. After the course, the experimental group showed a more overall positive attitude of 84%, with an increase of 23%, while the control group had a 76% positive attitude with a 5% increase from pre-course levels.

Table 2. Comparing the similar questions of the pre and post course questionnaires for the experimental and control groups.

Group	% of positive attitude		Mean (Stdev)		Paired t-test
	Pre	*Post*	*Pre*	*Post*	*p-value*
Experimental	61%	84%	3.79 (0.82)	4.46 (0.41)	.009
Control	71%	76%	4.00 (0.43)	4.23 (0.62)	.191

Further, a *paired t-test* was used to assess the differences between pre-course and post-course levels. The difference in attitudes from pre to post course levels was *statistically significant* for the experimental group at the 0.05 level, whereas it was *not statistically significant* for the control group.

3.3. Analysis of Attitude Measures

For a more detailed evaluation of the effect that the examples had on the attitudes of students, the questions were divided into different categories (attitude measures) to evaluate the students' attitudes toward each category. Similarly to the previous study, the attitude measures were Lectures (questions Q11 to Q15), Tutorials (questions Q16 to Q19), General Course (questions Q3, Q9, Q10 and Q20 to Q22), and Software Engineering (questions Q1, Q2, Q4 to Q8). Table 3 presents the distribution of answers considering all questions and participants for the post-course questionnaire. The experimental group showed a more overall positive attitude (85%) in comparison to the control group (78%).

Table 3. Distribution of answers for the post-course questionnaire for the experimental and control groups.

Item	Experimental Group					Control Group				
	SD	DI	NE	AG	SA	SD	DI	NE	AG	SA
Overall Attitudes	2%	1%	12%	19%	66%	2%	6%	14%	25%	53%
Lectures	0%	0%	8%	19%	73%	0%	1%	5%	36%	58%
Tutorials	0%	0%	6%	23%	71%	0%	2%	13%	25%	60%
General Courses	5%	1%	19%	22%	53%	7%	11%	13%	30%	39%
Software Engineering	1%	2%	14%	14%	69%	1%	9%	20%	13%	57%

From Table 4, when comparing the overall means of the attitudes of the experimental and control group using a *one-tailed independent samples t-test* the differences between attitudes was *statistically significant* at 0.05 level, whereas the previous cohort was not.

The attitudes of the experimental group are more skewed towards positive attitudes for all measures in comparison to the control group. The experimental and control groups showed a similar positive attitude of 92% and 94%, respectively, towards the lectures. For the tutorials, the overall positive attitude for the control group was 85% in comparison to 94% for the experimental group. Analyzing the overall attitudes of both groups for the *Lectures* and *Tutorials* attitude measures using a *one-tailed independent samples t-test* the differences between attitudes was not statistically significant (Table 4) for both attitude measures. In contrast, the previous cohort *was statistically significant* at 0.1 level for both attitudes. For the *General Course* attitude measure, the experimental group showed a 75% positive attitude in comparison to the 69% of the control group. From Table 4, the difference in mean attitudes for this measure was *statistically significant* at the 0.05 level. For the *Software Engineering* attitude measure, the overall positive attitude of the experimental group was 83% in

comparison to 70% for the control group and was also *statistically significant* at the 0.05 level (Table 4).

Table 4. Analysis of the attitude measures for the experimental and control groups.

Attitude Measure	Current Study			2012-2013 Study		
	Mean (Stdev)		one-tailed t-test	Mean (Stdev)		one-tailed t-test
	Group [E]	*Group [C]*	*p-value*	*Group [E]*	*Group [C]*	*p-value*
Overall Attitudes	4.47 (0.36)	4.20 (0.40)	0.03	4.29 (1.01)	3.80 (1.23)	0.18
Lectures	4.65 (0.44)	4.49 (0.39)	0.15	4.49 (0.57)	3.85 (1.12)	0.06
Tutorials	4.65 (0.46)	4.43 (0.39)	0.12	4.14 (1.30)	3.28 (1.22)	0.08
General Course	4.18 (0.54)	3.83 (0.59)	0.05	4.11 (1.11)	3.58 (1.22)	0.17
Software Engineering	4.49 (0.40)	4.17 (0.52)	0.03	4.39 (0.96)	4.15 (1.25)	0.32

For all attitude measures (Table 4), the current cohort has higher positive attitudes than the previous study's cohort. In addition, the previous experimental cohort had a statistically significant difference from the control group in the *Lectures* and *Tutorials* measures, whereas the current cohort had a statistically significant difference for the *Overall*, *General Course* and *Software Engineering* measures. For both studies, the experimental group had higher positive attitudes in comparison to the control group for all attitude measures. An observed difference between the two cohorts is that the current cohort benefited from an exceptional student from the experimental group who was engaged in the lectures through questions and discussion which lead to further discussions and questions by the rest of the students.

Even though the results of both studies differ in the attitude measure that had been affected by the use of familiar examples to the students' environment; these experiments have produced a positive effect on the experimental group that justifies further exploration into the localization of the teaching of software engineering.

4. Discussion

This work has emphasised the importance of *context* when delivering *content*. The use of familiar examples helped focus class discussion on the subject matter instead of the technical details of the example in itself. This had a very positive effect in teaching. The motivation of the experimental group was exceptionally clear during the progression of the teaching, especially during the project group work. A speculation is that because the students were aware of how and where the examples could be applied in their environment, they understood more. Most importantly, the local examples eliminate the *confusion factor*, i.e., the example's scenario, which could hinder the students' understanding of the subject matter. The main challenge of conducting this course was finding and preparing industrial case scenarios in sufficient detail to be used as local examples in software engineering. Due to the lack of literature documenting software engineering experiences in Brazil, the generic examples stated previously were used.

The study described in this paper, has empirically shown that student motivation and perception of the subject matter and the learning experience had improved when

using generic examples familiar to the students' environment. This paper is an initial step towards the goal of contextualization of the software engineering curriculum. However, this is not enough to provision for a complete curriculum. For educators to incorporate local examples across the curriculum a targeted study needs to be conducted to collect and analyze experiences and examples from local software industries. These must align with the objectives of the curriculum, in addition to the development goals of the software development sector. To achieve sustainability and continuity, collaboration and cooperation must be fostered between the software industry and academia and research.

Acknowledgment

The authors would like to thank CNPq, CAPES, FAPEAM, and INDT for the financial support for this research.

References

J. Fendler and H. Winschiers-Theophilus, "Towards Contextualised Software Engineering Education: An African Perspective," in ICSE '10 Cape Town, South Africa, 2010, pp. 599-607.

M. Vesisenaho, H. H. Lund, and E. Sutinen, "Contextual Analysis of Students' Learning during an Introductory ICT Course in Tanzania," in 4th IEEE TEDC'06. Iringa, Tanzania, 2006, pp. 9-13.

P. Silveira-Neto, J. Gomesb, E. Almeida, J. Leite, T. V. Batista, and L. Leite, "25 years of software engineering in Brazil: Beyond an insider's view," Journal of Systems and Software, vol. 86, no. 4, 2013, pp. 872-889.

R. Osman and A. Dias-Neto, "Motivating by examples: an empirical study of teaching an introductory software engineering course in Brazil," in 38th COMPSAC 2014. Västerås, Sweden, 2014.

R. Osman, "Teaching software engineering in developing countries: a position paper," in 36th COMPSAC 2012. Izmir, Turkey, 2012.

R. S. Pressman, Software Engineering: A Practitioner's Approach, 7th Edition, International ed: McGraw-Hill, 2010.

R. Santos, C. Werner, H. Costa, and S. Vasconcelos, "Supporting Software Engineering Education through a Learning Objects and Experience Reports Repository," in 23rd SEKE'2011. Miami, USA, 2011, pp. 272-275.

S. Santos and F. Soares, "Authentic assessment in software engineering education based on PBL principles: a case study in the telecom market," in 2013 ICSE. San Francisco, USA, 2013, pp. 1055-1062.

V. F. de Lucena, A. Brito, P. Gohner, and N. Jazdi, "A Germany-Brazil Experience Report on Teaching Software Engineering for Electrical Engineering Undergraduate Students," in 19th CSEET, 2006, pp. 69-76.

V. Lustosa-Neto, R. Coelho, L. Leite, D. Guerrero, and A. Mendonça, "POPT: a problem-oriented programming and testing approach for novice students," in 2013 ICSE. San Francisco, USA, 2013, pp. 1055-1062.

A Brief Experience Report on Teaching Software Methods in a Software Engineering Undergraduate Course

Valdemar Vicente Graciano Neto[1,2], Wylker Moreno[1], Vinćius Sebba Patto[1]
Edmundo Sérgio Spoto[1], Juliano Lopes de Oliveira[1]

[1] Instituto de Informática (INF/UFG), Universidade Federal de Goiás, Goiânia, Brasil

[2]ICMC, Universidade de São Paulo, São Carlos, Brasil

valdemarneto@inf.ufg.br, wylkerxpz@gmail.com

viniciussebba@inf.ufg.br, edmundo@inf.ufg.br, juliano@inf.ufg.br

Abstract. *Software methods comprise systematic approaches for software construction. The selection of appropriate methods has significative impacts on the success of software development projects. However, there is a lack of available content regarding this topic. The Guide to the Software Engineering Body of Knowledge (SWEBOK) presents some brief guidelines and provides pointers for supplementary material, but we could not find any bibliographical resource which investigates this topic in depth. In this sense, we report an experience on teaching software methods for a Software Engineering undergraduate course. First insights reveal a necessity of conducting research on this topic. Such investigation and respective results could suitably ground this area, benefiting software engineering pragmatics and theory.*

1. Introduction

In an academic environment, professors are concerned with the systematization of contents. They use their previous knowledge and experience, and align them to the theoretical basis delivered by reference authors to build a corpus of content. They elaborate presentations which can bring light to the theoretical topics. In a near future, that knowledge serves as basis for professionals' decisions in real organizational scenarios where their students are going to work. In this sense, it is important that the plethora of knowledge received by students have a strong basis which ground it to ensure the correctness and reliability of the content which they receive. However, Software Engineering (henceforth SE) is a young science and some areas do not have a well-established theoretical foundation yet. Software methods can considered one of those areas.

Software methods correspond to a well-defined set of related activities performed in a systematic way to accomplish software development properly [IEEE Computer Society 2014]. They are useful for software engineers since they prescribe the main principles the software team should follow to correctly deliver a software product. However, when software engineers need to apply one of those methods, there is no detailed explanation on how the methods should be structured, which recurrent activities comprise them, how they are related to the software development processes, and how they could be associated to produce a better result. Thus, in those situations, software engineers come back again to the pragmatism and experience-based decisions, since there is a lack of theoretical basis to ground their decisions.

Considering this gap, this paper briefly presents an experience report on an effort we have conducted in the last year. A teaching experience was performed on an undergraduate Software Engineering course in Universidade Federal de Goiás to systematize the available knowledge on software methods. We expect that this report can contribute to the improvement of undergraduate teaching of Software Engineering by exposing an important research for SE, instigating academic community to investigate this topic, and showing how we have dealt with the lack of structured content on this subject.

The remainder of the paper is organized as follows: Section 2 presents the theoretical concepts needed to discuss the topic. Section 3 reports our experiences and practices in software methods education. Section 4 discusses and analyses these practices and experiences. Section 5 presents conclusions and perspectives for future work.

2. Methods in Software Engineering

Considering Software Engineering as a layered technology, methods are in the context of software development processes [Pressman 2010]. A process is a framework which involves methods (composed by techniques and activities), and tools [IEEE Computer Society 2014], as depicted in Figure 1. A process can have many methods associated (i.e., a process can have Object-Oriented methods and Agile methods applied in association). A method can be a part of one or many processes (e.g., formal methods are used in some processes, but many software development processes do not use it). One method has at least one inherent technique (to exist), or many techniques related to it (e.g., classes identification via grammar classes for OO-method, stand-up meetings for agile methods, formal specification for formal methods, among others). And one technique can be part of many processes, as domain engineering, that analyses commonalities and variabilities, and can be a part of feature-driven development, or product-line software development.

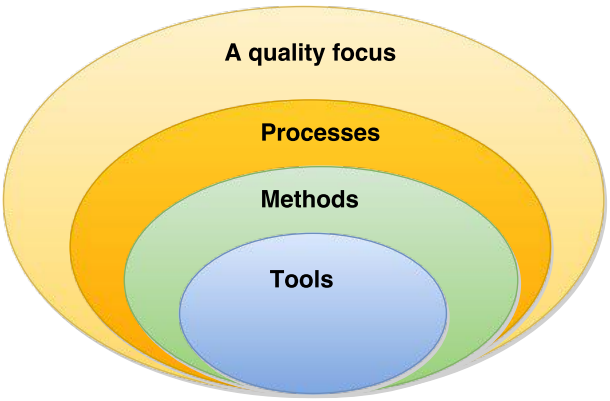


Figure 1. Software Engineering as a layered technology [Pressman 2010].

Processes, methods, techniques, and activities are part of Software Engineering theory. As such, they should be enough to predict how the work should be conducted to avoid repetitive labor and error [Johnson et al. 2012]. However, this is not true yet for software engineering.

Regarding methods, SWEBOK structures their theory in four well-delimited categories: Heuristic Methods (Procedural paradigm, data modeling paradigm, and object-oriented paradigm), Formal Methods (program specification and validation, and pro-

gram refinement and derivation), Prototyping Methods (styles, target, and evaluation techniques), and Agile Methods (Rapid Application Development (RAD), eXtreme Programming (XP), Scrum, and Feature-Driven Development (FDD)) [IEEE Computer Society 2014].

SWEBOK already gathers knowledge on methods. However, only the classical methods are comprised there, while emerging topics such as Agent-Oriented Software Engineering (AOSE), Aspect-Oriented Software Development (AOSD), and Model-Driven Development (MDD) and their inherent methods are only briefly mentioned majorly in other sections than Methods section. Additionally, methods are, by definition, a set of activities and techniques. Then, those activities and techniques should also be externalized to guide the software engineer on how to apply its knowledge. However, it does not happen.

Indeed, methods are crosscutting in software life cycle. Even if a team adopt only one type of method, there are intrinsic activities related to the method in each one of the software development life cycle steps. For instance, if Agile method is adopted, there are agile techniques in requirements engineering [Cao and Ramesh 2008, Fægri and Moe 2015], agile techniques in design [Turk et al. 2014, Duque M et al. 2014], agile techniques for programming [Martin 2003], agile techniques for testing [Talby et al. 2006], and for deployment [Pikkarainen et al. 2012]. In parallel, methods are not self-exclusive. Conversely, it is quite common to find situations where agile methods are used in association with object-oriented method, prototyping method, and data modeling method.

3. Teaching Software Methods - A brief report on the application of agile methods for Requirements Engineering

In the SE undergraduation course [de Lucena et al. 2008], an specific course on Software Methods is offered annually. Students attend to this specific course in the 6th semester. The SE undergraduation is structured in eight semesters. Henceforth, we use the word 'course' to denote the specific course of Software Methods. In 2014, this course was conducted by one of the authors. The course is integrated to a Software Factory, a curricular element of SE undergraduation which acts as a client for the courses. Students are motivated to develop software for real problems whose product owner is the Software Factory. In the last year, students were divided in two groups: one should develop a software to manage the *Lato Sensu* Courses selection process, while the another group should develop a software to manage the Young Mentoring program available at Instituto de Informática.

For the course context, students were required to conduct only the Requirements Engineering process for those elected projects. The forthcoming steps would be conducted in subsequent semesters. They were required to use Agile Methodologies (AM), the current software development life cycle adopted by the Software Factory. They should deliver products such as a list of user requirements, user stories, and a list of system requirements. AM was a suitable process since 1) they should deliver such products in the end of short iterations, 2) with a strong response to changes, 3) intensive communication, and 4) with use of self-contained and short artifacts, .

For the specific context of the course, the professor motivated students to deliver an additional product: a characterization of agile method used during requirements engi-

neering step described in terms of techniques. For requirements engineering, students did not have many difficulties. They described their agile method in function of the recurrent agile techniques for requirements elicitation, analysis, and documentation available in literature, such as Structured Interview [Hove and Anda 2005], viewpoint-oriented requirements definition [Kotonya 1999], and ethnography [Hughes et al. 1995].

In a second activity, students were required to characterize the SWEBOK's methods (such as prototyping or formal methods) as a set of specific techniques, using as literature content as their own experience. Next, they were required to also characterize the emerging methods which come from the new software development technologies as a set of techniques, such as Agent-Oriented Software Engineering [Jennings 1999], Aspect-Oriented Software Development [Filman et al. 2004], and Behavior-Driven Development [Solis and Wang 2011]. For both situations, they had remarkable difficulties. During an interview with them to elicit the key factors which influenced them to have bad results, they highlighted:

1. **A lack of references to support their activities**. We suspect that they were right. SWEBOK has a small amount of references about the classical methods; and emerging methods are not treated or even discussed, as in SWEBOK as in other theoretical initiatives as papers, conferences or journal articles;
2. **Difficulties to use their experience to solve this problem**. In fact, it is a really hard task. Decompose a method in techniques and activities require a perfect and holistic comprehension about the project is being conducted, a self-observation effort to elicit each of the performed activities to characterize the method, and a well-grounded experience, which students do not have yet;
3. **Difficulties to externalize even the most common activities**. Students had difficulties to characterize, for example, the agile method for the design step in software development. In that step, the only one activity they envisioned was 'design'. And, to perform this activity, they used a technique they called 'modeling'. In short, they are right. Even in technical material, there is not an extensive discussion on what exactly characterizes the agile method in function of activities in each one of the steps of the software development process.

4. Discussion

Given those difficulties, we selected some possibilities to address those problems in methods teaching for each one of the highlighted difficulties:

Lack of references. Recent initiatives to propose unifying theories for Software Engineering[1] have been reported [Sjøberg et al. 2008, Johnson et al. 2012, Johnson et al. 2013, Ekstedt 2013, Fitzgerald 2013, Ralph et al. 2014]. Thus, an endeavor regarding software methods is also necessary to produce knowledge which can guide professors, students, and professionals around this topic.

Difficulties on the use of own experience. As a matter of fact, even professors can face some drawbacks when founding in their own experience to elicit techniques in a method. Case studies must be conducted to support the creation of a concise theory of methods for software engineering.

Difficulties to externalize activities. Maybe this knowledge already exist and it is spread

[1] Software Engineering Method and Theory: http://semat.org/

in literature. Agile methods, for instance, are widespread. The recommended practice already disseminates several techniques which compose the agile method, such as pair-programming, stand-up meetings, and user stories. However, a systematization of this matter is welcome. A literature review should be conducted to support a suitable identification of the techniques which compose their methods. It is necessary to investigate how practitioners and academics have applied methods in their projects in each one of the software life cycle steps, completely characterizing each one of the SWEBOK's methods.

5. Final Remarks

This paper briefly discusses and reports an experience on teaching software methods to an Software Engineering undergraduate course. We observed that students had difficulties to characterize even well-disseminated methods, such as agile based and prototyping, as a set of techniques. We observed situations in which students were required to use both specialized literature and their own experience to characterize and choose methods in several contexts. As a result, they had difficulties to find solutions in both sources (literature and experience).

We claim that a systematization of knowledge on software methods is necessary, gathering the knowledge related to each specific method, and delivering it to practitioners, teachers, and students. In point of fact, a unified theory of methods in software engineering is also necessary to 1) support practitioners on the decisions about the methods which fit better for each situation; 2) to guide how to conduct methods within their software development life cycles; and 3) ground a theoretical basis for Software Engineering, enabling its evolution as a science and as an engineering.

References

Cao, L. and Ramesh, B. (2008). Agile requirements engineering practices: An empirical study. *Software, IEEE*, 25(1):60–67.

de Lucena, F. N., de Oliveira, J. L., and Vincenzi, A. M. R. (2008). Bacharelado em Engenharia de Software na Universidade Federal de Goiás. FEES '08, pages 1–4, Campinas, Brazil. UNICAMP.

Duque M, N., Pulgarin M, M., and Rios P, J. (2014). Foundations for a development methodology software agile model - driven from a modeling graphic customer-oriented language. In *Computing Colombian Conference (9CCC), 2014 9th*, pages 29–34.

Ekstedt, M. (2013). An empirical approach to a general theory of software (engineering). In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 1525–1526, Piscataway, NJ, USA. IEEE Press.

Fægri, T. E. and Moe, N. B. (2015). Re-conceptualizing requirements engineering: Findings from a large-scale, agile project. In *Scientific Workshop Proceedings of the XP2015*, XP '15 workshops, pages 4:1–4:5, New York, NY, USA. ACM.

Filman, R., Elrad, T., Clarke, S., and Ak?it, M. (2004). *Aspect-oriented Software Development*. Addison-Wesley Professional, first edition.

Fitzgerald, B. (2013). Uncovering theories in software engineering. In *2nd SEMAT Workshop on a General Theory of Software Engineering (GTSE)*, ICSE '13, pages 1525–1526, Piscataway, NJ, USA. IEEE Press.

Hove, S. and Anda, B. (2005). Experiences from conducting semi-structured interviews in empirical software engineering research. In *Software Metrics, 2005. 11th IEEE International Symposium*, pages 10 pp.–23.

Hughes, J., O'Brien, J., Rodden, T., Rouncefield, M., and Sommerville, I. (1995). Presenting ethnography in the requirements process. In *Proc. of RE*, pages 27–34.

IEEE Computer Society (2014). *Software Engineering Body of Knowledge (SWEBOK) - V3*. Angela Burgess, EUA.

Jennings, N. R. (1999). Agent-oriented software engineering. In Imam, I., Kodratoff, Y., El-Dessouki, A., and Ali, M., editors, *Multiple Approaches to Intelligent Systems*, volume 1611 of *LNCS*, pages 4–10. Springer Berlin Heidelberg.

Johnson, P., Ekstedt, M., and Jacobson, I. (2012). Where's the theory for software engineering? *IEEE Software*, 29(5):96.

Johnson, P., Jacobson, I., Goedicke, M., and Kajko-Mattsson, M. (2013). 2nd semat workshop on a general theory of software engineering (gtse 2013). In *Proc. of ICSE '13*, pages 1525–1526, Piscataway, NJ, USA. IEEE Press.

Kotonya, G. (1999). Practical experience with viewpoint-oriented requirements specification. *Requirements Engineering*, 4(3):115–133.

Martin, R. C. (2003). *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Pikkarainen, M., Salo, O., Kuusela, R., and Abrahamsson, P. (2012). Strengths and barriers behind the successful agile deployment—insights from the three software intensive companies in finland. *Empirical Software Engineering*, 17(6):675–702.

Pressman, R. (2010). *Software Engineering: A Practitioner's Approach*. McGraw-Hill, Inc., New York, NY, USA, 7 edition.

Ralph, P., Exman, I., Ng, P.-W., Johnson, P., Goedicke, M., Kocata, A. T., and Yan, K. L. (2014). How to develop a general theory of software engineering: Report on the gtse 2014 workshop. *SIGSOFT Softw. Eng. Notes*, 39(6):23–25.

Sjøberg, D., Dybå, T., Anda, B., and Hannay, J. (2008). Building theories in software engineering. In Shull, F., Singer, J., and Sjøberg, D., editors, *Guide to Advanced Empirical Software Engineering*, pages 312–336. Springer London.

Solis, C. and Wang, X. (2011). A study of the characteristics of behaviour driven development. In *37th SEAA*, pages 383–387.

Talby, D., Keren, A., Hazzan, O., and Dubinsky, Y. (2006). Agile software testing in a large-scale project. *Software, IEEE*, 23(4):30–37.

Turk, D., France, R. B., and Rumpe, B. (2014). Limitations of agile software processes. *CoRR*, abs/1409.6600.