IX Fórum de Educação em Engenharia de Software

CBSOFT.ORG

REALIZAÇÃO: SBC

EXECUÇÃO: UEM Universidade Estadual de Maringá; UTFPR Câmpus Campo Mourão; DIN Departamento de Informática

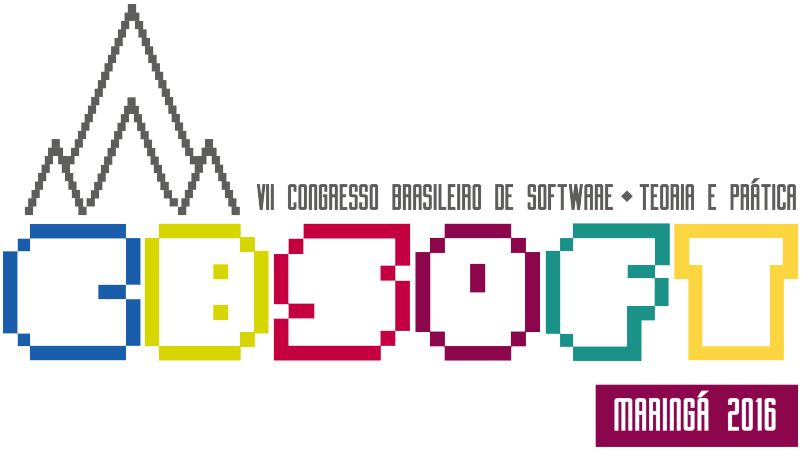
ORGANIZAÇÃO: TASA Eventos

APOIO: Convention & Visitors Bureau Maringá e Região

FOMENTO: CNPq; Paraná Governo do Estado; Fundação Araucária; UEM Universidade Estadual de Maringá; CAPES

PATROCINADORES GIGA: UniCesumar; COLIVRE

PATROCINADORES MEGA: ESPWEB; ThoughtWorks



IX FÓRUM DE EDUCAÇÃO EM ENGENHARIA DE SOFTWARE (FEES 2016)

22 a 23 de setembro de 2016 | *September 21 - 22, 2016*
Maringá, PR, *Brazil*

ANAIS | *PROCEEDINGS*

Sociedade Brasileira de Computação – SBC

COORDENADOR DO COMITÊ DE PROGRAMA | *PROGRAM COMMITTEE CHAIR*
Marcelo Hideki Yamaguti (PUCRS)

EDITORES | *PROCEEDINGS CHAIRS*
Marco Aurélio Graciotto Silva (UTFPR)
Willian Nalepa Oizumi (IFPR)

COORDENADORES GERAIS | *GENERAL CHAIRS*
Edson Oliveira Júnior (UEM)
Thelma Elita Colanzi (UEM)
Igor Steinmacher (UTFPR)
Ana Paula Chaves Steinmacher (UTFPR)
Igor Scaliante Wiese (UTFPR)

REALIZAÇÃO | *REALIZATION*
Sociedade Brasileira de Computação (SBC)

EXECUÇÃO | *EXECUTION*
Universidade Estadual de Maringá (UEM) – Departamento de Informática (DIN)
Universidade Tecnológica Federal do Paraná (UTFPR) – Câmpus Campo Mourão (UTFPR-CM)

ISBN: 978-85-7669-341-3

Apresentação

O IX Fórum de Educação em Engenharia de Software (FEES 2016) é um evento participante do XXX Simpósio Brasileiro de Engenharia de Software que visa a ser um espaço e um momento em que, por meio das publicações, apresentações e discussões das pesquisas e relatos de experiências, como comunidade científica e educacional, possa-se avançar em temas relacionados ao ensino-aprendizagem em Engenharia de Software, em particular no contexto brasileiro.

Os temas serão abordados por meio de painel, sessões de apresentação de artigos e atividades em grupo de trabalho. Nesta edição, o FEES 2016 promoverá espaço para a apresentação do trabalho de 'Refinamento de Competências do Egresso do Curso de Engenharia de Software' que foi construído ao longo de várias edições do FEES. Também haverá espaço para a discussão de um Currículo de Referência da SBC para o Bacharelado em Engenharia de Software.

Esta edição do FEES recebeu a inscrição de 35 trabalhos, com a submissão final de 33 artigos. Todos os artigos foram avaliados por, no mínimo, 3 revisores com conhecimento nos temas do evento. Dos artigos submetidos 13 foram aprovados, o que gera uma taxa de aceitação de 39,39%.

Aos palestrantes, autores e demais membros da comunidade que contribuíram para a realização deste evento, um agradecimento pelo reconhecimento e confiança no evento. Ao Comitê de Programa pela disponibilidade e dedicação ao trabalho de avaliação de artigos, um agradecimento especial.

A todos os participantes: boas-vindas e desejos de que todos possam aproveitar ao máximo as atividades da programação desta edição do Fórum de Educação em Engenharia de Software.

Marcelo Hideki Yamaguti (PUCRS)
Coordenação do FEES 2016

Comitê técnico | *Technical committee*

Coordenador de comitê de programa | *PC chair*

Marcelo Hideki Yamaguti - Pontifícia Universidade Católica do Rio Grande do Sul

Comitê de programa | *Program committee*

Ana Paula Bacelo - Pontifícia Universidade Católica do Rio Grande do Sul
Ana Regina Rocha - COPPE/Universidade Federal do Rio de Janeiro
Christina Chavez - Universidade Federal da Bahia
Claudia Werner - COPPE/Universidade Federal do Rio de Janeiro
Daltro Nunes - Universidade Federal do Rio Grande do Sul
Edmundo Spoto - Universidade Federal de Goiás
Eduardo Figueiredo - Universidade Federal de Minas Gerais
Fabio Rocha - Universidade Tiradentes
Francisco Dantas - Universidade Federal do Rio Grande do Norte
Gleison Santos - Universidade Federal do Estado do Rio de Janeiro
Heitor Costa - Universidade Federal de Lavras
Hilmer Neri - Universidade de Brasília
Ingrid Nunes - Universidade Federal do Rio Grande do Sul
Jair Leite - Universidade Federal do Rio Grande do Norte
João Pablo S. da Silva - Universidade Federal do Pampa
Juliano Lopes de Oliveira - Universidade Federal de Goiás
Julio Leite - Pontifícia Universidade Católica do Rio de Janeiro
Leila Ribeiro - Universidade Federal do Rio Grande do Sul
Marcelo Barbosa - Pontifícia Universidade Católica de Minas Gerais
Marcelo Hideki Yamaguti - Pontifícia Universidade Católica do Rio Grande do Sul
Marco Aurélio Wehrmeister - Universidade Tecnológica Federal do Paraná
Marco Aurélio Graciotto Silva - Universidade Tecnológica Federal do Paraná
Maria Augusta Vieira Nelson - Pontifícia Universidade Católica de Minas Gerais
Maurício Aniche - Universidade de São Paulo
Márcio Barros - Universidade Federal do Estado do Rio de Janeiro
Milene Serrano - Universidade de Brasília
Paulo Meirelles - Universidade de Brasília
Rafael Prikladnicki - Pontifícia Universidade Católica do Rio Grande do Sul
Regina Braga - Universidade Federal de Juiz de Fora
Roberta Coelho - Universidade Federal do Rio Grande do Norte
Rodolfo Resende - Universidade Federal de Minas Gerais
Rodrigo Reis - Universidade Federa do Pará
Thiago Mendes - Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Uirá Kulesza - Universidade Federal do Rio Grande do Norte
Valter Camargo - Universidade Federal de São Carlos

Palestra de abertura | *Keynote*

Artigos Técnicos | *Technical papers*

Artigo - Grupo de Trabalho | *Paper – Workgroup*

Auto Percepção da Empregabilidade em Engenheiros de Software

Bruno do S. Rocha[1], César França[2]

[1]Centro de Estudos e Sistemas Avançados do Recife (CESAR)
R. do Brum, 77 - Recife, PE, 50030-260

[2]Departamento de Informática e Estatística (DEINFO)
Universidade Federal Rural de Pernambuco.

bdsr74@gmail.com, cesar@franssa.com

Abstract. *The frequent technological evolution generates an unexpected effect over the labour world: the constant risk for software engineers to get professionally obsolete. With aims to understand how software engineers deal with this risk, and interpret their (current and future) employability, this research conducted a survey with 106 software engineers represented mostly by programmers, system analysts and testers in Brazil. The results reveal that less experienced engineers tend to be more optimistic about their future, but that is a trap. The more optimistic engineers are, the less they are aware of the external factors over their employability. These results may help to inform engineers about their own employability, as well as for labour counsellors to guide them more effectively.*

Resumo. *A frequente evolução tecnológica gera um efeito colateral no mundo do trabalho que é o constante risco de os Engenheiros de Software tornarem-se profissionais defasados. Com o objetivo de entender como os engenheiros interpretam a sua empregabilidade (atual e futura), conduzimos um survey com 106 engenheiros de software representados principalmente por programadores, analistas de sistemas e testadores brasileiros. Os resultados apontam que os menos experientes tendem a ser mais otimistas em relação ao futuro, mas os mais otimistas tendem a desconsiderar a influência de fatores externos sobre a sua empregabilidade. Estes resultados podem auxiliar tanto na melhoria da auto percepção dos próprios engenheiros, como também no aconselhamento e planejamento de carreira para os mesmos.*

1. Introdução

O termo empregabilidade pode ser definido, em poucas palavras, como a capacidade de um indivíduo obter um emprego idealmente seguro, sustentável e gratificante, manter-se no mesmo ou se necessário, conseguir um novo [McQuaid & Lindsay 2005; McGrath 2009]. Nesse contexto, um dos maiores desafios do profissional de tecnologia da informação (TI) é a capacidade de se adaptar às constantes mudanças impostas pelo mercado. Diferente de outras profissões, onde o conhecimento e as habilidades se dissolvem mais lentamente, estima-se que um profissional da área de TI

leve apenas dois anos para encontrar-se defasado profissionalmente [Miller & Dettori 2008; Ang & Slaughter 2000].

A defasagem profissional ou obsolescência, na literatura técnica, é descrita como o grau em que os profissionais de uma organização não têm o conhecimento ou habilidades necessárias para manter um desempenho eficaz tanto em seus papéis de trabalho atuais, quanto em futuros [Dubin 1972].

Apesar de obter na academia as habilidades e conhecimentos necessários para ingressar, manter-se e desenvolver-se no mercado de trabalho [Miller & Dettori 2008], uma vez empregado, passará a ser de responsabilidade do profissional manter-se em sintonia com as necessidades da indústria [Swigon 2011].

A soma das habilidades e competências individuais adquiridas por meio de experiência profissional ou transferência de conhecimento, bem como a posição em que o indivíduo se encontra refletem o nível de empregabilidade do mesmo [Swigon 2011]. Contudo, existem fatores externos, fora do controle do indivíduo, que também exercem influência no seu nível de empregabilidade [McQuaid & Lindsay 2005]. A capacidade de percepção dos fatores intrínsecos e extrínsecos ao profissional será determinante para o processo de auto-gestão do conhecimento e empregabilidade [Jefferson 2006].

Neste trabalho, um survey foi realizado com a intenção de explorar a percepção de engenheiros de software quanto à influência de um conjunto de fatores sobre a sua empregabilidade. Para tal, apresentamos uma síntese de fatores que influenciam na empregabilidade a partir da literatura técnico-científica. Fez-se então uso de uma pesquisa estruturada, distribuída por meio da internet, obtendo-se 106 respostas de profissionais da engenharia de software. No questionário, perguntamos acerca da sua auto-percepção sobre a sua empregabilidade, em diferentes fases da carreira profissional, e obtivemos um panorama sobre a atitude dos engenheiros no combate da sua obsolescência.

Os resultados apresentados nesta pesquisa podem ter uma utilidade direta de conscientização de engenheiros de software preocupados com a sua obsolescência e empregabilidade, em particular daqueles em início de carreira. Para gestores de programas de formação, este artigo traz dados que fundamentam o argumento de que uma das habilidades mais importantes a serem desenvolvidas ao longo de um curso é o auto-desenvolvimento, para garantir profissionais bem preparados ao longo do tempo.

Nas seções seguintes serão discutidos, respectivamente, conceitos relacionados com a empregabilidade do indivíduo, desenvolvimento das hipóteses, metodologia aplicada, resultados e discussões, e por fim, as conclusões dos autores sobre o estudo.

2. Revisão Bibliográfica

2.1. Empregabilidade e fatores empregáveis

O conceito de empregabilidade é investigado na literatura técnica-científica sob três perspectivas distintas. A primeira refere-se à empregabilidade da massa trabalhadora, e está relacionada com as políticas do governo, preocupadas com as mudanças naturais do fluxo de trabalho em massa. A segunda perspectiva diz respeito a empregabilidade como estratégia de recursos humanos, e aborda a empregabilidade

baseada em habilidades e flexibilidades pessoais adquiridas através do emprego. Por fim, a terceira perspectiva está relacionada com a empregabilidade do indivíduo, isto é, a capacidade de formar graduados com as habilidades que os empregadores necessitam [McGrath 2009; Kaufman 1974].

Neste trabalho, daremos ênfase à segunda abordagem, que define o termo empregabilidade como a capacidade de um indivíduo obter um emprego (idealmente) seguro, sustentável e gratificante, manter-se no mesmo ou se necessário, conseguir um novo. Sob esse ponto de vista, os fatores que exercem influência direta sobre a empregabilidade do indivíduo são divididos em três grupos [McQuaid & Lindsay 2005]:

- Fatores internos: Dizem respeito às habilidades, conhecimentos, atitudes e outras características do indivíduo, sejam elas adquiridas ou moldadas na formação básica, na academia, ou já no exercício das suas atividades. Por exemplo: Experiência profissional, domínio de tecnologia, idiomas, perfil de comportamento, entre outros.
- Circunstâncias pessoais: Referem-se à conjuntura contextual que cerca o indivíduo em um determinado momento, não possuindo relação direta com suas atividades profissionais. São exemplos situação familiar e compromissos domésticos, cultura de trabalho e acesso a transporte público ou privado.
- Fatores externos: São aspectos e agentes externos, que não estão exatamente sobre o controle do indivíduo, mas que impactam diretamente na sua profissão, tais como a situação macroeconômica, mercado de trabalho, políticas de emprego, etc.

A missão genérica de um programa de formação superior em qualquer que seja a área é proporcionar ao graduando o desenvolvimento de um conjunto de competências necessárias para ingressar, manter-se e desenvolver-se no mercado de trabalho por tanto tempo quanto seja necessário ao longo da sua vida produtiva. No caso da realidade específica da área de tecnologia da informação, as técnicas e ferramentas tornam-se obsoletas rapidamente, e, portanto, demanda-se que estes programas de formação desenvolvam habilidades que não estão diretamente ligadas a uma tecnologia ou plataforma, como visto no quadro abaixo [Miller & Dettori 2008].

Tabela 1. Habilidades trabalhadas em engenheiros de software

Tópico	Descrição
Abstração	Saber como aplicar abstração como um princípio organizador para gerenciar a complexidade de um projeto de tecnologia da informação.
Interface com usuário	Entender como controles para interfaces gráficas são modelados como objetos e suas propriedades e métodos correspondentes.
Modelagem de domínio	Ser capaz de representar o mundo real em objetos de uma aplicação ou tabelas de um banco de dados.
Separação de interface e implementação	Estar aptos a separar conceitualmente a implementação básica do sistema e interagir com o usuário ou outros sistemas

2.2. Trabalhos relacionados

Segundo Joseph & Ang (2011), para ter um desempenho satisfatório, o profissional de tecnologia da informação (TI) precisa manter a sua base de conhecimentos em constante atualização. Fu (2011) atribui tal necessidade à globalização dos meios de produção e à competitividade natural da indústria de TI, que promove o surgimento frequente de tendências tecnológicas ou melhorias nas ferramentas já existentes. Ang (2000) se utiliza destes fatores, entre outros, para estimar que o tempo de vida do conhecimento e habilidades do profissional de TI seja de apenas dois anos.

Considerando a rápida dissolução da base de conhecimentos da área de TI, Miller & Dettori (2008) propõem investir mais esforços no ensino de técnicas que não possuam dependência direta de ferramentas ou plataformas, como forma de construir uma base de conhecimentos mais sólida e menos volátil. Saravanan (2006), por sua vez, aponta a capacidade de aprendizado contínuo e a administração do conhecimento como habilidades fundamentais para estes profissionais, reconhecendo a importância da retenção de competências não técnicas, moldadas com o exercer da profissão, mas enaltecendo também a capacidade cognitiva.

A retenção e renovação do conhecimento são igualmente objetos de estudos que investigam como o profissional se atualiza e a sua relação com a percepção do risco da obsolescência, atribuindo à percepção do risco da obsolescência uma relação positiva com a atualização em função do mercado e negativa com a atualização por diversão [Joseph et al. 2011]. O risco da obsolescência é também citado como fator de stress e um forte indicador para medir a intenção de indivíduo mudar de emprego ou ainda de área de atuação [Joseph & Ang 2001].

Kaufman (1974) relaciona a obsolescência com a perda da empregabilidade e afirma que o grau de comprometimento com a sua carreira profissional será preponderante para que o indivíduo se incline a tomar uma direção contrária ao risco da obsolescência ou conviva passivamente com o mesmo, até que ele exerça influência sobre a vida profissional. Nestes casos, a capacidade de percepção dos fatores intrínsecos e extrínsecos ao profissional será determinante para o processo de auto-gestão do conhecimento e empregabilidade [Jefferson 2006].

3. Metodologia

Nesta seção serão apresentadas hipóteses acerca da percepção da empregabilidade por engenheiros de software, baseadas e (ou) fundamentadas pelo referencial teórico apresentado na seção anterior, bem como detalhes sobre o método desenhado para realização da pesquisa.

3.1. Desenvolvimento de hipóteses iniciais

Partindo de um pressuposto inicial de que a auto-avaliação da empregabilidade seria reflexo exclusivamente da atitude do indivíduo perante a situação, propomos a investigação da primeira hipótese:

> H1: *A auto-avaliação da empregabilidade dos engenheiros de software é aproximadamente a mesma, em diferentes momentos da carreira.*

Se H1 for confirmada, então é verdade que a auto-avaliação dos indivíduos sobre a sua empregabilidade, em geral, é muito pouco afetada pelo seu momento na carreira. Por outro lado, se H1 for refutada, abre-se espaço para novos questionamentos, especialmente sobre quais são efetivamente os fatores que influenciam de forma significativa a mudança nesta auto-avaliação dos engenheiros.

Sendo assim, apesar do esforço da academia para preparar os profissionais adequadamente para o mercado de trabalho, habilidades não-técnicas dificilmente são assimiladas em sala de aula, uma vez que são adquiridas e desenvolvidas no decorrer da carreira profissional, quando o indivíduo se torna mais maduro e confortável com a realização das suas atividades, como por exemplo: a habilidade de comunicação, pensamento crítico e resolução de problemas, trabalho em equipe, aprendizado contínuo, empreendedorismo, ética e profissionalismo, e por fim, habilidades de liderança [Saravanan 2006]. Portanto, propomos avaliar se:

> H2: *O tempo de serviço é um fator que modera a auto-avaliação da empregabilidade de um momento anterior para um momento posterior.*

Este fenômeno, se confirmado, configuraria um otimismo generalizado, uma vez que o tempo de serviço só aumenta ao longo do tempo. Por otimismo, entendemos que auto-avaliação da empregabilidade dos engenheiros em um momento posterior seria sempre significativamente maior do que a nota atribuída para a auto-avaliação da sua empregabilidade anterior.

Mas como discutido na seção anterior, existem outros fatores externos ao indivíduo que podem influenciar na auto-avaliação da sua empregabilidade. No entanto, os engenheiros podem não ser conscientes destes fatores, e atribuírem à sua empregabilidade um conjunto de fatores exclusivamente internos (que estariam sob seu próprio controle), exclusivamente externos (que estariam fora de seu controle), ou ambos. Sendo assim, é plausível investigar também se este *lócus* de controle da empregabilidade é percebido de maneira diferente entre engenheiros, e se esta diferença de percepção influencia na auto-avaliação da empregabilidade. Por isso, apresentamos as seguintes hipóteses finais:

> H3: *O tempo de serviço está correlacionado com o lócus de controle.*
>
> H4: *O lócus de controle da empregabilidade modera a auto-avaliação da empregabilidade de um momento anterior para um momento posterior.*

Na Figura 1 ilustramos o modelo de investigação de forma simplificada para facilitar o entendimento das hipóteses e orientar a análise.

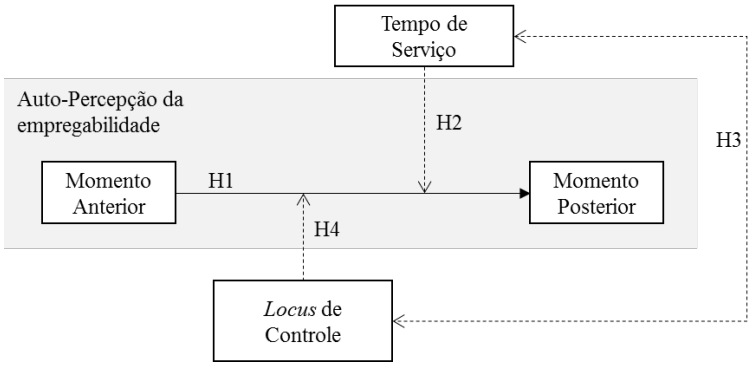


Figura 1. Modelo de investigação da percepção da empregabilidade por grupo

3.2. Instrumento e coleta de dados

Para avaliar então as hipóteses apresentadas na seção anterior, optou-se por uma abordagem positivista indutiva, utilizando como método de procedimento um survey estruturado [Easterbrook, Singer & Storey 2008]. Para alinhar a interpretação dos participantes, ao apresentar o questionário explicou-se que o conceito de empregabilidade investigado neste trabalho se referia à "capacidade de um indivíduo obter um emprego (idealmente) seguro, sustentável e gratificante, manter-se no mesmo ou se necessário, conseguir um novo" [McQuaid & Lindsay 2005].

Todas as perguntas colocadas no questionário eram objetivas, com alternativas de respostas pré-fixadas. A parte inicial do questionário diz respeito ao perfil. Foram formuladas perguntas para identificar a faixa etária e tempo de profissão. Na sequência, foram distribuídas três questões acerca da visão do entrevistado sobre a sua empregabilidade, em diferentes estágios da carreira profissional: Início da carreira (quando saiu da universidade), avaliação atual e como o mesmo classificaria a sua empregabilidade daqui a 10 anos. Para todas elas, foi dada a opção ao entrevistado de classificar a sua percepção utilizando uma escala tipo Likert, com itens que variavam de 1 (para muito baixa) a 5 (para muito alta).

Por fim, como forma de obter a percepção do entrevistado sobre os fatores empregáveis (internos e externos) mais relevantes, foram apresentados 5 itens e solicitado que o participante apontasse, livremente, aqueles que, na sua opinião, exercem influência significativa na empregabilidade de um profissional de TI. Foram eles: Mercado de trabalho (fator externo); Legislação trabalhista (fator externo); Experiência profissional (fator interno); Domínio de tecnologias ou plataformas de trabalho modernas (fator interno); Formação acadêmica, certificados e títulos (fator interno); Outros (texto livre).

O questionário foi enviado primeiramente para profissionais da engenharia de software da rede profissional, pessoal e acadêmica dos autores e então repassado para as suas redes de contato. A coleta foi realizada por meio de um formulário eletrônico, disponibilizado através da ferramenta *Google Forms*, mantido no ar por 15 dias corridos, durante o mês de fevereiro de 2016. Apesar da amostragem ser realizada por auto-seleção, onde os respondentes não necessariamente possuem algum vínculo com os autores, esta é uma limitação à validade desta pesquisa.

3.3. Análise dos dados

O primeiro passo para analisar os dados foi distinguir os entrevistados de acordo com as suas características de experiência profissional e percepção de fatores empregáveis. Consideramos como profissionais POUCO EXPERIENTES aqueles que indicaram que tinham até 10 anos de carreira, e o complemento como PROFISSIONAIS EXPERIENTES e consolidados no mercado. Em relação ao *lócus* de controle da empregabilidade, agrupamos os indivíduos que indicaram exclusivamente fatores INTERNOS, EXTERNOS e AMBOS.O passo seguinte foi confrontar a auto percepção da empregabilidade de cada grupo, confrontando as auto-avaliações dos momentos anteriores com os momentos posteriores: passado→presente, passado → futuro e presente→futuro.

Para avaliar as hipóteses H1, H2 e H4 estatisticamente foi utilizado o Teste de Wilcoxon, que consiste em um teste não-paramétrico concebido para avaliar a significância da diferença entre dois conjuntos de dados ordinais pareados, a partir de amostras [Rosner, Glynn & Lee 2006]. Já para avaliar a hipótese H3, utilizamos o teste do qui-quadrado, uma vez que uma das variáveis (*lócus* de controle) era nominal. Para aplicar ambos os testes recorremos à ferramenta *Social Science Statistics*[1], que disponibiliza uma ferramenta para realizar o cálculo online, a partir de duas amostras.

4. Resultados e Discussão

4.1. Descrição dos dados

Dos aproximadamente 150 envios, tivemos 106 respostas válidas, atingindo uma taxa de resposta de 70%, considerada excelente para este tipo de pesquisa online. Todos os participantes exercem funções relacionadas com a engenharia de software, destacando-se entre os entrevistados: programadores, analistas de sistemas e testadores. A Figura 2 e a Figura 3 ilustram a distribuição da faixa etária e do tempo de profissão dos participantes respectivamente.

A faixa etária predominante foi entre 31 e 40 anos (53%); o tempo médio de experiência profissional foi entre 5 e 15 anos (64%). A maior parte dos entrevistados enxerga com bons olhos a sua empregabilidade em diferentes momentos, como pode ser visto na Figura 4, Figura 5 e Figura 6. Dentre os fatores empregáveis mais facilmente percebidos entre os participantes obtivemos a experiência profissional (86%), o domínio de tecnologias ou plataformas de trabalho modernas (79%) (ambas referentes ao *lócus* interno de controle da empregabilidade), seguidos pelo mercado de trabalho (68%). Os participantes indicaram ainda outros fatores adicionais, tais como rede de relacionamentos profissional (5), domínio da língua inglesa (2), e reputação entre os pares (2), que correspondem juntos a 9% das respostas.

[1] Disponível no endereço: http://www.socscistatistics.com/, último acesso em 23/06/2016.

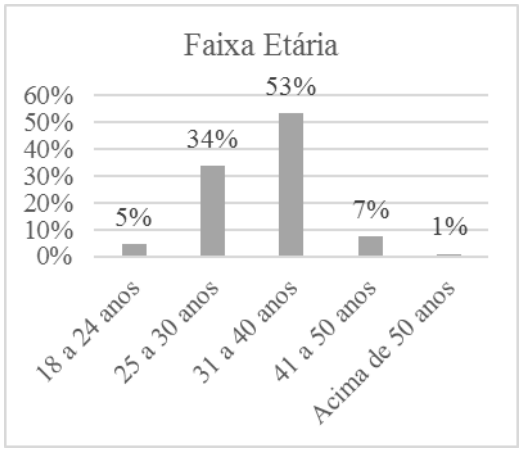


Figura 2 - Distribuição dos dados por faixa etária

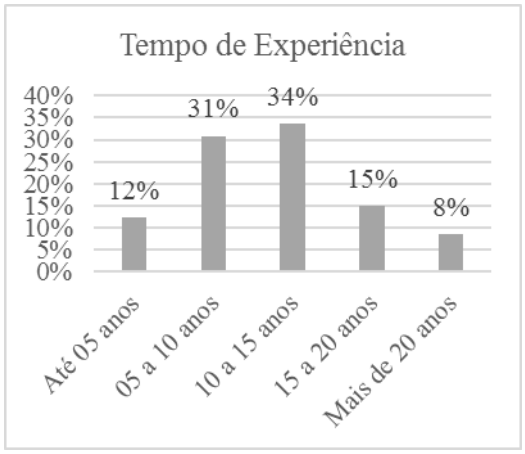


Figura 3 - Distribuição dos dados por tempo de experiência profissional

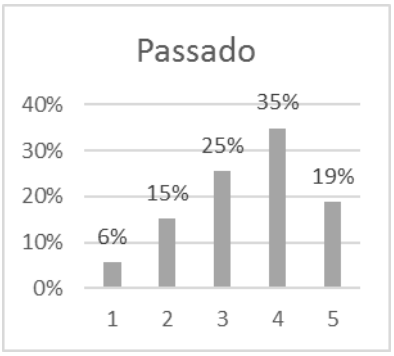


Figura 4 - Distribuição da auto-avaliação da empregabilidade no início da carreira

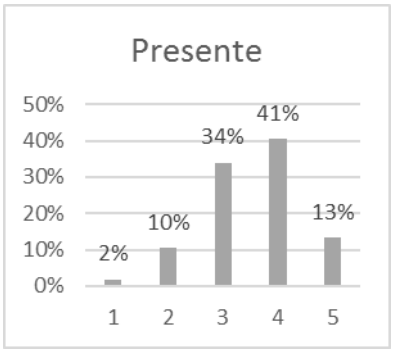


Figura 5 - Distribuição da auto-avaliação da empregabilidade no momento atual

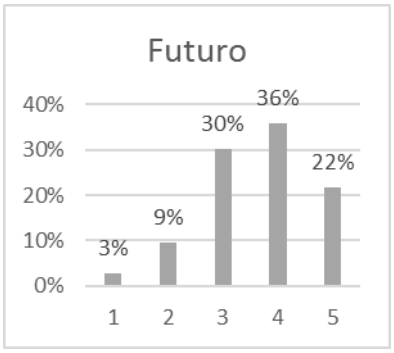


Figura 6 - Distribuição da auto-avaliação da empregabilidade daqui a 10 anos

Ainda sobre os fatores empregáveis, aproximadamente 31% dos entrevistados apontaram apenas fatores internos como determinantes para a sua empregabilidade, 8% apenas fatores externos e 61% dos entrevistados ambos os fatores.

4.2. Testes de hipóteses

Inicialmente buscamos verificar a normalidade dos dados. O próprio teste de Wilcoxon revelou que os dados se aproximam da distribuição normal de forma suficiente. Para avaliar a primeira hipótese, tomamos todos os respondentes e executamos o teste de Wilcoxon, comparando as respostas de passado→presente (*Z=-04*, *p= 0.67*), passado→futuro (*Z= -11, p=0.25*) e presente→futuro (Z= -0.88, *p=0.37*). Em nenhum dos três casos houve diferenças considerando um nível de significância de 95%. Consequentemente, a hipótese H1 foi confirmada.

> *H1: A auto-avaliação da empregabilidade dos engenheiros de software é aproximadamente a mesma, em diferentes momentos da carreira. CONFIRMADA* ✓

Em seguida, separamos a amostra em dois grupos, de acordo com o tempo de serviço. Para grupo dos participantes com POUCA EXPERIÊNCIA, obtivemos os seguintes resultados quando comparados as suas auto-avaliações em diferentes momentos da carreira: passado→presente (*Z=310.5, p=.72*), passado→futuro (*Z=-2.09, p=.03**) e presente→futuro (*Z=-2.55, p=.01**). Já para o grupo de engenheiros EXPERIENTES, encontramos os seguintes valores: passado→presente (*Z=460.5, p=-0.87*), passado→futuro (*Z=616, p=-.20*) e presente→futuro (*Z=-1.65, p=.09*), revelando que em nenhum dos cenários há diferença significativa da avaliação da sua empregabilidade. Sendo assim, estes dados são suficientes para refutar a segunda hipótese:

> *H2: O tempo de serviço é um fator que modera a auto-avaliação da empregabilidade de um momento anterior para um momento posterior. REFUTADA* ✗

No entanto, estes dados revelam que H2 seria verdade apenas para profissionais com pouca experiência, uma vez que para este grupo, apenas para o par passado→presente a diferença não se revelou significativa. Passado→presente faz sentido não serem diferentes, no entanto uma vez que, neste grupo em particular, o tempo que separa o passado do presente tende a ser muito curto. Portanto, ao analisar estes dados, descobrimos colateralmente que:

> *H2': O tempo de serviço é um fator que modera a auto-avaliação da empregabilidade de um momento anterior para um momento posterior em profissionais pouco experientes. CONFIRMADA* ✓

Para avaliar a terceira hipótese, que tratava da correlação entre experiência e percepção do *lócus* de controle sobre os fatores de empregabilidade, utilizamos o teste do qui-quadrado, com os dados descritos na Tabela 1, apresentada a seguir. O resultado do teste do qui-quadrado (χ*=25.40, p=0.001*) indicou uma relação significante entre as duas variáveis. Portanto, temos:

> *H3: O tempo de serviço está correlacionado com o lócus de controle. CONFIRMADA* ✓

Tabela 1 - Dados para cálculo do Qui-quadrado

	Lócus de controle			
	INTERNO	AMBOS	EXTERNO	Totais
Menos de 5 anos	7 (4.05) [2.15]	0 (7.24) [7.24]	6 (1.72) [10.68]	13
5 a 10 anos	8 (10.27) [0.50]	22 (18.37) [0.72]	3 (4.36) [0.42]	33
10 a 15 anos	10 (10.90) [0.07]	23 (19.48) [0.64]	2 (4.62) [1.49]	35
15 a 20 anos	5 (4.98) [0.00]	10 (8.91) [0.13]	1 (2.11) [0.59]	16
Mais de 20 anos	3 (2.80) [0.01]	4 (5.01) [0.20]	2 (1.19) [0.55]	9
Totais	33	59	14	106

Por fim, para avaliar a última hipótese, dividimos a amostra em grupos de profissionais de acordo com o seu *lócus* de controle da auto-avaliação da empregabilidade. Para profissionais com *lócus* INTERNO (n=33), obtivemos os seguintes resultados: passado→presente (*Z=-.05, p=.56*), passado→futuro (*Z=-2.28, p=.022**) e presente→futuro (*Z=-2.11, p=.03**). O grupo de profissionais com *lócus* de

controle EXTERNO (n=8) não se configurou suficientemente grande para fazer os cálculos estatísticos. Já para o grupo de profissionais com *lócus* de controle AMBOS (n=65), obtivemos os seguintes resultados: passado→presente (*Z=-0.66, p=.50*), passado→futuro (*Z=-.11, p=.91*) e presente→futuro (*Z=-1.00, p=.31*). Sendo assim, estes dados indicam que H4 é verdade, uma vez que profissionais que consideram apenas fatores internos tendem a avaliar mais positivamente um momento posterior da carreira do que momentos anteriores, enquanto profissionais que enxergam a interação entre fatores internos e externos tendem a não apresentar diferença significante da sua auto-avaliação de momentos distintos. Logo,

> *H4: O lócus de controle da empregabilidade modera a auto-avaliação da empregabilidade de um momento anterior para um momento posterior. CONFIRMADA* ✓

A nossa interpretação de H3 (✓) e H4 (✓), em conjunto, é de que quanto mais maduro o profissional, mais consciente ele fica de que a sua empregabilidade não depende apenas do seu próprio esforço, mas também de outras variáveis que fogem ao seu controle. O que também se mostra consistente com (H2' ✓), uma vez que o tempo de serviço é um fator que modera a auto-avaliação da empregabilidade de um momento anterior para um momento posterior em profissionais pouco experientes, e consequentemente com *lócus* de controle prioritariamente interno (H3 ✓)

6. Considerações Finais

A empregabilidade do profissional da engenharia de software é objeto de vários estudos. Parte deles se preocupa com a preparação do indivíduo para o mercado, trabalhando habilidades e competências requeridas pela indústria naquele momento. Contudo, o ingresso no mercado de trabalho é apenas uma das etapas da empregabilidade, sendo a manutenção e o desenvolvimento profissional de responsabilidade do então empregado. A avaliação equivocada acerca da própria empregabilidade pode levar o profissional à obsolescência e posteriormente à perda do grau empregável. Diferente de outras profissões, conhecimentos técnicos e superficiais, adquiridos na academia, tendem a se dissolver num curto período de tempo para engenheiros de software. Dessa maneira é altamente recomendável a reavaliação e renovação constantes das habilidades e conhecimentos específicos para a própria área de atuação, independente do estágio em que o profissional se encontra atualmente.

O objetivo deste trabalho foi, a partir de um amplo survey com profissionais da engenharia de software, entender a auto percepção dos mesmos acerca da empregabilidade e dos fatores considerados influentes para tal. A partir das respostas, foi possível observar uma impressão otimista do futuro em profissionais menos experientes. No entanto, essa visão otimista pode ser enviesada pela não percepção de que fatores externos ao controle do indivíduo também influenciam na sua empregabilidade real. Já para os profissionais mais experientes (que possuem mais de 10 anos no mercado), não é observada discrepância entre a sua avaliação de empregabilidade em diferentes momentos da carreira.

Como trabalho futuro, pretende-se realizar uma pesquisa acerca das diferenças curriculares entre as grades oferecidas no passado e as grades atualizadas dos cursos de ciências da computação e cursos correlatos, como forma de verificar se profissionais já

consolidados no mercado possuem familiaridade com as técnicas, metodologias de trabalho e tecnologias ensinadas atualmente na academia e medir o quanto isso será determinante para a sua empregabilidade.

Referências

Ang, S., and Slaughter, S. "The Missing Context of Information Technology Personnel: A Review and Future Directions for Research", *Framing the Domains of IT Management: Projecting the Future From the Past,* R. W. Zmud (ed.), Pinnaflex Educational Resources, Cincinnati, OH, (2000): 305-327.

Dubin, Samuel S. "Obsolescence or lifelong education: A choice for the professional." *American Psychologist* 27.5 (1972): 486

Easterbrook, S., Singer, J., Storey, M. A., & Damian, D. (2008). Selecting empirical methods for software engineering research. In *Guide to advanced empirical software engineering* (pp. 285-311). Springer London.

Fu, Jen-Ruei. "Understanding career commitment of IT professionals: Perspectives of push–pull–mooring framework and investment model."*International Journal of Information Management* 31.3 (2011): 279-293.

Jefferson, Theresa L. "Taking it personally: personal knowledge management." *VINE* 36.1 (2006): 35-37.

Joseph, Damien, and Soon Ang. "The threat-rigidity model of professional obsolescence and its impact on occupational mobility behaviors of IT professionals." *ICIS 2001 Proceedings* (2001): 73.

Joseph Damien, Mei Ling Tan, and Soon Ang. "Is Updating Play or Work?." (2011).

Kaufman, Harold G. "Obsolescence & Professional Career Development." (1974).

McGrath, Simon. "What is employability." *Learning to support employability project paper* 1 (2009): 15.

McQuaid, Ronald W., and Colin Lindsay. "The concept of employability."*Urban studies* 42.2 (2005): 197-219.

Miller, Craig S., and Lucia Dettori. "Employers' perspectives on it learning outcomes." *Proceedings of the 9th ACM SIGITE conference on Information technology education*. ACM, 2008.

Rosner, Bernard, Robert J. Glynn, and Mei-Ling T. Lee. "The Wilcoxon signed rank test for paired comparisons of clustered data." *Biometrics* 62.1 (2006): 185-192.

Saravanan, V. "Sustainable employability skills for engineering professionals." *The Indian Review of World Literature in English* 5.2 (2006): 1-9.

Swigon, Marzena. "Personal Knowledge Management (PKM) and Personal Employability Management (PEM)–Concepts Based on Competences."*Proceedings of The 3rd European Conference on Intellectual Capital held at University of Nicosia, Cyprus*. 2011.

Apêndice A – Questionário

Empregabilidade de profissionais de Tecnologia da Informação (TI)

Empregabilidade é a capacidade individual de obter um emprego, manter-se empregado ou mudar de emprego, se necessário, e idealmente ter um trabalho gratificante, sustentável e seguro. Gostaríamos de saber a sua percepção pessoal sobre sua empregabilidade, em diferentes estágios da sua carreira profissional.

São apenas 6 perguntas, que não tomarão mais do que 2 minutos do seu tempo.

Faixa etária
( ) 18 a 24 anos
( ) 25 a 30 anos
( ) 31 a 40 anos
( ) 41 a 50 anos
( ) Acima de 50 anos

Tempo de profissão
( ) Até 5 anos
( ) De 5 a 10 anos
( ) De 10 a 15 anos
( ) De 15 a 20 anos
( ) Acima de 20 anos

Como você enxergava a sua empregabilidade, quando saiu da universidade?

Muito baixa				Muito alta
1	2	3	4	5

Como você avalia a sua empregabilidade atual?

Muito baixa				Muito alta
1	2	3	4	5

Como você classificaria a sua empregabilidade daqui a 10 anos?

Muito baixa				Muito alta
1	2	3	4	5

Quais fatores você acredita exercer mais influência na empregabilidade de um profissional de TI?
[ ] Mercado de trabalho
[ ] Legislação trabalhista
[ ] Experiência profissinal
[ ] Domínio de tecnologias ou plataformas de trabalho modernas
[ ] Formação acadêmica, certificados e títulos
Outros: _____

Construção de Plataformas Digitais durante o Ensino de Engenharia de Software: um Relato de Experiência

Simone S. R. Souza[1], Bruno H. Oliveira[2], Filipe Grillo[2], Christian De Cico[2]

[1]Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo - ICMC/USP
Av. Trabalhador São-carlense, 400 – 13566-590 – São Carlos – SP

[2]Arquivei Serviços On Line Ltda
Av. Dr. Carlos Botelho, 1869 – São Carlos – SP

srocio@icmc.usp.br, {bruno,filipe, christian}@arquivei.com.br

Abstract. *Teaching Software Engineering involves all the concepts of software production, exploring the different development processes and project management activities. During the teaching of this discipline, the main challenges are showing in practice the importance of the software engineering to the quality of the software produced, as well as keeping the students motivated. This paper reports the experience of adopting a theoretical and practical methodology for teaching of software engineering course from the Department of Computer Systems (ICMC/USP). The proposed methodology adopts the Capstone projects approach in which students developed digital platforms, in partnership with a startup. The methodology allowed the students to apply in practice the concepts regarding software engineering, realizing its importance and at the same time, creating real software products with high impact to the Brazilian society*

Resumo. *O ensino de Engenharia de Software envolve todos os conceitos sobre construção de software, caracterizando os diferentes processos de desenvolvimento e as atividades de gestão de projetos. Durante o ensino dessa disciplina, os principais desafios são mostrar na prática o quanto que a Engenharia de Software é importante para a qualidade do software produzido, como também manter os alunos motivados. Nesta direção, este artigo relata a experiência em adotar uma metodologia teórico-prática durante o ensino da disciplina de Engenharia de Software para o curso de Engenharia de Computação do Departamento de Sistemas de Computação do ICMC/USP. A metodologia proposta adota a abordagem Capstone projects na qual os alunos desenvolveram plataformas digitais, com a parceria de uma startup. A metodologia permitiu que os alunos pudessem aplicar na prática os conceitos a respeito da Engenharia de Software, percebendo sua importância e, ao mesmo, tempo, gerando produtos de software reais e de impacto para a sociedade.*

1. Introdução

A engenharia de software é uma área que aplica os conceitos de engenharia para produzir software com maior qualidade e com menor custo. Ao longo de 45 anos, várias metodologias de apoio, incluindo processos, métodos, técnicas e ferramentas foram propostas

visando dar o suporte necessário para que os engenheiros de software atinjam seu objetivo. Aliado a isso, diferentes processos de desenvolvimento de software e atividades de gestão de projetos são propostas [Sommerville 2011].

No contexto de ensinar engenharia de software, é necessário apresentar aos estudantes um panorama dessa área, oferecendo uma visão geral dos processos de desenvolvimento de software, das atividades de desenvolvimento e de apoio, por exemplo, atividades de gestão de projetos, garantia de qualidade e de gerenciamento de configuração.

O desafio imposto é discutir esses assuntos aliando a teoria com a prática, de modo a construir um conhecimento sólido e crítico. Esse aspecto é desafiador pois nem sempre é possível ilustrar adequadamente todas as metodologias existentes na engenharia de software com a profundidade necessária. O desafio é encontrar um problema de software complexo o suficiente para que faça sentido utilizar os conceitos de engenharia de software e, ao mesmo tempo, simples o suficiente para que seja compreendido e possa ser adotado no âmbito de uma disciplina. Nesse aspecto, o currículo na área de Engenharia de Software [ACM/IEEE 2004] destaca a necessidade de mover-se para além do formato de aula expositiva. Prikladnicki et al [Prikladnicki et al. 2009] descrevem as abordagens mais comuns para ensinar engenharia de software, incluindo a substituição de aulas expositivas por discussão de casos práticos [Gnatz et al. 2003], dinâmicas de grupo, jogos [v. Wangenheim and Shull 2009] e *Capstone projects* [Goold and Horan 2009]. Segundo a IEEE/ACM [ACM/IEEE 2004], *Capstone projects* é que uma abordagem em que grupos de alunos planejam e executam um projeto de software do inicio até o fim durante um semestre inteiro. Assim, é uma abordagem interessante que permite a internalização de conhecimento e habilidades por meio do *learning by doing*.

Este artigo apresenta uma metodologia utilizada para ensinar Engenharia de Software por meio de *capstone projects*. A metodologia foi aplicada no 1^o semestre de 2016 a uma turma do 4^o ano do curso de Engenharia de Computação, durante a disciplina de Engenharia de Software do Departamento de Sistemas de Computação do ICMC/USP. Durante a disciplina, os alunos desenvolveram um projeto real, simulando em sala de aula um ambiente real de desenvolvimento de software. Para o desenvolvimento dos projetos, os alunos utilizaram o método ágil SCRUM [Schwaber and Beedle 2001]. Esse método foi escolhido por ser iterativo, evolutivo, adaptável e flexível. Ele inclui práticas que promovem agilidade e rapidez nas mudanças durante o desenvolvimento do projeto, dentre elas a interação constante com o cliente. Os projetos desenvolvidos (cada equipe escolheu um projeto dentre dois) envolviam duas aplicações de carácter social e inovadoras, proposta pela empresa parceira que participou da disciplina. Essa metodologia de ensino proporcionou que os alunos pudessem de fato 'viver' a engenharia de software e não somente aprendê-la na teoria, tornando-os mais motivados e interessados nas aulas de engenharia de software.

O artigo está organizado da seguinte forma: na Seção 2 são discutidos alguns trabalhos que se relacionam com a metodologia proposta neste artigo. Na seção 3 é descrita a metodologia de ensino adotada para a disciplina de engenharia de software. Na Seção 4 são apresentados os projetos escolhidos para a construção da plataformas digitais. Na Seção 5 são discutidos os resultados obtidos com a adoção da metodologia de ensino e na Seção 6 são apresentadas as conclusões e os desdobramentos deste artigo.

2. Trabalhos Relacionados

Prikladnicki et al [Prikladnicki et al. 2009] descrevem um conjunto de experiências de ensino em Engenharia de Software vivenciadas por quatro Instituições de Ensino Superior no Brasil. As experiências envolvem mudanças das formas de ensinar de modo a envolver cada vez mais os alunos no aprendizado. Dentre as experiências, uma delas se relaciona a proposta neste artigo que é o uso de *capstone projects*. Porém, a diferença é que na experiência relatada não existe a participação de empresa colaborando com projetos reais.

Billa e Cera [Billa and Cera 2012] propõem o uso da abordagem *PBL - Problem Based Learning* para o ensino de Engenharia de Software, focando em cursos de graduação de Engenharia de Software. Nessa abordagem os alunos solucionam problemas reais aplicando conceitos de engenharia de software.

Silva e Vasconcelos [Silva and Vasconcelos 2014] apresentam um ambiente integrado para apoio ao ensino de engenharia de software. Os autores destacam a importância da prática aliada ao ensino de engenharia de software, de modo a preparar os alunos para a realidade do mercado que é dinâmico e envolve tomada de decisões importantes.

Andrade et al [Andrade et al. 2015] descrevem uma abordagem de ensino para a disciplina de engenharia de software téorico-prático, melhorando a metodologia proposta anteriormente pelos próprios autores [Andrade et al. 2008]. Os autores realizam um survey com os estudantes para avaliarem a metodologia. Os resultados indicam que a metodologia proporcionou uma maior motivação aos estudantes. A diferença dessa metodologia, comparada àquela proposta neste artigo é que o projeto prático é desenvolvido extra-classe, com o uso de redes sociais para acompanhamento e discussão.

3. Metodologia de Ensino

Nesta seção é descrita a metodologia adotada na disciplina de Engenharia de Software do Departamento de Sistemas de Computação do ICMC/USP, ministrada para os alunos de Engenharia de Computação no 1^o semestre de 2016 (total de 32 alunos). A disciplina é obrigatória e é oferecida no 4^o ano do curso, possuindo quatro créditos-aula (sessenta horas-aula no semestre, divididas em duas aulas semanais). O objetivo desta disciplina é oferecer uma visão geral do processo de desenvolvimento de software e das técnicas que podem ser utilizadas em cada fase do ciclo de vida do software. Desta forma, o conteúdo da disciplina envolve ciclo de vida de software, processos e as fases do desenvolvimento, qualidade de software e gerenciamento de projetos. Na grade curricular do curso, esta disciplina ocorre posteriormente a disciplina de Análise e Projeto Orientado a Objetos, na qual os alunos já viram os conceitos de análise e projeto, focado no processo unificado.

A metodologia foi definida de modo que toda a teoria a respeito da engenharia de software fosse ministrada antes do início do desenvolvimento dos projetos. O objetivo principal foi o aluno ter uma visão geral e abrangente da engenharia de software antes da aplicação prática. Desta forma, a disciplina foi dividida em duas partes de igual quantidade de horas:

- Aulas teóricas: compreende aulas expositivas, exercícios (intra e extra-classe) e discussões em grupo, realizadas em sala de aula convencional;
- Aulas práticas: compreende o desenvolvimento do projeto desde a sua especificação até a sua implementação, com atividades intra e extra-classe. As

aulas foram realizadas em sala preparada para atividades em grupo, com mesas redondas que facilitam a interação entre as equipes.

Com relação às aulas teóricas, além do ensino dos conceitos téoricos, foram realizadas dinâmicas para capacitar os alunos em algumas habilidades necessárias para o desenvolvimento dos projetos. A primeira dinâmica se relaciona ao assunto Elicitação de Requisitos. Nessa dinâmica os alunos realizam grupos que representavam 'Clientes' e 'Desenvolvedores'. As equipes precisavam interagir por meio de entrevistas com o objetivo de extrair os requisitos do estudo de caso e estabelecer um acordo de preço e prazos para o desenvolvimento. A segunda dinâmica se relaciona com o método ágil Scrum [Schwaber and Beedle 2001]. Nessa dinâmica, chamada 'Hora Ágil', foi proposto o desenvolvimento de um protótipo de sistema, usando a técnica *paper prototype*, aplicando práticas do Scrum, tais como reuniões de *sprint planning* e *sprint review*, construção de *backlog do produto* e revisão com o cliente.

Antes de iniciarem as aulas práticas, os alunos conheceram os dois projetos, descritos na Seção 4. Os alunos se dividiram em cinco equipes e cada equipe escolheu um dos projetos propostos. Os alunos tomaram conhecimento também dos artefatos a serem produzidos (Tabela 1) e aprenderam o método ágil Scrum, adotado para o desenvolvimento dos projetos. Além disso, cada equipe escolheu um aluno para ser o *PO - Product Owner* da equipe, que é a pessoa que interage com o cliente. O ensino sobre o Scrum foi feito pela empresa, a qual já tem vasta experiência na adoção do mesmo. Optou-se por equipes de seis a sete alunos porque um dos objetivos foi ensinar aos alunos a habilidade de gestão do projeto, divisão de tarefas e acompanhamento. Além disso, esse tamanho de equipe favorece o uso do Scrum.

A disciplina contou também com o apoio de um aluno de doutorado da área de Engenharia de Software, que exercia o papel de monitor da disciplina, auxiliando nas atividades práticas da disciplina.

Tabela 1. Artefatos a serem entregues pelas equipes de cada projeto

Atividade	**Descrição**
Planejamento da Entrevista	Documento com a descrição do roteiro a ser seguido durante a entrevista com o cliente
Mapa conceitual (mind map)	Construção de um mapa contendo os conceitos sobre o projeto, derivados a partir da entrevista com o cliente
Protótipo em papel	Protótipo construído com base nos requisitos derivados da entrevista
Validação do Protótipo	Documento contendo o registro a respeito da validação do protótipo com o cliente
Plano de projeto	Documento contendo descrição do escopo e objetivos do projeto, matriz de riscos, organização da equipe para o desenvolvimento, cronograma, recursos utilizados, resultados esperados e recursos para monitoração e acompanhamento do projeto
Plano de testes	Documento contendo o que e como vai ser testado, os testes a serem aplicados, prazos e os recursos alocados
Backlog do produto	Construção da lista contendo todas as funcionalidades desejadas para o produto (requisitos) e descrição das funcionalidades na forma de histórias de usuário
Detalhamento de cada *sprint*	Construção da lista de funcionalidades do *sprint*, priorizando adequamente
Relatório de execução dos testes	Resultados da execução dos testes, com histórico de *bugs* encontrados

Conforme descrito na Seção 4, os projetos foram propostos pela empresa parceira. A empresa indicou sócios para atuarem como clientes de cada equipe. Isso permitiu a aplicação realista do método ágil Scrum, o qual prega interação constante com o cliente. A interação com o cliente foi realizada em algumas aulas e também extra-classe de acordo com as necessidades de cada equipe. Foram definidos *sprints* semanais, ou seja, um

conjunto de funcionalidades a serem desenvolvidas em cada semana. Assim, no final de uma semana, as equipes terminavam um *sprint* e planejavam o próximo. Em geral, as equipes desenvolveram os projetos em sete *sprints*.

Para construção do *backlog do produto* e de cada *sprint*, cada equipe construiu um *kanban board* utilizando cartolinas e fichas para descrever as histórias de usuário e compor as funcionalidades e sua sequência de implementação. Na Figura 1 tem-se um exemplo de *kanban board* construído por uma das equipes.

Figura 1. Exemplo de *kanban board* construído pelos alunos

Para gerenciar os projetos, as equipes utilizaram a ferramenta *Redmine*[1], que é gratuita e permite compartilhar o planejamento e evolução do projeto com a equipe e com os responsáveis pela disciplina. Além disso, essa ferramenta permite relacionar as atividades descritas com a implementação das mesmas por meio de ferramentas de controle de versões. Nos projetos, os alunos adotaram o *GitHub*[2] para o versionamento do projeto.

A interação com os alunos da disciplina ocorreu durante as aulas teóricas e práticas. Em particular, as aulas práticas permitiram estreitar a comunicação entre as equipes, o professor e monitor. Além disso, todo o material da disciplina, prazos e avisos foram disponibilizados no site da disciplina[3]. Foi utilizado também o *Facebook*[4] como um canal de comunicação, no qual um histórico das aulas práticas foi mantido por meio de fotos e materiais disponibilizados.

[1] http://www.redmine.org/
[2] https://github.com/
[3] disciplinas.stoa.usp.br
[4] http://facebook.com/

4. Projetos propostos: Plataformas Digitais

A escolha dos projetos a serem desenvolvidos na disciplina foi realizada após algumas interações com a empresa parceira, uma startup da cidade, da qual são sócios ex-alunos de graduação e mestrado do ICMC/USP. A preocupação principal era propor projetos desafiadores e, ao mesmo tempo, viáveis de serem desenvolvidos na disciplina. Outra preocupação foi aliar os interesses de ambos, no qual a startup sempre foca no produto final, enquanto que para a disciplina era importante avaliar o processo de desenvolvimento. Com base nessas premissas, os dois projetos reais propostos visam resolver dois problemas reais da sociedade brasileira: a escassez de doadores de sangue e a dificuldade de doação de notas fiscais para instituições sem fins lucrativos. Assim, nasceram duas propostas de projetos, descritos a seguir.

4.1. Projeto 1: Doação de Sangue

O projeto Doação de Sangue propõe o desenvolvimento de uma plataforma web para conectar organizadores de campanhas de doação de sangue e doadores, por meio de mídia sociais, como o Facebook. É conhecido que pessoas que já doaram sangue, quando conhecem campanhas de doação se sensibilizam em doar novamente e essa é a motivação para o desenvolvimento da plataforma. Na plataforma, qualquer usuário pode fazer o cadastro de uma campanha de doação de sangue e sugerir o tipo de sangue de que mais estão precisando. O doador, ao se cadastrar na plataforma por meio do Facebook (ou outra mídia social), é notificado pela plataforma sempre que uma nova campanha de doação for cadastrada na região em que ele mora Além disso, o doador pode acessar a plataforma e visualizar as campanhas que estão sendo realizadas em sua região, informando se tem interesse em participar de uma delas.

4.2. Projeto 2: Doação de Notas Fiscais Paulista

O projeto Doação de Notas Fiscais Paulista propõem o desenvolvimento de um aplicativo móvel e uma plataforma web para facilitar a doação de notas fiscais para entidades sociais. O aplicativo deve ler o *QR Code* existente na nota fiscal e extrair as informações necessárias, dentre elas o valor da nota. Para o projeto, fixou-se uma entidade social parceira para receber as doações. Essas informações devem ser enviadas para a plataforma web que possibilita que os funcionários da entidade social acessem esses dados, aceitem as doações e enviem para o governo. A motivação para o desenvolvimento desse projeto é que hoje as entidades sociais fazem esse cadastro de notas fiscais manualmente e o Governo só aceita a doação das notas que são registradas até o 20º dia do mês seguinte à emissão da nota. Ou seja, se a instituição não realizar o registro antes desse prazo, perde a chance de receber o benefício.

Os projetos propostos são desafiadores pois envolvem relacionar a plataforma com mídias sociais (Projeto 1), com o site do governo estadual (Projeto 2), desenvolvimento de plataformas Web (Projeto 1 e Projeto 2) e desenvolvimento de aplicativo móvel (Projeto 2). Vale ressaltar que os alunos não tinham feito disciplinas específicas sobre desenvolvimento Web e móvel. A empresa fez uma estimativa de quanto tempo seria necessário para o desenvolvimento dos projetos, considerando que os alunos estavam desenvolvendo e aprendendo ao mesmo tempo. Desse modo, o risco dos alunos não conseguirem finalizar o projeto a tempo foi mitigado por meio da delimitação do escopo dos projetos, definindo-se um conjunto mínimo de requisitos para cada projeto.

5. Resultados Obtidos

A partir da apresentação dos projetos propostos, as equipes escolheram seus projetos, sendo que duas equipes desenvolveram o Projeto 1 (Doação de Sangue) e três equipes desenvolveram o Projeto 2 (Doação de Notas Fiscais). Cada equipe tinha um cliente diferente e, portanto, cada uma seguiu com diferenças sutis de requisitos, o que proporcionou diferentes versões para os mesmos problemas. A empresa decidiu escolher o melhor produto de cada projeto, realizando uma premiação dos melhores projetos. Esse aspecto trouxe uma motivação a mais para as equipes em desenvolver projetos com qualidade e seguindo as premissas da engenharia de software.

Como a maioria dos alunos não tinham experiência com o método ágil, várias dificuldades foram encontradas, principalmente no início do projeto. Por exemplo, o primeiro *sprint* que as equipes definiram foram irrealistas e a maioria das equipes não conseguiu finalizá-lo em uma semana. De posse disso, as equipes analisaram suas dificuldades e passaram a definir *sprints* razoáveis de modo que conseguiram finalizá-los no espaço e tempo que tinham em uma semana de trabalho. Outra dificuldade foi conciliar o planejamento do projeto com atividades de outras disciplinas, o que dificultou bastante o desenvolvimento, principalmente das equipes que eram compostas por alunos que faziam disciplinas de diferentes anos do curso. Outra dificuldade a ser destacada é a falta de conhecimento dos alunos em desenvolvimento web e móvel. Essa dificuldade foi amenizada porque alguns alunos de cada equipe conheciam a respeito desses domínios de aplicação e puderam ensinar suas equipes.

Talvez pela simplicidade do método ágil, os alunos compreenderam rapidamente como trabalhar com ele. Percebeu-se que alguns alunos tinham mais facilidade para trabalhar com as premissas do método que envolvem agilidade, integração e interação contínua. Esse aspecto indica realmente que o sucesso do método ágil é fortemente relacionado com a cultura ágil presente na equipe de desenvolvimento, conforme já observado no trabalho de Oliveira [Oliveira and Souza 2014]. Isso foi observado em uma das equipes que não conseguiu aplicar com sucesso o método. No final, eles mesmos observaram o quanto isso prejudicou o desenvolvimento do projeto e a participação da equipe. Os próprios alunos concluiram que usar um processo de desenvolvimento é primordial para o sucesso do produto desenvolvido.

Apesar das dificuldades, todas as equipes conseguiram finalizar seus projetos no prazo previsto, produzindo projetos de alta qualidade. Os projetos foram apresentados no último dia de aula. A empresa foi convidada para avaliar os projetos e proceder com a escolha e premiação dos melhores projetos. Ressalta-se que a premiação foi independente da avaliação que considerou diferentes critérios. Mesmo assim, os projetos premiados também foram os projetos que receberam as maiores notas na disciplina. Para a apresentação dos projetos, foi solicitado às equipes um vídeo demonstrativo das suas plataformas e a construção de um *readme*, além de prepararem uma apresentação abordando essencialmente: 1) as decisões de projeto (ferramentas e tecnologias adotadas); 2) o que foi implementado daquilo que foi proposto, indicando partes que não foram concluídas; 3) o que mais poderia ser feito para melhorar o produto; e 4) o que não fariam ou fariam de maneira diferente para um novo projeto.

A partir dos depoimentos dos alunos, foi possível observar que a metodologia da disciplina enriqueceu o ensino da engenharia de software. A existência de um projeto real

e desafiador a ser desenvolvido, com um cliente real, expôs os estudantes a uma situação que eles possivelmente só encontrariam no mercado de trabalho. Dentre os depoimentos dos alunos, destacam-se os seguintes:

- 'Nós precisamos nos planejar bastante em relação ao tempo. Precisamos usar as ferramentas de organização de equipe (Redmine) para conseguir conciliar o projeto com todos os outros trabalhos e atividades da Universidade. Foi um aprendizado muito bom'.
- 'Essa é uma das coisas que mais me deixa feliz ao fazer o trabalho: aquilo que desenvolvemos não vai só virar uma nota da disciplina, vai ajudar pessoas e uma instituição que tem um trabalho muito legal'.
- 'A lista de inovações desse projeto foi imensa. Geralmente, nossos trabalhos são focados em uma determinada matéria. Dessa vez, realmente conseguimos integrar todos os conhecimentos em prol de um serviço à comunidade, uma plataforma pública'.
- 'A maior proposta da professora foi nos fazer viver a engenharia de software, não só estudar'.
- 'A disciplina possibilitou que tivéssemos contato com tecnologias e conceitos que não teríamos visto senão tivéssemos feito esse projeto. Isso é um gatilho para despertar o interesse por outras áreas'.

Na Figura 2 é ilustrado um dos aplicativos desenvolvidos para o projeto de Doação de Notas Fiscais. A Figura 3 apresenta um exemplo de tela de uma das plataformas desenvolvidas para Doação de Sangue. As apresentações de todos os projetos estão disponíveis em: `https://goo.gl/bU6mIJ`. As equipes premiadas foram convidadas pela empresa para disponibilizar para o público suas plataformas. Em breve, estarão disponíveis para uso.

Figura 2. Aplicativo desenvolvido para Doação de Notas Fiscais

Com relação à avaliação dos projetos na disciplina, notas individuais foram dadas aos alunos conforme a participação no projeto. Os critérios de avaliação envolviam três aspectos: 1) documentação do projeto (planejamento do projeto, protótipo construído, mapa conceitual, plano de testes e relatório da execução dos testes); 2) uso da metodologia de desenvolvimento (organização das atividades no redmine, no *kanban board*, divisão das tarefas pelos membros do grupo e uso adequado do Scrum para o gerenciamento do projeto); e 3) produto desenvolvido (organização do código no *Github*, se o atende os requisitos (porcentagem de funcionalidades), se está correto, qualidade em termos de usabilidade e manutenibiliade e apresentação do produto). Esses três aspectos

Figura 3. Umas das Plataformas para apoio à Doação de Sangue

foram computados para o cálculo da nota final do grupo. Para cada aluno do grupo, um fator multiplicador de valor máximo 1 foi usado para pontuar sua participação, considerando: quantidade de tarefas que o aluno implementou ou realizou, complexidade das tarefas do aluno e envolvimento das atividades da equipe. Essa avaliação individual foi possível pelo acompanhamento durante as aulas práticas e também pelas informações registradas no *Github* e *Redmine*. Na maioria dos casos, a participação dos alunos foi equilibrada nos grupos.

6. Conclusão

Este artigo apresentou uma metodologia empregada para o ensino de engenharia de software que alia teoria com prática, por meio da abordagem *Capstone projects*. Na aplicação dessa abordagem, os alunos desenvolveram, do início ao fim e aplicando conceitos de engenharia de software, plataformas digitais para Doação de Notas Fiscais Paulistas e Doação de Sangue. O objetivo foi aliar o conhecimento científico e teórico da academia com a vivência prática de uma startup (empresa parceira do projeto), que usa a metodologia ágil de desenvolvimento de software no dia a dia.

Ao aplicar a metodologia de ensino, pode-se observar que os alunos puderam colocar em prática as habilidades de dividir o problema a ser desenvolvido em atividades, priorizá-las e estabelecer cronogramas razoáveis para o desenvolvimento do projeto. O desenvolvimento dos projetos em sala de aula permitiu ao professor acompanhar e monitorar o seu desenvolvimento. Só o tempo em sala de aula não foi suficiente para desenvolver todo o projeto, mas uma boa parte foi feita em aula, contribuindo com a troca de experiências entre as equipes e o acompanhamento. Além disso, o uso de ferramentas colaborativas (como Redmine e GitHub) propiciaram aos alunos entender melhor sobre gestão de projetos e versionamento de projetos. Além disso, o professor pode acompanhar o desenvolvimento dos projetos e identificar o nível de participação dos alunos nos projetos.

Criar um ambiente virtual de desenvolvimento de software não é trivial, pois exige do professor conhecimento sobre as tecnologias e ferramentas que podem ser empregadas, como conhecimento sobre o processo de desenvolvimento escolhido. Sem dúvida, para o sucesso desse ambiente de aprendizagem é importante uma participação efetiva da indústria, que complementa o ensino com sua visão de mercado.

Os resultados da aplicação da metodologia de ensino proposta foram muito positivos e motivadores. Como continuidade deste trabalho, pretende-se replicar a meto-

dologia em outras turmas da disciplina, considerando novos projetos e novas parcerias. Outro desdobramento, é utilizar as plataformas digitais desenvolvidas em *Capstone projects* futuros, focando em outras atividades importantes da engenharia de software, como manutenção e garantia de qualidade de software.

Referências

[ACM/IEEE 2004] ACM/IEEE (2004). *Software Engineering Curriculum. Guidelines for Undergraduate Degree Programs in Software Engineering*.

[Andrade et al. 2008] Andrade, R. M. C., Marinho, F. G., Lelli, V., and Rocha, L. S. (2008). Metodologia para o ensino de engenharia de software. In *Proceedings of the I Forum de Educacao em Engenharia de Software*, FEES 2008.

[Andrade et al. 2015] Andrade, R. M. C., Santos, I. S., Araujo, I. L., and Carvalho, R. M. (2015). Uma metodologia para o ensino teorico e pratico da engenharia de software. In *Proceedings of the VIII Forum de Educacao em Engenharia de Software*, FEES 2015, pages 60–71.

[Billa and Cera 2012] Billa, C. Z. and Cera, M. C. (2012). Utilizando resolucao de problemas para aproximar teoria e pratica na engenharia de software. In *Proceedings of the V Forum de Educacao em Engenharia de Software*, FEES 2012.

[Gnatz et al. 2003] Gnatz, M., Kof, L., Prilmeier, F., and Seifert, T. (2003). A practical approach of teaching software engineering. In *Proceedings of 16th Conf. Software Eng. Education and Training*, IEEE CS Press, pages 120–128.

[Goold and Horan 2009] Goold, A. and Horan, P. (2009). Foundation software engineering practices for capstone projects and beyond. In *Proceedings of 15th Conf. Software Eng. Education and Training*, IEEE CS Press, pages 140–146.

[Oliveira and Souza 2014] Oliveira, B. H. and Souza, S. R. S. (2014). The agile quality culture - a survey on agile culture and software quality. In *Proceedings of International Conference on Software Engineering and Knowledge Engineering*, SEKE 2014, pages 1–4.

[Prikladnicki et al. 2009] Prikladnicki, R., Albuquerque, A. B., von Wangenheim, C. G., and Cabra, R. (2009). Ensino de engenharia de software: Desafios, estrategias de ensino e licoes aprendidas. In *Proceedings of the II Forum de Educacao em Engenharia de Software*, FEES 2009, pages 1–8.

[Schwaber and Beedle 2001] Schwaber, K. and Beedle, M. (2001). *Agile Software Development with Scrum*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

[Silva and Vasconcelos 2014] Silva, S. V. and Vasconcelos, A. P. P. (2014). Ambiente integrado como apoio ao ensino da engenharia de software. In *Proceedings of the VII Forum de Educacao em Engenharia de Software*, FEES 2014.

[Sommerville 2011] Sommerville, I. (2011). *Engenharia de Software*. Pearson Brasil, 9a ed. edition.

[v. Wangenheim and Shull 2009] v. Wangenheim, C. G. and Shull, F. (2009). To game or not to game? *IEEE Software*, 26(2):92–94.

Da Teoria à Prática em Desenvolvimento de Jogos Digitais: Um Estudo Sobre os Modelos de Processo Utilizados no Mercado Paraibano

José Raul de Brito Andrade, Vanessa Farias Dantas

Departamento de Ciências Exatas – Universidade Federal da Paraíba (UFPB) – Campus IV – Rio Tinto, PB – Brasil.

{raul.andrade, vanessa} @dce.ufpb.br

Abstract. *In recent years, the games have been gaining prominence in the world market. However, in IT courses, this area is still rarely addressed. In order to assist the academy to monitor developments in this area, this paper presents an investigation of the models currently used for game development. For this, it was made a review in the literature and it was verified how the game production in the Paraíba companies is conducted. With the collected data, it was possible to identify content that could be addressed by the academy to better prepare students for this market, as well as to benefit the enterprises, forming more prepared professionals.*

Resumo. *Nos últimos anos, os jogos vêm ganhando destaque no mercado mundial. Entretanto, nos cursos de Tecnologia da Informação, essa área ainda é abordada superficialmente. Com o intuito de auxiliar a academia a acompanhar a evolução dessa área, o presente trabalho apresenta uma investigação sobre os modelos utilizados atualmente para o desenvolvimento de jogos. Para isso, foi feito uma revisão na literatura e verificado como funciona, na prática, a produção de jogos nas empresas da Paraíba. Com os dados coletados, foram identificados conteúdos que poderiam ser abordados pela academia para preparar melhor os alunos para esse mercado, assim como, beneficiar as empresas, formando profissionais mais preparados.*

1. Introdução

Um jogo pode ser definido como uma atividade lúdica estruturada, composta por um conjunto de ações e decisões, delimitadas por regras bem definidas, para atingir um determinado objetivo [Schuytema 2008]. Devido à ludicidade e a necessidade constante de interação do usuário com a aplicação, os jogos estão sendo elaborados para outras finalidades, além do entretenimento, como: educação, publicidade, capacitação, entre outras [Rankin and Sampayo 2008].

Além disso, o mercado de jogos é uma das indústrias que mais cresce no mundo do entretenimento. Nos últimos anos, o Brasil tem se destacado nesse mercado, sendo atualmente o terceiro país que mais cresce por ano no ramo [Portal G1 2015]. Em um estudo do BNDES, publicado em 2014, o número de empresas de desenvolvimento de jogos no Brasil chegou a pouco mais de três vezes o registrado em 2008, de acordo com a Associação Brasileira de Desenvolvedores de Jogos Digitais (ABRAGAMES) [Fleury

et al. 2014]. O avanço nesse mercado se dá principalmente pelos maiores investimentos na área, além de fatores como a evolução das plataformas, tipos de jogos e melhoramento das metodologias de desenvolvimento.

Entretanto, apesar dos avanços, a área de jogos ainda é pouco abordada nos cursos de Tecnologia da Informação (TI). Isso pode refletir tanto na qualidade dos profissionais que estão sendo formados, como na desmotivação dos alunos pela área, que poderia ser mais uma possibilidade de carreira. Com o intuito de auxiliar a academia a abordar a área de jogos, o presente trabalho apresenta uma investigação sobre a teoria e prática em modelos de processo para o desenvolvimento de jogos, evidenciando os componentes importantes que poderiam ser trabalhados por ela.

Para a realização deste estudo, primeiramente foi feito uma revisão de literatura sobre os modelos de processo já propostos para o desenvolvimento de jogos. Em seguida, foi aplicado um questionário com membros de algumas empresas desenvolvedoras de jogos da Paraíba com o intuito de saber quais são as metodologias de desenvolvimento adotadas por elas, além de identificar habilidades e conhecimentos fundamentais para quem pretende seguir nessa área.

Com os dados coletados, foi identificado um conjunto de conteúdos que poderiam ser trabalhados pela academia para preparar melhor os alunos para o mercado em questão, assim como, beneficiar as empresas do ramo, formando profissionais mais preparados para essa área. Esses dados foram organizados de modo a servir de material de apoio no estudo do desenvolvimento de jogos, verificando também como funciona a teoria e prática desse processo.

Este artigo foi dividido em 5 seções: a primeira delas contextualiza a temática, e a segunda os resultados da revisão de literatura sobre os modelos de desenvolvimento para jogos. Na terceira seção, está descrita a abordagem metodológica adotada, na quarta seção são apresentados os dados coletados nas empresas paraibanas desenvolvedoras de jogos e na quinta seção são apresentadas as considerações finais sobre o trabalho.

2. Modelos de Desenvolvimento de Jogos

Por ser uma categoria de software, o jogo, assim como os outros tipos de aplicação, se faz necessário o uso das práticas de engenharia de software, por mais que os métodos utilizados sejam distintos [Sommerville 2011]. O processo de produção de um jogo é composto por um conjunto característico de etapas, que busca dar suporte às atividades desse tipo de desenvolvimento. Nessas etapas, são abordados aspectos, como: narrativa, mecânica, arte, documentação, programação e testes, além das especificidades de cada tipo de jogo.

O ciclo de vida do processo de produção de um jogo apresenta cinco fases principais: Concepção, Pré-produção, Produção, Pós-produção e Pós-lançamento [Sloper 2002]. Os principais papéis assumidos durante sua execução são: produtor, artista, *game deginer*, programador, engenheiro de áudio e engenheiro de qualidade [Olsen 2003]. A seguir, na Figura 1, são apresentadas as fases do processo de produção de um jogo, os artefatos gerados, e a atuação dos papéis durante o processo.

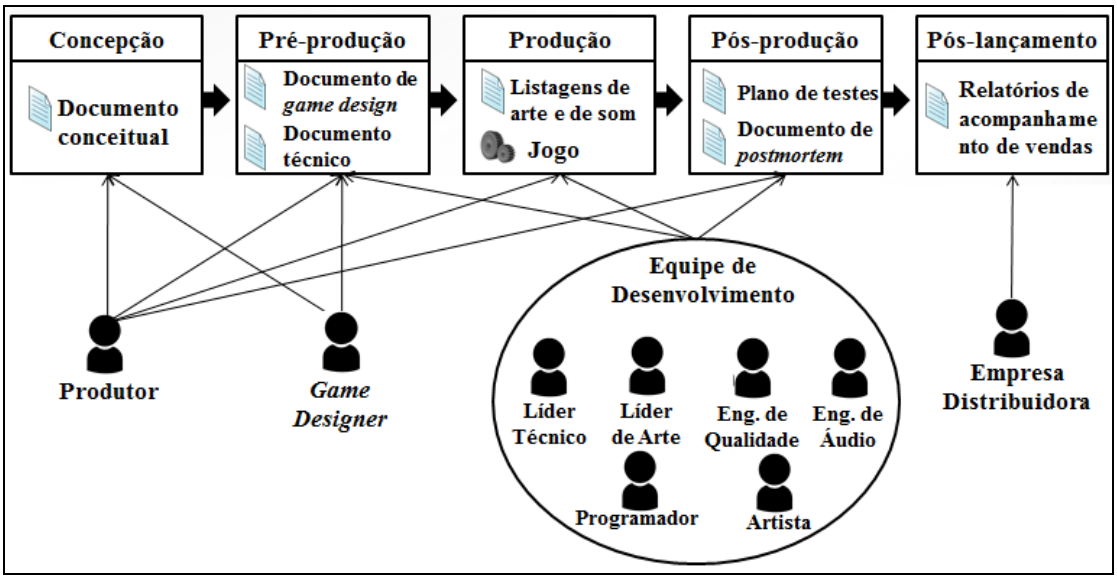


Figura 1. Visão geral do processo de desenvolvimento de jogos

Fonte: Barros, 2009

Devido a suas particularidades, o desenvolvimento de jogos apresenta alguns desafios. De acordo com Barros (2007), os problemas mais comuns são: grande dinamicidade de elementos fundamentais do projeto, e equipes multidisciplinares. Assim sendo, os modelos tradicionais de desenvolvimento de software não conseguem dar o suporte necessário para esse tipo de projeto, fazendo-se necessária a elaboração de modelos de desenvolvimento específicos para essa finalidade [Barros, 2007].

Alguns autores vêm propondo adaptações de modelos de processos da Engenharia de Software para serem utilizados no desenvolvimento de jogos, com intuito de atender às especificidades do desenvolvimento em questão, além de sugestões de ajustes nas etapas, papéis e artefatos. Entre os modelos de desenvolvimento de jogos propostos na literatura, podem ser destacados o *Game Waterfall Process*, o *Game Unified Process* e o *Extreme Game Development*.

2.1. *Game Waterfall Process*

O *Game Waterfall Process* (GWP) é a versão do tradicional modelo de desenvolvimento de software Cascata [Royce 1970] para jogos. Nessa adaptação, foi mantida a estrutura sequencial de execução de suas fases, não permitindo avanços, retornos ou sobreposições entre elas, e só iniciando uma fase quando a anterior está completa e correta.

Além das fases, no GWP foram propostas adaptações nas atividades que compõem o modelo original. As principais mudanças foram na documentação do produto, onde foram incluídos os documentos de dinâmica (*Game Design*) e aspectos técnicos (*Design Bible*) do jogo, e na forma de realizar os testes. Apesar de bastante utilizado, o GWP apresenta diversos problemas, ocasionados principalmente devido à linearidade intrínseca de suas fases. Em projetos nos quais mudanças ocorrem imprevisivelmente, utilizar esse modelo pode resultar em fracasso, tanto pela não adequação ao escopo, quanto pelo não cumprimento de custos e prazos [Flood 2003].

2.2. *Game Unified Process*

O *Game Unified Process* (GUP) foi um modelo desenvolvido em 2003 pelo gerente de projetos de jogos Kevin Flood, com o intuito de unir características presentes no RUP e no XP [Flood 2003]. O que motivou a elaboração desse modelo foram os problemas gerados com o GWP, devido às suas atividades sequenciais. Essas atividades vão de encontro com o que é defendido em ambos os modelos que foram base para o GUP, pois estes defendem que o desenvolvimento de software é um processo iterativo e não linear.

No GUP são adotadas práticas do XP, como a realização de teste contínuo durante todo o projeto, junto com a forte documentação proposta pelo RUP [Brauwers 2011]. Além disso, ele apresenta fases curtas, o que auxilia na fluidez da comunicação entre as equipes de desenvolvimento e arte. O criador do GUP, apesar de propor essa metodologia, não detalha seu funcionamento. Acredita-se que o principal objetivo dessa metodologia é mostrar que o Cascata não é um modelo adequado para esse tipo de desenvolvimento, e que modelos incrementais e ágeis podem ser mais apropriados para lidar com projetos dinâmicos e suscetíveis a mudanças.

2.3. *Extreme Game Development*

O *Extreme Game Development* (XGD) é uma metodologia ágil para desenvolvimento de jogos, criada pela Titus, empresa francesa de desenvolvimento de jogos, baseada no modelo idealizado por Kent Beck, o XP [Beck 2004]. Ele foi elaborado a partir da necessidade de adaptar algumas práticas do modelo original para outros papéis que compõem uma equipe de desenvolvimento de jogos, como artistas, *designers* e programadores, pois, por ter sido projetado por programadores, o XP tem maior foco no desenvolvimento e não dá suporte às necessidades de uma equipe multidisciplinar [Barros 2007].

Nessa adaptação, foram mantidos os valores (*feedback*, comunicação, simplicidade, coragem e respeito [Beck 2004]) e boa parte das práticas, algumas com modificações.Além dessas e de outras práticas do XP, o XGD adota técnicas do RUP, como a utilização de UML. Isso torna o modelo em questão fácil de adaptar para equipes de diferentes contextos. No entanto, a desvantagem de utiliza-lo é que, assim como o XP, existe pouca preocupação com a documentação. Dessa forma, os artefatos produzidos podem não ser suficientes para dar o suporte necessário aos envolvidos no projeto.

2.4. Modelos de Desenvolvimento para Aplicações do Jogo

Além das características comuns de jogos, um aspecto que também pode influenciar na metodologia, principalmente no que se refere ao levantamento de requisitos, é o propósito para o qual ele está sendo desenvolvido, classificado por Novak (2010) como aplicação de jogo. Dentre as aplicações possíveis, estão: Educação, Conscientização, Recrutamento e treinamento e *Marketing*.

Na literatura, são encontradas cada vez mais propostas de modelos de processo para o desenvolvimento de jogos com determinado propósito. Essas iniciativas são elaboradas com o intuito de dar suporte às particularidades que cada tipo de jogo apresenta, como é o caso do *Agile Game Process* [Araújo 2006] para *advergames* (jogos publicitários), e o *Serious Games Unified Process* [Rodrigues *et al.* 2010] para jogos

educacionais. Porém ambas, apesar de bem documentadas, não foram utilizadas em um amplo número de projetos para medir seus resultados e a eficácia de seus processos.

3. Metodologia da Pesquisa

Este trabalho apresenta uma pesquisa qualitativa e exploratória. A escolha da abordagem exploratória se deu pelo fato dela proporcionar uma visão geral acerca do fenômeno estudado (Gil, 2010). A obtenção dos dados se deu através de uma revisão na literatura e da aplicação de questionários em empresas de desenvolvimento de jogos do estado da Paraíba.

3.1. Planejamento da Pesquisa

O planejamento da pesquisa foi composto pelas seguintes etapas: 1. Análise dos modelos propostos na literatura para o desenvolvimento de jogos, apresentado na seção anterior; 2. Pesquisa nas empresas paraibanas desenvolvedoras de jogos; e 3. Análise dos dados coletados. De acordo com o I Censo da Indústria Brasileira de Jogos Digitais [Fleury *et al.* 2014], realizado em 2014, existem cinco empresas desse segmento na Paraíba. Porém, não foram encontradas informações atualizadas sobre duas delas, o que leva a crer que possam não estar mais em atividade. Dessa forma, o convite para as empresas participarem da pesquisa foi realizado através de um grupo de discussão em uma rede social, formado por desenvolvedores de jogos da Paraíba, onde foram identificadas novas empresas. Aceitaram participar da pesquisa 7 empresas, sendo 5 delas de João Pessoa, e 2 de Campina Grande.

Na etapa de pesquisa nas empresas, foi aplicado um questionário composto por 15 questões para obtenção dos dados. O objetivo era identificar algumas das definições e práticas utilizadas nas empresas questionadas. O questionário era composto por questões abertas e com múltiplas escolhas, e foi enviado a gerentes de projetos ou desenvolvedores com amplo conhecimento dos processos de desenvolvimento de suas empresas.

Antes de ser aplicado, o questionário foi avaliado por três especialistas da área de Computação (um Mestre e dois Doutores em Ciência da Computação), e posteriormente aplicado experimentalmente com 5 alunos do curso de Licenciatura em Ciência da Computação da UFPB/ *Campus* IV, que trabalham em um programa institucional desenvolvendo jogos digitais educacionais.

3.2. Coleta de Dados

Para coletar os dados, inicialmente foi realizado um convite para as empresas locais, através de um grupo de discussão de desenvolvedores de jogos da Paraíba. Depois de identificadas, as empresas foram contatadas e inteiradas sobre os objetivos do estudo. A aplicação dos questionários ocorreu via e-mail, sendo retornados dentro do prazo de uma semana.

3.3. Análise dos Dados Coletados

Após a aplicação dos questionários, foi feita a análise dos dados baseado na pesquisa qualitativa, buscando conhecer os tipos de jogos desenvolvidos pelas empresas

paraibanas, assim como os aspectos relevantes de suas metodologias de desenvolvimento.

4. Modelos de Desenvolvimento de Jogos das Empresas Paraibanas

Para esta pesquisa foram entrevistadas sete empresas. Foi identificado que 85% delas trabalha com o desenvolvimento de jogos educacionais e 57% desenvolvem para plataformas móveis. No Quadro 1 são apresentadas informações gerais das empresas pesquisadas.

Quadro 1. Informações gerais das empresas pesquisadas

	Tempo de mercado	**Quantidade de funcionários de TI**	**Quantidade de jogos desenvolvidos**	**Tipos de jogos desenvolvidos**
Empresa A	Menos de 1 ano	Entre 3 e 4 funcionários	2 ou 3 jogos	Casual
Empresa B	Entre 5 e 6 anos	Entre 3 e 4 funcionários	6 ou 7 jogos	Educacional
Empresa C	Entre 1 e 2 anos	Entre 5 e 6 funcionários	4 ou 5 jogos	Casual, Educacional e *Advergame*
Empresa D	Entre 5 e 6 anos	Entre 5 e 6 funcionários	4 ou 5 jogos	Educacional
Empresa E	Entre 1 e 2 anos	Até 2 funcionários	4 ou 5 jogos	Casual, Educacional e *Advergame*
Empresa F	Entre 5 e 6 anos	Entre 5 e 6 funcionários	6 ou 7 jogos	Casual e Educacional
Empresa G	Entre 1 e 2 anos	Entre 7 e 8 funcionários	8 ou mais jogos	Educacional

Para obtenção dos dados foi aplicado um questionário. O questionário objetivou obter informações sobre práticas utilizadas nas empresas de jogos paraibanas. Ele foi dividido em módulos e as questões foram as seguintes: (os gráficos das questões podem ser acessados pelo link: https://goo.gl/FsJTuI)

4.1. Módulo I: Atividades

O objetivo das questões desse módulo era identificar como são realizadas as principais atividades da produção de jogos nas empresas entrevistadas.

1. Qual (is) grupo (s) é (são) consultado (s) e em qual (is) momento (s) da etapa de requisitos?

Na etapa de levantamento de requisitos, a participação do cliente e da equipe é comum em todas as empresas entrevistadas. Nesse processo, a participação do usuário não é tão presente. Porém, são consultados o legado (jogos semelhantes) e tendências de mercado, para que sejam identificados os aspectos que fazem mais sucesso. No caso da validação dos requisitos, os usuários e a equipe são os mais presentes. Em alguns casos, o cliente também participa dessa validação.

2. No caso de não haver um cliente específico.

a) Como é feito o levantamento dos requisitos?

"As regras de negócio do jogo são definidas pelas tendências de mercado ou sugestões da equipe". No caso dos jogos educacionais, assim como proposto no SGUP, um especialista na área do jogo educativo é chamado para consulta. Dentre as empresas entrevistadas, apenas uma afirmou trabalhar comumente com cliente específico.

b) Como é feita a validação dos requisitos?

"Dependendo do tipo de requisito, alguns são validados pela própria equipe, outros são validados com a comunidade formada ao redor do jogo, ou especialistas no caso de jogos educacionais". Geralmente são desenvolvidos protótipos, e eles são testados com alguns usuários.

3. Em que momento do projeto os testes costumam ser realizados e qual (is) tipo (s) de teste (s) são realizados?

Das empresas entrevistadas, em cinco os testes ocorrem no final de cada iteração e em três, no final de cada *release*. Apenas uma empresa afirma que realiza testes nos dois momentos. Os tipos de testes mais comuns, assim como proposto no GWP, são os testes com usuários (*PlayTests*). Outra prática comum desse tipo de desenvolvimento são os testes a partir do *feedback* de desenvolvedores de outras empresas de jogos.

4.2. Módulo II: Papéis

As questões desse módulo tiveram o objetivo de verificar os cargos e papéis desempenhados nas empresas entrevistadas, assim como a forma que eles são definidos e organizados.

1. Quais cargos são desempenhados na empresa?

Existe a presença unanime de três cargos fundamentais para o desenvolvimento de jogos, são eles: *game designer*, artista e programador. O cargo de gerente de projeto, apesar de não estar presente em todas as empresas entrevistadas, tem suas tarefas desempenhadas, em alguns casos, pelo *game designer*. Em relação ao cargo de testador, não existe um consenso. Em muitas empresas, esse não chega a ser um cargo específico, pois na própria ferramenta de desenvolvimento são realizados os testes. Além disso, é comum esse tipo de produto ser testado diretamente com o usuário.

2. Como funciona a atribuição de papéis?

Para essa pergunta, a maior parte das empresas respondeu que os funcionários executam tarefas além de seu cargo. Dentre os fatores que motivam essa prática, destacam-se: o quadro limitado de funcionários e a preferência das empresas por trabalhar com equipes menores, devido à complexidade de gerenciamento. Outros fatores também citados foram: a importância da interação entre os diferentes cargos para o produto final, a existência de papéis que não requerem cargos específicos, e a carência de profissionais no mercado. As atividades do testador normalmente não são de um cargo específico. Como dito anteriormente, o programador, os especialistas e alguns usuários realizam essa atividade.

4.3. Módulo III: Artefatos

Nesse módulo as questões tiveram o objetivo de identificar quais artefatos são produzidos durante a execução dos projetos de jogos das empresas entrevistadas.

1. Quais artefatos costumam ser desenvolvidos nos projetos de sua empresa?

Os artefatos produzidos variam entre as empresas. O documento de *game design* é o mais utilizado, por ter uma maior abrangência dos elementos do projeto. Pôde-se perceber que, de modo geral, a metodologia adotada nas empresas prioriza a simplicidade. Sendo assim, a documentação é objetiva e, em alguns casos, apesar do único documento produzido ser o de *game design,* é abordado nele mais informações do que o modelo original propõe.

2. Como é feita a modelagem do projeto dos jogos?

Quase todas as empresas entrevistadas realizam a modelagem do projeto de maneiras distintas. Uma das empresas desenvolveu uma ferramenta própria para essa finalidade, chamada *Game Design Tools,* enquanto as outras empresas trabalham com mapas de navegação e *wireframes* (protótipo de interface para sugerir a estrutura e os relacionamentos de uma aplicação).

3. Após lançar um jogo, a empresa gera relatórios de acompanhamento de vendas?

Nesta questão, apenas uma das empresas afirmou não gerar os relatórios de acompanhamento de vendas após o lançamento do jogo.

4.4. Módulo IV: Ferramentas

O objetivo das questões desse módulo era identificar quais as principais ferramentas utilizadas pelas empresas entrevistadas.

1. Informe qual (is) o (s) ambiente (s) é (são) utilizado (s) para as seguintes finalidades:

A) Ferramentas de edição de imagem

A ferramenta mais usada pelas empresas entrevistadas para edição de imagens foi o Photoshop. Sua escolha em projetos se dá pela consolidação da ferramenta no mercado e experiência de funcionários em utiliza-la. Também pelo número de usuários, algumas empresas adotam o *Illustrator* e o *3ds Max*. O *Coreldraw* é utilizado para desenhos vetoriais, e o *Gimp* por ser um software livre.

B) Ambientes de desenvolvimento

Todas as empresas utilizam alguma *game engine* (ferramentas com um conjunto de funcionalidades para facilitar o desenvolvimento de um jogo). Essas ferramentas facilitam o processo de desenvolvimento e testes. Para desenvolver os jogos, o ambiente mais utilizado é o *Unity*. Dentre os motivos citados para utiliza-lo estão: a praticidade, a facilidade em encontrar materiais de apoio, e o fato de ser multiplataforma. O *Construct* 2 também é utilizado por algumas empresas, devido à baixa curva de aprendizado e à possibilidade de desenvolvimento para diferentes plataformas, assim como o *Unity*.

C) Outras ferramentas

Outras ferramentas que são adotadas nos projetos de jogos são: o *Git,* para controle de versão, *e o Trello*, para o gerenciamento do projeto e controle de cronograma. Para gerenciar o cronograma também são utilizados o *Google Docs* e o Excel.

4.5. Módulo V: Práticas

As questões desse módulo tiveram o objetivo de identificar as principais práticas das metodologias adotadas nas empresas entrevistadas.

1. Como pode ser definida a metodologia adotada para desenvolver seus jogos?

As empresas adotam em seus projetos o *Scrum*, metodologia híbrida entre *Scrum* e XP, ou metodologias próprias, com características de metodologias ágeis. O fato da maior parte das empresas preferir adaptar os modelos ágeis originais se dá pela necessidade em adequar as etapas para membros da equipe que não são de TI, além de acrescentar algumas atividades relacionadas a conteúdo, no caso dos jogos educacionais.

2. Qual (is) fator (es) motiva (m) o uso dessa metodologia?

A produtividade e o tamanho da equipe são os principais motivadores. Pôde ser observada a necessidade de ser objetiva e ágil a metodologia de desenvolvimento adotada nessas empresas.

3. Quais das seguintes etapas estão presentes na metodologia adotada na sua empresa?

É comum em todas as empresas o *brainstorming* (reunião para definir a ideia do jogo), testes (validação do jogo) e desenvolvimento (implementação do jogo). Foi observado que a elaboração da *design bible* (documento com as especificações técnicas do jogo), apesar de não estar presente em todas as empresas, muitos de seus tópicos são abordados em outros documentos produzidos por elas.

4. Qual é a base da elaboração de um cronograma?

Foi identificado que nessa etapa é comum confiar na experiência dos membros, sem usar métodos mais formais de estimativa. Essa prática dificulta a mudança de equipes no projeto.

5. As equipes do projeto costumam se auto-organizar?

Apesar da maioria das empresas entrevistadas fazerem uso de características do *Scrum*, na maior parte delas as equipes precisam do gerente de projetos para organizar o andamento do projeto.

6. Como é realizada a comunicação entre os profissionais envolvidos no projeto?

As formas mais utilizadas para realizar a comunicação entre a equipe são: através de mensagens instantâneas e troca de *e-mails*. As reuniões presenciais ocorrem na maioria dos casos semanalmente, com exceção de duas empresas que realizam reuniões do tipo S*tand-up meetings* diariamente. Essas duas empresas também realizam diariamente as reuniões retrospectivas do projeto, e as demais realizam essa reunião apenas ao final de cada iteração.

7. Como é feita a comunicação com os clientes em um projeto na fase de desenvolvimento do sistema?

A comunicação com os clientes é feita principalmente por e-mail e reuniões regulares na própria empresa. Na maioria dos casos, a participação do cliente no processo de produção do jogo acontece nas reuniões, quando participam da etapa de requisitos.

5. Considerações Finais

O mercado de jogos digitais é um dos que mais cresce atualmente. O número progressivo de novas empresas desse segmento comprova esse fato. Com o crescimento, os projetos de jogos passam a ser mais caros e complexos, e torna-se necessária a adoção de medidas que elevem o nível de qualidade do produto final e diminuam a taxa de erros de projetos dessa natureza. Na comunidade acadêmica, essa área ainda é abordada de forma superficial, o que pode resultar em poucos e despreparados profissionais formados para o mercado, e produtos finais com pouca qualidade.

Buscando auxiliar as universidades no conhecimento sobre os processos para desenvolver jogos digitais, este trabalho foi idealizado para verificar os processos e as metodologias propostos na literatura e também utilizados nas empresas paraibanas. Assim, o objetivo deste trabalho foi a realização de uma análise destas metodologias para identificar quais aspectos desse tipo de desenvolvimento deveriam ser tratados nos cursos de TI. Dentre o que foi observado com esta pesquisa, considera-se fundamental ser abordado na academia que:

- A maior parte das empresas trabalha com desenvolvimento para plataformas móveis, por isso é importante trabalhar esse tipo de desenvolvimento.
- A documentação é fundamental nesses projetos. No entanto, o que produzido é especificamente para jogos digitais. Assim sendo, é importante abordar o documento conceitual, o documento de game design, e a *design bible*, destacando as diferenças e objetivo de cada um deles.
- A modelagem nesses projetos é feita a partir de mapas de navegação, *storyboards* e fluxogramas. A prática dessas técnicas deve ser estimulada.
- O uso de *game engines* é comum a todas as empresas. Essas ferramentas podem auxiliar no desenvolvimento, modelagem dimensional das imagens e nos testes. Por essa razão, é fundamental discuti-las ao se tratar de produção de jogos.

Além desses tópicos, algumas observações devem ser feitas em relação à abordagem do desenvolvimento de jogos em sala de aula. São elas:

- Os jogos mais desenvolvidos na Paraíba são os educacionais. Assim, é importante ressaltar a importância da presença de um especialista na área para a qual o jogo está sendo desenvolvido. Ele participa da etapa de definição do conteúdo, análise e validação dos requisitos.
- A definição e análise dos requisitos tem o diferencial de utilizar jogos semelhantes e tendências de mercado. É importante entender os elementos do jogo e formas de identificá-los para extrair os aspectos fundamentais para o produto a ser desenvolvido.

- Os testes podem ocorrer de três formas: (i) Teste pela *game engine*; (ii) Demonstrações a outros desenvolvedores; e (ii) Testes de funcionalidade e usabilidade – pela equipe, cliente e usuário. Essas diferentes formas de teste devem ser conhecidas e exploradas.
- As equipes geralmente são pequenas e multidisciplinares. Dessa forma, é importante o profissional ser proativo e entender, ainda que superficialmente, as atividades desenvolvidas por outros membros, mesmo fora de sua área de formação;
- Por ser uma documentação muito utilizada, deve-se destacar a importância do relatório de acompanhamento de vendas após o lançamento do jogo.

A partir deste estudo foi possível identificar as principais características dos modelos de desenvolvimento de jogos, assim como a importância de iniciativas que diminuam a lacuna existente entre a teoria e prática nessa área. Também foi observado que, no contexto das empresas paraibanas, é pouco trabalhado o que é proposto na literatura. Isso ocorre devido a não adequação das práticas propostas com a realidade das empresas, além de, em alguns casos, as empresas não possuem conhecimento sobre essas propostas. Espera-se que esse trabalho venha contribuir para a academia na abordagem do desenvolvimento de jogos, para preparar melhor os alunos para esse mercado e assim beneficiar as empresas, formando profissionais mais preparados. Espera-se também contribuir para a literatura sobre o tema, que tem se mostrado ainda deficiente em fornecer informações desta categoria de projetos de software.

A maior dificuldade encontrada na realização desta pesquisa foi a pouca disponibilidade de tempo dos entrevistados. Entretanto, os mesmos foram prestativos e colaboraram sugerindo informações que originalmente não eram abordados no questionário.

Dando continuidade ao trabalho, sugere-se: realizar um estudo aprofundado das principais características dos modelos de desenvolvimento de jogos com propósito (educação, treinamento, publicidade); realizar um estudo aprofundado sobre o perfil das empresas paraibanas de desenvolvimento de jogos; e produzir material instrucional sobre o tema para ser usado nas universidades.

Referências

Araújo, A. R. S. (2006) "*Agile Game Process*–Metodologia Ágil para Projetos de *Advergame*s", Trabalho de conclusão de graduação em Ciências da Computação, Universidade Federal de Pernambuco, Recife.

Barros, R. (2009) "O processo de desenvolvimento de jogos". Disponível: http://www.cin.ufpe.br/~game/aulas/gerencial. Acesso: Março/ 2016.

Barros, R. L. B. (2007) "Análise de metodologias de desenvolvimento de software aplicadas ao desenvolvimento de jogos eletrônicos", Trabalho de conclusão de graduação em Ciência da Computação, Universidade Federal de Pernambuco, Recife.

Beck, K. (2004) "Programação extrema explicada: acolha as mudanças". Porto Alegre: Bookman.

Brauwers, R. (2011) “Estendendo e instanciando o *game agile methods applied* (GAMA)”. Trabalho de conclusão de graduação em Ciência da Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre.

Fleury, A. Sakuda, L. O. Cordeiro, J. H. D. O. (2014) "I Censo da Indústria Brasileira de Jogos Digitais". Disponível: http://goo.gl/1f2XSM.

Flood, K. (2003) “*Game Unified Process* (GUP)”, GameDev.net *Articles*. Disponível: http://www.gamedev.net/reference/articles/article1940.asp. Acesso: Maio/ 2016.

Gil, A.C. (2010) “Como elaborar Projetos de Pesquisa”, São Paulo: Editora Atlas. 5ª.

Novak, J. (2010) “Desenvolvimento de games”, São Paulo: Cengage Learning, 354-355.

Olsen, J. (2003) “*Game Development Salary Survey*”. Gamasutra *Article*. Disponível: http://goo.gl/0bnbfO. Acesso: Maio/ 2016.

Portal G1. Mercado de *games* fatura cerca de US$ 1 bilhão por ano no Brasil. Diponível: http://goo.gl/t7gOND. Acesso: Março/ 2016.

Rankin, J. R., Sampayo, S. (2008) “A review of Serious Games and other game categories for Education”, SimTect 2008, Melbourne, Australia, pp. 305-311.

Rodrigues, H. F., Machado, L. S., Valença, A. M. G. (2010) "Definição e Aplicação de um Modelo de Processo para o Desenvolvimento de *Serious Games* na Área de Saúde", Proc. Congresso da Sociedade Brasileira de Computação-Workshop de Informática Médica.

Royce, W.W. (1970) “*Managing the development of large software systems: concepts and techniques*”, Proc. IEEE Westcon, Los Angeles, CA.

Schuytema, P. (2008) “*Design de games*: uma abordagem prática”, Cengage Learning.

Sloper, T. (2002) “*Following Up After the Game is Released: It’s not Over when it’s Over*”, *Game Design Perspectives*, p. 372.

Sommerville, I. (2011) “Engenharia de Software”, 9ª edição, Tradução: Selma Shin Shimizu Mel-nikoff, Reginaldo Arakaki, Edilson de Andrade Barbosa. São Paulo: Pearson Addison-Wesley, v. 22, p. 103.

Ensinando Melhoria de Processos de Software na Prática com a norma ISO/IEC 29110: Um Estudo de Caso

Jean Carlo Rossa Hauck[1], Richard Henrique de Souza[1, 2]

[1] Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC) – Florianópolis – SC – Brasil

[2] Universidade do Sul de Santa Catarina (Unisul) – Campus Grande Florianópolis Palhoça – SC – Brasil.

jean.hauck@ufsc.br, richardhenrique@gmail.com

Abstract. *The Software Engineering learning can often become a boring experience for students. However, experiments have shown that the practical usage of standards can support education in this area. The ISO/IEC 29110 was developed to be a reference model for quality processes in small organizations and has been successfully used to support education. Thus, this paper presents a practical experience of using the standard in the classroom, as a case study. Students are involved in processes assessment of real-world organizations. The student's reports rise early indications that the practical experience was positive in the motivation for the use of the standard.*

Resumo. *O aprendizado de Engenharia de Software muitas vezes pode se tornar uma experiência cansativa para os alunos. Entretanto, experiências têm demonstrado que o uso prático de normas pode apoiar o ensino nessa área. A norma ISO/IEC 29110 nasceu da necessidade de um modelo de referência para qualidade de processos em micro e pequenas organizações e tem sido utilizada com sucesso no apoio ao ensino. Assim, este artigo apresenta uma experiência prática de utilização da norma em sala de aula na forma de um estudo de caso. Os alunos são envolvidos na avaliação de processos em organizações reais. Os relatos dos estudantes levantam primeiros indícios de que a experiência foi positiva na motivação para o uso da norma.*

1. Introdução

Os altos índices de reprovações e desistências de alunos de Sistemas de Informação e Ciências da Computação têm sido uma preocupação recorrente da pesquisa em educação nessa área [Bosse and Gerosa 2015]. O ensino de Engenharia de Software nesses cursos, especialmente nas primeiras disciplinas, pode ser maçante para os alunos quando há somente uma abordagem teórico-expositiva, muitas vezes gerando desmotivação em relação ao ensino distanciado da prática [Bessa et al. 2012; Souza and Resende 2010].

No entanto, experiências têm mostrado resultados positivos na utilização de normas e modelos de referência para guiar atividades práticas no ensino de conceitos de Engenharia de Software [Laporte and Connor 2014]. A norma ISO/IEC 29110 [ISO/IEC 2012] foi desenvolvida a partir da percepção de que Micro e Pequenas Organizações

(MPOs) de software, em geral, não utilizam normas e modelos de referência para qualidade de processos [Laporte et al. 2008].

Um dos importantes conceitos atuais da Engenharia de Software é a Melhoria de Processos de Software, que consiste nas "medidas tomadas para mudar os processos de uma organização para que sejam executados de forma mais eficaz e/ou eficiente" [ISO/IEC/IEEE 2010]. Como o fator motivacional dos alunos tem forte influência sobre o aprendizado na área da computação [Souza and Resende 2010], uma abordagem atrativa para o ensino consiste no envolvimento dos estudantes em casos práticos, aplicando conceitos em problemas reais da indústria [Dingsøyr et al. 2000].

Assim, este artigo apresenta um estudo de caso de ensino de Melhoria de Processos de Software em turmas de Engenharia de Software, tomando por base uma abordagem prática de avaliação de processos de acordo com a norma ISO/IEC 29110.

O restante deste artigo está divido nas seguintes seções: a seção 2 elenca os trabalhos correlatos; a seção 3 mostra o planejamento, a realização e os resultados do estudo de caso enquanto a seção 4 apresenta as conclusões do trabalho.

2. Trabalhos Correlatos

Apesar da abordagem prática de ensino de conceitos de Melhoria de Processos de Software, por meio de casos reais aplicando normas, não ser muito comum na literatura, pode-se destacar o trabalho de Laporte e Connor (2014), que apresentam diversas aplicações nas quais a norma ISO/IEC 29110 foi utilizada no ensino de Engenharia de Software. Realizando interações com a indústria em diversos países, os autores concluem que o uso da norma é útil no ensino de profissionais e para aplicação prática nas empresas.

Da mesma forma, Dingsøyr et al. (2000) apresentam uma abordagem de ensino baseada em estudos de caso reais da indústria, aplicada durante uma disciplina de Melhoria de Processos. Os resultados mostraram satisfação dos estudantes e melhoria nas notas.

Além desses estudos, existe uma forte tendência no uso de jogos e simulações para dinamizar o ensino de Engenharia de Software, tais como [Bessa et al. 2012; Souza and Resende 2010; Wangenheim et al. 2012, 2013]. Em [Wangenheim et al. 2009] os autores elencam diversos jogos e simulações utilizados no ensino de Engenharia de Software e a forma como são avaliados.

3. Estudo de Caso

O presente trabalho tem como principal objetivo o ensino dos conceitos relacionados à Melhoria de Processos de Software utilizando uma abordagem prática apoiada por uma norma. Para esse fim, é conduzido um Estudo de Caso Exploratório, seguindo a abordagem metodológica definida em [Runeson and Höst 2009]. Neste estudo, alunos de três turmas de disciplinas introdutórias de Engenharia de Software, do curso de

Sistemas de Informação da universidade Unisul[1] são envolvidos ativamente na avaliação de alinhamento dos processos de organizações de software segundo a norma ISO/IEC 29110. Seguindo a abordagem metodológica escolhida, nesta seção são apresentados: a definição, execução análise e interpretação e os resultados iniciais do Estudo de Caso.

3.1. Definição do Estudo de Caso

Dentre os diversos conceitos envolvidos na Melhoria de Processos, foi escolhida a aplicação de uma avaliação de processos seguindo um método de avaliação, pela viabilidade em uma disciplina de um semestre, aplicada na forma de um trabalho em equipes, planejado para a duração de 48 horas, entre atividades em sala (20 horas) e fora de sala (28 horas) – vide a Tabela 1.

A amostra dos alunos e das organizações participantes foi escolhida por conveniência, pois os autores ministraram as disciplinas e os alunos matriculados nas turmas foram envolvidos no trabalho como uma atividade avaliativa incluída no plano de ensino. Da mesma forma as organizações de software foram escolhidas pelos próprios alunos, pois em cada equipe, ao menos um dos membros trabalhava em uma organização de desenvolvimento de software, que, após solicitada pelo aluno a autorização da chefia imediata, foi envolvida de forma anônima no estudo.

Com base na motivação desta pesquisa, os objetivos de aprendizagem para o estudo de caso foram definidos como: (i) Aplicar a avaliação de processos de software em um ambiente real, e; (ii) Motivar para o uso de normas e modelos de referência na melhoria de processos de software. A partir desses objetivos, a pergunta de pesquisa para o estudo de caso foi definida: “Como a realização de avaliações de alinhamento de processos à norma ISO/IEC 29110 impacta na motivação dos alunos para a necessidade de adoção de normas nas MPOs?”.

Os dois objetivos gerais de aprendizagem são decompostos nos objetivos específicos das atividades presentes na Tabela 1 (vide coluna Objetivos). A avaliação do atendimento dos objetivos gerais de aprendizagem se dá por meio da coleta dos *feedbacks* fornecidos pelos alunos no próprio relatório entregue por cada equipe, no qual os alunos foram instruídos a incluir a sua impressão pessoal da experiência de aplicação da norma e sua importância na melhoria de processos de software.

No trabalho prático deste estudo de caso os alunos, divididos em grupos, avaliam o alinhamento dos processos de organizações de software aos processos da norma ISO/IEC 29110, elaborando um relatório na forma de um pequeno artigo (máximo 4 páginas) no qual os alunos apresentam os resultados e uma avaliação subjetiva do atingimento dos objetivos pedagógicos da experiência.

Plano de Aulas

As atividades realizadas neste estudo de caso envolvem aulas expositivas, atividades práticas realizadas em sala e fora de sala de aula, elaboração de um relatório e uma apresentação final. Os detalhes do plano das atividades são apresentados na Tabela 1.

[1] http://www.unisul.br/

Tabela 1: Planejamento das atividades

Hs	Tópico	Estratégia	Objetivos
02	Apresentação do tema, do trabalho e do fluxo geral	Em sala de aula. Aula expositiva	- Motivar para a realização do trabalho
04	Apresentação geral dos conceitos e técnicas de Melhoria de Processos e de avaliação de processos	Em sala de aula. Aula expositiva Utilização das ferramentas (planilha de avaliação)	- Compreender os conceitos básicos de Melhoria de Processos - Compreender e saber utilizar o processo de avaliação
04	Apresentação geral da norma ISO/IEC 29110	Em sala de aula. Aula expositiva	- Adquirir conhecimento básico da norma ISO/IEC 29110
02	Contextualização da organização	Em sala de aula. Alunos formam os grupos e procuram identificar e contextualizar a organização de software	- Definir o contexto da organização de software
02	Definição da entrevista	Em sala de aula. Alunos definem perguntas e preparam a planilha de coleta de dados para avaliação	- Preparar materiais necessários para a avaliação
16	Coleta de evidências e afirmações (avaliação inicial dos processos)	No local da organização de software. Alunos entrevistam os membros da organização, coletam evidências diretas, indiretas e afirmações	- Coletar evidências diretas e indiretas - Coletar afirmações - Iniciar preenchimento da planilha
04	Avaliação dos processos	Em sala de aula. Alunos realizam a avaliação e pontuação dos processos	- Avaliar e pontuar os processos
08	Elaboração do relatório	Em casa. Alunos elaboram o relatório final da avaliação na forma de um artigo de relato de experiência	- Elaborar relatório final - Avaliar o estudo de caso
04	Agrupamento e análise dos dados	Somente o professor realiza o agrupamento e análise dos dados recebidos dos alunos	- Consolidar as avaliações
02	Apresentação geral dos resultados das avaliações	Professor apresenta os resultados gerais das avaliações dos processos	- Conhecer os resultados globais das avaliações

3.2. Execução

O estudo de caso foi aplicado em três turmas da disciplina de Engenharia de Software da Unisul no segundo semestre de 2014 e no segundo semestre de 2015, envolvendo o total de 50 alunos. Logo na primeira aula do estudo de caso, os alunos foram agrupados em equipes de até 3 alunos (por afinidade), formando 20 equipes no total. As demais aulas e atividades seguiram o planejamento geral (Tabela 1). Os dados foram obtidos por meio de entrevistas e coleta de evidências diretas, indiretas e afirmações pelos alunos presencialmente nas empresas avaliadas. Após a consolidação dos dados (Figura 1) é possível perceber que os resultados esperados de Gerência de Projetos são melhor atendidos, enquanto os de Implementação de Software tiveram menor alinhamento.

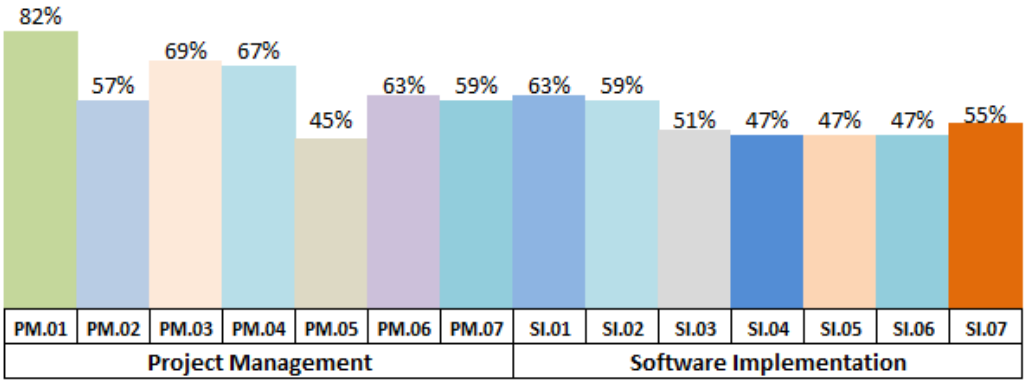


Figura 1: Resultados das Avaliações

3.3 Análise dos Resultados do Estudo de Caso

Das 20 equipes envolvidas, 03 (15%) não conseguiram realizar completamente a avaliação no prazo do trabalho, sendo que uma delas indicou dificuldades de acesso à empresa. Apesar disso, todas as demais equipes relataram como positiva a experiência de realizar uma avaliação de processos real utilizando uma norma internacional.

Em relação à pergunta de pesquisa, 16 das 17 equipes que concluíram o trabalho (~94%) relataram aumento da motivação para o uso de normas na melhoria de processo de software. Alguns relatos confirmam essa percepção:

- "O seguimento das normas ISO/IEC pode ser muito interessante e benéfico para as MPEs e também para grandes empresas [...]";
- "[...] É preciso mudar a mentalidade e cultura de desenvolvedores e empresas de software para aumentar não apenas a qualidade de software, mas a rentabilidade das empresas de software em geral";
- "A adoção das normas também pode trazer outros benefícios como: reconhecimento internacional, aumento da confiança e satisfação dos clientes".

Em somente uma equipe (~6%) foi percebido que os integrantes não se sentiram motivados ao uso de normas, considerando-as de pouca importância prática.

É importante ressaltar que o envolvimento direto dos pesquisadores com o objeto da pesquisa, pode representar uma ameaça à validade dos resultados observados [Runeson and Höst 2009]. Além disso, a pequena amostra (3 turmas, 50 alunos e 20 empresas) também pode representar limitações à validade dos resultados.

Os resultados das avaliações apresentados na seção 3.2 foram revisados pelos professores, entretanto, por terem sido coletados por alunos sem experiência anterior em avaliação de processos, podem conter erros de interpretação dos processos da norma ou das evidências coletadas.

4. Conclusão

Este trabalho apresenta um estudo de caso onde a norma ISO/IEC 29110 é utilizada como apoio ao ensino dos conceitos de Melhoria de Processos de Software no contexto da disciplina de Engenharia de Software. Os alunos, divididos em grupos, realizam avaliações de processos em organizações de software reais com base na norma.

Os dados coletados pelos alunos nas avaliações indicam que as organizações de software atendem, em média, a aproximadamente 58% dos resultados esperados da norma, com destaque positivo para os processos de planejamento de projetos (82%) e negativo para áreas técnicas como testes unitários (47%) ou análise de riscos (47%).

Uma das principais dificuldades enfrentadas se deu no acesso de algumas equipes às evidências diretas ou indiretas de processos gerenciais, que muitas vezes contém dados sigilosos e não acessíveis a toda a equipe. Mas, de forma geral, todos conseguiram contornar essa dificuldade e obtendo acesso ao menos às evidências necessárias para realizar a avaliação.

Quanto à pergunta de pesquisa, os relatos coletados nos relatórios das equipes indicam que todos avaliaram como positiva a experiência prática de avaliação de

processos e 94% dos membros das equipes consideram-se motivados a utilizar normas e modelos para a melhoria de processos de software.

Referências

Bessa, B., Cunha, M. Da and Furtado, F. (2012). ENGSOFT: Ferramenta para Simulação de Ambientes Reais para auxiliar o Aprendizado Baseado em Problemas (PBL) no Ensino de Engenharia de Software. *Imago.Ufpr.Br*,

Bosse, Y. and Gerosa, M. A. (2015). Reprovações e Trancamentos nas Disciplinas de Introdução à Programação da Universidade de São Paulo : Um Estudo Preliminar. *WEI - Workshop sobre Educação em Computação*, p. 1–10.

Dingsøyr, T., Jaccheri, M. L. and Wang, A. I. (2000). Teaching software process improvement through a case study. *Computer Applications in Engineering Education*, v. 8, n. 3-4, p. 229–234.

ISO/IEC (2012). ISO/IEC CD TR 29110-5-1 Software Engineering - Lifecycle Profiles for Very Small Entities (VSE) - Part 5-1-1: Management and Engineering Guide - Basic Profile. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=60389.

ISO/IEC/IEEE (2010). ISO/IEC/IEEE 24765 - Systems and software engineering -- Vocabulary. n. 1, p. 1–418.

LAPORTE, C. Y., ALEXANDRE, S., O'CONNOR, R. V. (2008) A software engineering lifecycle standard for very small enterprises. In: European Conference on Software Process Improvement. Springer Berlin Heidelberg, 2008. p. 129-141.

Laporte, C. Y. and Connor, R. V. O. (2014). An Innovative Approach in Developing Standard Professionals by Involving Software Engineering Students in Implementing and Improving International Standards. In *ICES 2014 – International Cooperation for Education about Standardization Conference*.

Runeson, P. and Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, v. 14, p. 131–164.

Souza, M. and Resende, R. (2010). SPARSE: Um Ambiente de Ensino e Aprendizado de Engenharia de Software Baseado em Jogos e Simulação. *Anais do Simpósio Brasileiro de Informática na Educação*, v. 1, n. 1.

Wangenheim, C. G. von, Savi, R. and Borgatto, A. F. (2012). DELIVER! - An educational game for teaching Earned Value Management in computing courses. *Information and Software Technology*, v. 54, n. 3, p. 286–298.

Wangenheim, C. G. von, Savi, R. and Borgatto, A. F. (2013). SCRUMIA - An educational game for teaching SCRUM in computing courses. *Journal of Systems and Software*, v. 86, n. 10, p. 2675–2687.

Wangenheim, C. G. von, Kochanski, D. and Savi, R. (2009). Revisão Sistemática sobre Avaliação de Jogos Voltados para Aprendizagem de Engenharia de Software no Brasil. *Fórum de Educação em Engenharia de Software*, p. 8.

Zelkowitz, M. V. (2009). An update to experimental models for validating computer technology. *Journal of Systems and Software*, v. 82, p. 373–376.

FRAMES: Um *Framework* para o Ensino-Aprendizagem dos Tópicos de Engenharia de Software dos Currículos de Referência da ACM/IEEE e SBC

Carlos S. Portela[1,3], 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]Instituto de Ciências Exatas e Naturais – Universidade Federal do Pará (UFPA)
Belém – PA – Brasil

[3]Área de Ciências Exatas e Tecnologia – Centro Universitário do Pará (CESUPA)
Belém – PA – Brasil

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

Abstract. *The ACM/IEEE curriculum guidelines defines a structure of topics and knowledge units to specify teaching guidelines and professional qualification for the Software Engineering (SE) area. However, it is impracticable to cover all content suggested by this curriculum. This paper presents a framework to support the teaching and learning of SE topics recommended by the ACM/IEEE and Brazilian Computer Society (SBC) curriculum guidelines. This framework, called FRAMES, was defined from the results of a survey about the topics relevance and from a case study about the teaching approaches effectiveness applied in SE discipline. Both the methods were conducted with teachers and students.*

Resumo. *O currículo de referência da ACM/IEEE define uma estrutura de tópicos e unidades de conhecimento a fim de especificar diretrizes de ensino e formação profissional para a área de Engenharia de Software (ES). No entanto, é simplesmente inviável cobrir todo o conteúdo sugerido por este currículo. Este artigo apresenta um framework de apoio ao ensino-aprendizagem dos tópicos de ES recomendados pelos currículos de referência da ACM/IEEE e SBC. Este framework, denominado FRAMES, foi definido a partir dos resultados de um survey e um estudo de caso, ambos realizados com professores e alunos, sobre a relevância dos tópicos ministrados e sobre a efetividade de abordagens de ensino aplicadas na disciplina de ES.*

1. Introdução

A indústria de software brasileira apresenta escassez de profissionais devidamente qualificados para atuarem em profissões que envolvem as etapas do processo de desenvolvimento de software, compreendidas pela Engenharia de Software (ES) [ABES 2015]. De acordo com Lethbridge *et al*. (2007), esta escassez de profissionais qualificados pode estar relacionada com uma educação inadequada. Especialmente nos cursos de graduação, os tópicos de ES são normalmente ensinados de forma bastante superficial [Wangenheim e Silva 2009].

Lethbridge *et al*. (2007) realizaram 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.

Neste cenário, considerou-se inviável cobrir todos os tópicos sugeridos pelos currículos de referência da ACM/IEEE (2013) e da SBC (2005), especialmente considerando a carga horária disponível para a disciplina de ES [Wangenheim e Silva 2009]. Além disso, observou-se que as abordagens de ensino tradicionais não adotam estratégias que alterem este atual cenário do meio acadêmico [Meira, 2015]. Por fim, identificaram-se diversas limitações quanto o desenvolvimento de competências profissionais dos estudantes no ambiente acadêmico [Wangenheim e Silva 2009]. Estas limitações acabam por agravar a carência de profissionais habilitados na área de ES [Lethbridge *et al*. 2007].

Segundo Meira (2015), é necessário fazer uma revisão do que e como se cria oportunidades para aprender ES. Sendo assim, os autores deste artigo realizaram um *survey* que buscou identificar os tópicos mais adotados (o que) dos currículos de referência da ACM/IEEE (2013) e da SBC (2005), a fim de refinar a quantidade de conteúdo e, consequentemente, utilizar de maneira mais efetiva o tempo disponível para a disciplina de Engenharia de Software [Portela, Vasconcelos e Oliveira 2015]. Adicionalmente, realizaram um estudo de caso sobre o uso de abordagens para o ensino de ES, a fim de identificar quais são mais efetivas para a aprendizagem dos alunos. Por fim, realizaram um levantamento de estratégias de capacitação da indústria, a fim de compreender como melhor desenvolver as competências profissionais nos estudantes.

A partir dos resultados destas pesquisas, definiu-se o *framework* apresentado neste artigo, denominado FRAMES. O termo *framework* adotado neste trabalho define uma estrutura conceitual básica de itens de ensino/aprendizagem aplicada no domínio de Engenharia de Software, sendo adaptável e reutilizável no contexto de disciplinas da graduação em Ciência da Computação, fornecendo apoio aos professores e alunos.

Além desta seção introdutória, a Seção 2 deste artigo apresenta os trabalhos relacionados a esta proposta de *framework*. Os métodos de pesquisa seguidos são descritos na Seção 3. Na Seção 4, os princípios, os módulos e as atividades do FRAMES são apresentados. A Seção 5 apresenta um teste do uso deste *framework* em uma disciplina de ES e análises sobre os resultados obtidos. Por fim, as limitações, as conclusões e as próximas etapas deste trabalho são relatadas na Seção 6.

2. Trabalhos Relacionados

Hazzan e Dubinsky (2006) definiram um *framework* para o ensino de Métodos de Desenvolvimento de Software (MDS). Este *framework* define princípios conceituais e especifica práticas referentes às atividades de ensino que orientam os professores no processo de desenvolvimento da disciplina de ES e na avaliação dos estudantes. Enquanto os dez princípios são ideias conceituais em que o *framework* se fundamenta, as cinco práticas abordam a implementação concreta do *framework*.

A primeira prática é “Curso” que apresenta uma estrutura detalhada do curso, a segunda é “Papel” que apresenta um esquema detalhado do perfil dos envolvidos. A terceira prática é “Medição” que apresenta medidas que ajudam a avaliar o processo de desenvolvimento. A quarta prática é “*Coaching*” que consiste em analisar os resultados de treinamentos realizados por professores, e a quinta é “Avaliação”, onde o professor deve avaliar o trabalho desenvolvido pelos alunos.

O *framework* proposto por Hazzan e Dubinsky (2006) é bem compreensível e detalha como implementá-lo em cursos de graduação. No entanto, seu foco é no ensino de MDS. O *framework* proposto nesta pesquisa se baseará na sua estrutura conceitual, definindo princípios e práticas. Como diferencial, poderá ser implementado no ensino de qualquer uma das unidades de conhecimento da ES.

Outro *framework* foi proposto na área de ES por Sowe, Stamelos e Deligiannis (2006), que usa a metodologia de desenvolvimento de software *open source* e foi implementado na disciplina de “Introdução à Engenharia de Software” no departamento de Informática da *Aristotle University* na Grécia.

Este *framework* possui 3 fases, onde na Fase 1 os estudantes realizam 8 horas de leitura sobre os tópicos de desenvolvimento, testes e integração. Então, os estudantes escolhem um projeto e fazem uma apresentação, detalhando o histórico do projeto, o procedimento de relatar *bugs* e as ferramentas de testes utilizadas. Na Fase 2, os estudantes aprendem a registrar os seus projetos em ferramentas de rastreamento de *bugs* e iniciam os testes em softwares livres e *open source* dos repositórios do *sourceforge.net*. Por fim, na Fase 3, baseado na apresentação dos projetos e nas atividades de teste, os estudantes são avaliados.

A principal contribuição desta proposta para o trabalho apresentado é a exposição dos estudantes a projetos reais de ES e a divisão do *framework* em fases bem distintas. No entanto, esta proposta foca apenas na área de testes. O *framework* proposto neste artigo busca ser estruturado de forma a ser aplicado a qualquer unidade de conhecimento da ES. Quanto à abordagem de projetos reais, buscar-se-á convidar profissionais da área para palestrar sobre sua área de atuação, além de usar clientes reais no projeto prático da disciplina. Adicionalmente, serão incorporadas práticas de capacitação da indústria durante a realização deste projeto prático.

3. Métodos de Pesquisa

O desenho de pesquisa adotado neste trabalho é baseado em uma abordagem multi-métodos [Hesse-Biber 2010], combinando um *survey* (Subseção 3.1), um estudo de caso (Subseção 3.2) e um levantamento (Subseção 3.3). Neste desenho de pesquisa, é usada a técnica de triangulação para consolidar os resultados obtidos nos diferentes métodos, considerando que questões de pesquisa relacionadas foram investigadas em cada um destes métodos. Desta forma, a triangulação reforça as conclusões e completude do estudo, trazendo maior credibilidade para os resultados da pesquisa.

3.1. *Survey* com Professores e Alunos

O *survey* realizado objetivou coletar as opiniões de alunos e professores em relação ao ensino/aprendizagem dos tópicos de ES [Portela, Vasconcelos e Oliveira 2015]. Os dados foram coletados durante o período de 16 de Março a 29 de Maio de 2015.

Receberam-se respostas de 70 participantes, sendo 23 professores e 47 alunos. Os participantes representam 12 estados do Brasil, sendo 50% 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 e 20% são de instituições privadas.

Foram utilizados como instrumentos para aplicação do *survey* questionários auto administrados, com questões de pesquisa que tinham respostas pré-definidas. Estas questões de pesquisa foram fortemente baseadas nos *surveys* aplicados por Wangenheim e Silva (2009) e Lethbridge *et al*. (2007). 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.

A Tabela 1 mostra as questões de pesquisa feitas para os professores da disciplina de Engenharia de Software a fim de identificar quais currículos de referência e quais tópicos estão sendo abordados. Para os alunos, perguntou-se o grau de aprendizagem (numa escala *Likert* de 0 à 5) para cada uma das 125 aprendizagens esperadas.

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

A partir da análise dos dados coletados, identificaram-se as 6 unidades mais adotadas pelos professores nas ementas da disciplina de ES. Em seguida, correlacionou-se o percentual de tópicos relevantes com o percentual de aprendizagem dos alunos, conforme apresenta a Figura 1.

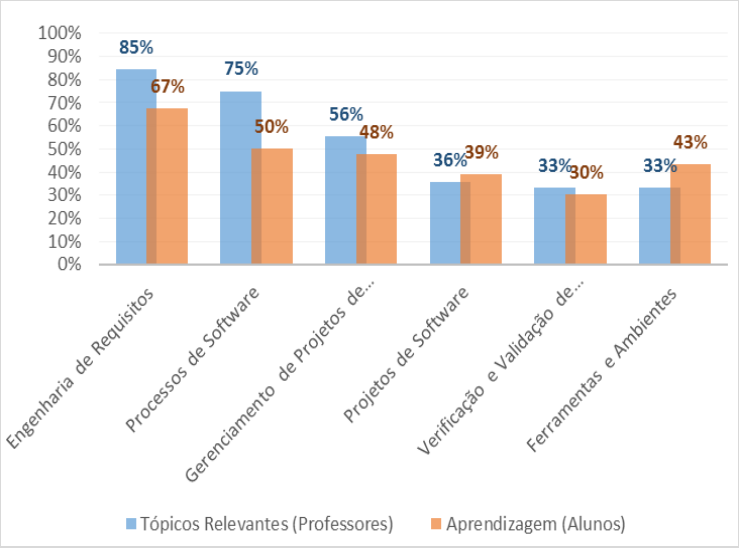


Figura 1 – Correlação entre Tópicos Mais Relevantes e Aprendizagem

Observa-se que Engenharia de Requisitos, de acordo com os resultados do *survey*, é a unidade que possui maior relevância, pois é amplamente contemplada nos currículos de Engenharia de Software, por 85% dos professores, e efetivamente aprendida por 67% dos alunos entrevistados. Em seguida, destacam-se as unidades de Processos de Software, ministrada por 75% dos professores e aprendida por 50% dos alunos, e Gerenciamento de Projetos de Software, ministrada por 56% dos professores e aprendida por 48% dos alunos entrevistados.

A fim de tratar a problemática da grande quantidade de tópicos a serem ministrados numa baixa carga horária da disciplina de ES, sugere-se priorizar os tópicos destas 6 unidades de conhecimento. As habilidades e competências a serem desenvolvidas foram selecionadas a partir das aprendizagens esperadas para cada um destes tópicos. Assim, estas unidades foram definidas como o foco de aplicação do FRAMES, integrando o seu módulo de Plano de Ensino. Para as demais unidades, a sugestão é que estas sejam abordadas em disciplinas complementares que envolvam ES.

3.2. Estudo de Caso sobre Abordagens de Ensino

De acordo com Prikladnicki *et al.* (2009), os métodos de ensino de ES podem estar focados no professor ou focados no aluno. Cada um destes métodos possui suas vantagens e desvantagens, em função do conteúdo a ser ministrado, das características pessoais dos alunos e da turma, entre outros fatores. No contexto desta pesquisa, realizou-se um estudo de caso na disciplina de Projeto Integrado em Engenharia de Software que consistiu na aplicação e análise de métodos de ensino, a fim de medir a sua efetividade no desenvolvimento de competências e na aprendizagem dos alunos.

Este estudo de caso foi conduzido no curso de Bacharelado em Ciência da Computação (BCC) do Centro Universitário do Estado do Pará (CESUPA), durante o segundo semestre de 2015. Assim, no primeiro dia de aula, a turma foi dividida em 2 equipes (A e B) de 7 alunos, onde cada uma seguiu um determinado conjunto de métodos de ensino: A – métodos focados no professor e B – métodos focados no aluno.

Foram utilizados como instrumentos para coleta e análise de dados questionários estruturados, dicionários de dados e mapas conceituais. Assim, ao final do estudo de caso, observou-se uma variação de 11% na aprendizagem da equipe B que seguiu métodos de ensino focados no aluno em relação à equipe A que seguiu uma abordagem focada no professor. A Tabela 2 apresenta os métodos de ensino considerados mais efetivos, de acordo com a análise de dados coletados durante este estudo de caso.

Tabela 2. Métodos de Ensino Focados no Aluno

Método	Descrição
Ensino Baseado no Desenvolvimento de Projetos Práticos de Software	Adequação de práticas de desenvolvimento de software para o contexto de uma disciplina de ES.
Ensino/Aprendizado Baseado em Dinâmicas	Apresentação dos conceitos básicos de ES, mesclando teoria e prática.
Aprendizagem Baseada em Problemas (PBL)	Uso de problemas para iniciar, direcionar e motivar o aprendizado.
Ensino/Aprendizagem Espiral	Combinação de aula tradicional, atividades em grupo, aprendizagem baseada em problemas, reflexão e prática.

Estes métodos foram incorporados, de maneira integrada, no modelo pedagógico do FRAMES. Desta forma, é possível apresentar, praticar e aplicar os tópicos das unidades de ES a fim de proporcionar uma aprendizagem mais contextualizada para os alunos. Adicionalmente, os dicionários de dados e os mapas conceituais se mostraram como efetivas ferramentas de avaliação de aprendizagem, sendo incorporados no módulo de Avaliação do *framework*. Os dicionários de dados permitem com que os alunos descrevam os conceitos aprendidos associados aos tópicos de ES e os mapas conceituais permitem com que representem como estes conceitos se relacionam entre si.

3.3. Levantamento de Estratégias de Capacitação

A fim de identificar como melhor desenvolver competências profissionais nos alunos, optou-se por realizar um levantamento com consultores em Melhoria do Processo de Software (MPS). Assim, consultaram-se 10 especialistas da área (consultores em MPS que também atuam como professores de graduação) sobre quais as estratégias que estes adotam para capacitar profissionais na indústria.

A partir da análise das respostas obtidas, onde identificaram-se as práticas de capacitação adotadas para cada uma das 6 unidades de conhecimento focos desta pesquisa, identificaram-se as práticas mais adotadas, conforme Figura 2.

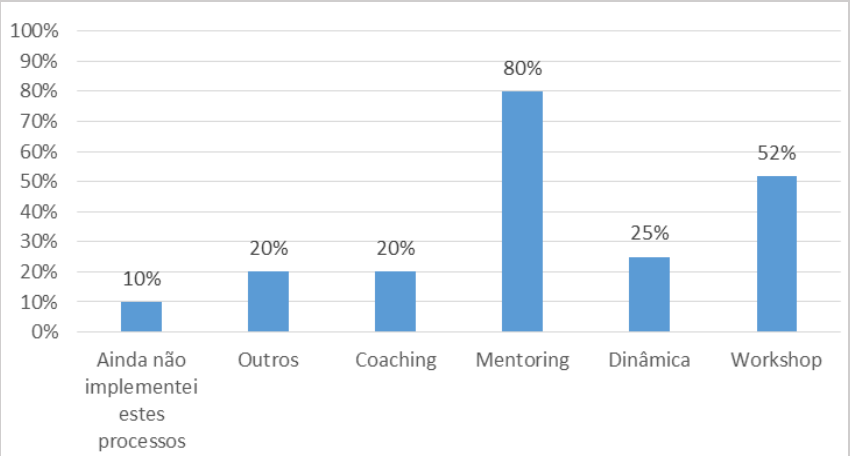


Figura 2 – Percentual de Adoção de Práticas de Capacitação

A prática de capacitação mais adotada foi *mentoring*, que consiste no consultor orientar e compartilhar com os profissionais da empresa-alvo da melhoria suas experiências e conhecimentos nos processos de MPS. A segunda prática mais adotada foi *workshop*, onde os consultores realizam um seminário de curta duração, apresentando técnicas e habilidades e demonstrando como estas podem ser aplicadas.

O uso de dinâmicas, onde um instrutor sugere uma atividade análoga a uma atividade técnica, e da prática de *coaching*, onde um profissional realiza um treinamento em técnicas específicas, ainda são poucos explorados por consultores.

A partir da identificação e da análise destas práticas, foram discutidas as estratégias de adequação destas para o ensino de ES do *framework*. Assim, estas foram incorporadas no módulo Modelo Pedagógico.

4. O *Framework* FRAMES

O FRAMES (acrônimo de *FRAMework* para o ensino de *E*ngenharia de *S*oftware) baseia-se em princípios (Subseção 4.1) que permitiram identificar e definir seus módulos (Subseção 4.2). Estes módulos, por sua vez, são compostos por itens de ensino/aprendizagem que podem ser instanciados, conforme descreve a Subseção 4.3.

4.1. Princípios do *Framework*

A proposta do FRAMES se baseia em 4 (quatro) princípios derivados de trabalhos relacionados que abordam o uso de métodos focados no aluno no ensino de ES:

I. O estudante deve ser o foco do processo de aprendizagem;

II. A aprendizagem deve ser baseada na resolução de problemas;

III. A realização de projetos práticos desenvolve competências e habilidades;

IV. O modelo pedagógico deve seguir uma abordagem iterativa.

De acordo com o Princípio I, o ensino é focado no estudante, onde o professor deve atua como um tutor, não expondo sua visão acerca do assunto, apenas guiando os aprendizes na busca de soluções. Desta forma, os estudantes tendem a adquirir conhecimento através do processo de autorreflexão e das relações com outros alunos.

O Princípio II preza pelo uso de problemas baseados no mundo real para estimular os alunos a desenvolverem o pensamento crítico, criarem habilidades para solução de problemas e adquirirem conhecimento sobre os principais conceitos da área em questão. Já o Princípio III recomenda o envolvimento dos estudantes em projetos práticos de desenvolvimento de software para que estes possam aplicar o conhecimento obtido de maneira satisfatória, ou seja, desenvolver suas competências e habilidades.

Por fim, o Princípio IV enfatiza que os estudantes tendem a aprender os tópicos de ES de forma mais eficaz através de abordagens iterativas, pois terão a oportunidade de realizar suas atividades em um ciclo, avaliar o seu trabalho ao final e, em seguida, aplicar o conhecimento adquirido em um próximo ciclo.

4.2. Módulos do Framework

A partir destes princípios, definiram-se os módulos do FRAMES. A necessidade dos itens de ensino/aprendizagem de cada um destes módulos foi levantada a partir da identificação dos principais conceitos envolvidos no ensino de ES.

Estruturalmente, o FRAMES possui 4 (quatro) módulos, conforme apresenta a Figura 3.

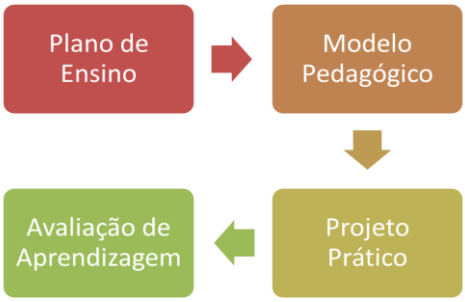


Figura 3. Módulos do *Framework* FRAMES

O módulo Plano de Ensino incorpora os tópicos das seguintes unidades de conhecimento: I) Engenharia de Requisitos; II) Processos de Software; III) Gerenciamento de Projetos de Software; IV) Projetos de Software; V) Verificação e Validação de Software; VI) Ferramentas e Ambientes. Estes tópicos foram identificados a partir do *survey* (Seção 3.1). Desta forma, pretende-se limitar a quantidade de tópicos ministrados, a fim de que estes possam ser adequar a carga horária disponível para a disciplina de ES.

O módulo Modelo Pedagógico, fundamentado nos conceitos de Behar (2009), foi definido a partir de métodos de ensino focados nos alunos, abordagens iterativas de ensino, práticas de capacitação da indústria e estratégias de avaliação. Este modelo pedagógico, representado na Figura 4, combina aprendizagem centrada em problema (PBL), atividades em grupo como discussões e dinâmicas, reflexão e projeto prático.

Figura 4. Modelo Pedagógico do FRAMES

Neste modelo pedagógico, inicialmente seleciona-se uma unidade de conhecimento que será objeto de estudo. O estudo de cada unidade de conhecimento começa a partir da identificação de um problema (baseado no Princípio II, a fim de fomentar as escolhas de práticas e técnicas), relacionado ao projeto a ser desenvolvido. Então, os alunos realizam uma Discussão com profissionais especialistas na unidade de conhecimento. Esta discussão pode se dar a partir da visita de um profissional à sala de aula, uma palestra em vídeo, videoconferência ou ainda a partir de uma visita técnica da turma a uma empresa de software.

Em seguida, os alunos colocam os conceitos de aula em Prática através da realização de dinâmicas ou *workshops* do uso de ferramentas. Esta etapa permite aos alunos internalizarem determinadas habilidades a partir de atividades práticas. Finalmente, os alunos integram as habilidades adquiridas a partir da realização de um projeto prático de desenvolvimento de software. Este projeto busca a Contextualização do uso dos tópicos de ES através da adoção de técnicas imersivas de ensino/aprendizagem, como o uso de clientes/problemas reais, definição de prazos (cronograma), definição de papéis e responsabilidades pelos alunos. Nesta etapa são realizadas as atividades de *mentoring* (papel assumido pelo professor) e *coaching* (papel assumido por alunos veteranos).

Paralelamente a todas estas etapas, deve ocorrer a Preparação dos alunos, que consiste na realização de leitura de artigos com relato de casos práticos da indústria ou

vídeo-aulas relacionadas às unidades a fim de exemplificar e criar uma compreensão de como estes tópicos são aplicados. Ao final de cada ciclo iterativo, os alunos realizam uma Reflexão sobre essa rápida experiência de aprendizagem. Este ciclo iterativo deve ser executado para cada unidade de conhecimento da ES ministrada na disciplina.

Já o módulo Projeto Prático sugere o uso de clientes/problemas reais, definição de prazos (cronograma), definição de papéis e responsabilidades pelos alunos. Para tal, disponibiliza um modelo de processo (*http://goo.gl/rKmctX*) que define atividades, papéis e artefatos de acordo com as 6 unidades de conhecimento do Plano de Ensino. Por fim, no módulo Avaliação de Aprendizagem, sugerem-se estratégias de avaliação como a entrega de produtos de trabalho (lista de requisitos, protótipos de tela, código-fonte, dentre outros), participação individual dos alunos, questionários, dicionários de dados e mapas conceituais.

4.3. Instanciação

O *framework* FRAMES estará disponível em uma plataforma de *wiki* (*http://frames.pbworks.com/*). Assim, sua instanciação deverá ocorrer em três etapas. Na *Etapa 1 - Configuração do Ambiente*, inicialmente o professor cria sua própria versão de *wiki*, instanciando-a para sua disciplina. Em seguida, o professor seleciona a unidade de conhecimento que deseja ministrar. Assim, deve instanciar os tópicos do módulo Plano de Ensino de acordo com a unidade de conhecimento selecionada.

Na *Etapa 2 - Aplicação do Framework*, o professor deverá fazer uso do módulo Modelo Pedagógico, onde terá acesso a uma base de vídeo-aulas, propostas de *workshops* e dinâmicas, bem como um formulário para registro das reflexões. No módulo Projeto Prático, os alunos terão acesso ao modelo de processo, para que possam instanciá-lo em seu projeto. Por fim, na *Etapa 3 - Aplicação do Framework*, o professor poderá instanciar as estratégias de avaliação do módulo Avaliação de Aprendizagem, onde terá acesso à *templates* e exemplos de itens de avaliação.

5. Teste de Uso do *Framework*

Antes de iniciar a disciplina, o FRAMES foi apresentado ao professor responsável, que possui mestrado e atua há mais de 10 anos na área de ES. O mesmo acessou a *wiki*, instanciou a sua versão e fez os ajustes necessários para aplicação dos métodos no plano de ensino da disciplina de Engenharia de Software do CESUPA.

Em seguida, no primeiro dia de aula, a turma foi dividida em 2 equipes (A e B), onde cada uma tinha 1 cliente/projeto específico e 1 aluno veterano para realizar *coaching*. O projeto da equipe A consistia no desenvolvimento de um aplicativo *mobile* para acessar o sistema de aluno do CESUPA, cujos clientes eram os próprios alunos da instituição. Já o projeto da equipe B consistiu no desenvolvimento de uma aplicação *web* para cadastrar ideias de projetos e profissionais interessados em integrarem a equipe destes projetos, cujo cliente era o professor da disciplina.

Inicialmente, os alunos preencheram um questionário sobre o conhecimento prévio (antes de iniciar a disciplina) dos tópicos de Processo de Software. Para cada tópico desta unidade, perguntou-se qual o grau de conhecimento do aluno numa escala *Likert* de 0 a 5: 0 - Não sei absolutamente nada; 1 - Sei vagamente; 2 - Sei o básico; 3 - Sei moderadamente; 4 - Sei muito; 5 - Sei em profundidade. Além disso, os alunos

criaram mapas conceituais e dicionários de dados das unidades de conhecimento da ES a partir do seu conhecimento prévio. As Figuras 5 e 6 apresentam, respectivamente, um exemplo de mapa conceitual e dicionário de dados da unidade de Processo de Software.

Figura 5 – Exemplo de Mapa Conceitual

Cascata: Modelo de processo clássico, que visa o desenvolvimento em sequência, passando por cada etapa individualmente.

Ciclo de Vida: Etapas de desenvolvimento de um sistema.

Fases de Desenvolvimento: Etapas a serem cumpridas no decorrer de um modelo de desenvolvimento.

Incremental: Modelo em que as etapas de um processo são integradas e desenvolvidas paralelamente.

Iterativo: Processo de software que visa a repetição de determinadas fases do projeto, visando a adequação dos requisitos.

Medição do Processo: Métrica utilizada para determinar a duração, viabilidade e adequação de um processo.

Métricas do Processo: Permitem ter ideia da eficácia de um processo existente.

Modelos de Processo: Conjunto de técnicas de desenvolvimento de um processo.

Figura 6 – Exemplo de Dicionário de Dados

Os tópicos da unidade de conhecimento Processo de Software foram ministrados durante 3 semanas, totalizando 9 aulas (9 horas ou 15% da carga horária da disciplina). Na aula 1, o estudo desta unidade iniciou-se a partir da identificação de um problema, como por exemplo "definir um processo para o desenvolvimento de uma aplicação *mobile* que acesse o portal do aluno" relacionado ao projeto a ser desenvolvido na disciplina. Então, na aula 2, os alunos realizaram leituras de artigos sobre definição de processos e, na aula 3, assistiram palestras sobre o *framework* ágil Scrum.

Em seguida, nas aulas 4 e 5, puderam vivenciar a aplicação prática destes conceitos através da dinâmica de fábrica de aviões [Prikladnicki *et al.* 2009] que consiste em produzir aviões de papel, onde o processo de produção é de decisão do time. Após a dinâmica, na aula 6, os alunos registraram as expectativas em relação à aplicação dos conceitos aprendidos no projeto prático da disciplina.

Por fim, os alunos integraram as habilidades adquiridas nestas etapas no projeto de software da disciplina de ES, nas aulas 7 e 8, modelando um processo, a partir das notações do padrão *Software Process Engineering Metamodel* (SPEM), a ser seguido no decorrer da disciplina. A sugestão para uso do padrão SPEM e o treinamento na ferramenta de modelagem foi uma das atividades de *coaching* dos alunos veteranos.

Após a realização desta atividade, os alunos realizaram uma nova reflexão, durante a aula 9, baseada nos aprendizados adquiridos no projeto prático. Ao final do ensino dos tópicos desta unidade de conhecimento, os alunos responderam os

questionários sobre o conhecimento adquirido e desenvolveram novos dicionários de dados e mapas conceituais.

6. Considerações Finais

6.1. Limitações

A proposta de adotar um *framework* para o ensino permite que o professor possa lecionar uma determinada unidade de conhecimento da ES, a fim de melhor utilizar a carga horária da disciplina. No entanto, tempo continua sendo um fator crítico. A partir do teste de aplicação (Seção 5), observou-se que se leva, em média, 3 aulas para lecionar os tópicos de uma unidade. Neste sentido, ajustes estão sendo feitos no Plano de Ensino e no Modelo Pedagógico a fim de reduzir este tempo para 2 aulas.

Outra limitação do FRAMES consiste na falta de experiência de professores que não atuaram na indústria que podem vim a fazer uso deste *framework*. Para esta limitação, sugere-se, quando possível, a participação de um profissional da indústria para apoiar, principalmente na etapa de Discussão (Figura 4). Adicionalmente, pretende-se criar vídeos e tutorias a fim de auxiliar a capacitação dos professores no *framework*.

Apesar destas limitações, espera-se que a disponibilização do FRAMES em uma *wiki* permita aos professores instanciarem métodos que melhor desenvolvam competências profissionais nos alunos. Além disto, estes poderão contribuir com a sua evolução, através do envio de vídeos, artigos, além de sugestões de melhoria. Os alunos poderão usar esta *wiki* para registrarem seu aprendizado em cada uma das unidades de conhecimento, instanciarem o modelo de processo no projeto prático e estudarem fora do ambiente de sala de aula, através dos vídeos e outras ferramentas disponíveis.

6.2. Conclusões

A possibilidade de instanciar o FRAMES permitirá ao professor escolher a unidade de conhecimento que deseja lecionar, a partir do uso de abordagens de ensino focadas no aluno, como PBL, projeto prático e processo iterativo. Além disso, permitirá ao professor adotar uma estratégia de avaliação que foca na análise da aprendizagem dos alunos e na execução de projetos práticos.

No entanto, destaca-se que o *framework* baseia seu plano de ensino em tópicos e aprendizagens que mudam constantemente, devido à própria dinâmica da área de Computação. A fim de tratar esta questão, será possível que os professores instanciem sua própria versão do FRAMES, atualizando os tópicos do Plano de Ensino.

Além disso, o *framework* permitirá aos alunos cadastrarem em uma *wiki* os conceitos adquiridos, em relação aos tópicos de ES, através de um dicionário de dados e suas reflexões sobre a aplicação destes no projeto prático da disciplina de ES. Assim, o professor poderá utilizar a própria *wiki* para disseminar a socialização do conhecimento produzido durante a disciplina.

6.3. Próximas Etapas

A etapa atual desta pesquisa consiste na finalização da alimentação da base de dados da *wiki* do *framework*. Paralelamente, está sendo definido o protocolo do estudo de caso

para aplicação deste *framework* em 3 disciplinas da área de ES de instituições de ensino diferentes. Sendo assim, a próxima etapa da pesquisa consiste na validação do uso do FRAMES para o ensino de todas as 6 unidades de conhecimento contempladas por este.

Agradecimentos

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

Referências

ABES (2015). Mercado Brasileiro de Software: panorama e tendências. 1ª. ed. São Paulo: Associação Brasileira das Empresas de Software.

ACM/IEEE (2013) "Computer Science Curricula 2013 – Curriculum Guidelines for Undergraduate Degree Programs in Computer Science", https://www.acm.org, June.

Behar, P. (2009) "Modelos Pedagógicos em Educação a Distância", Em: ArtMed, Porto Alegre.

Hazzan, O. e Dubinsky, Y. (2006). A Framework for Teaching Software Development Methods. *Proceedings of the 28th International Conference on Software Engineering*, pages 703-706. Shanghai, China.

Hesse-Biber, S. (2010). Mixed Methods Research - Mixing Theory and Practice. The Guilford Press.

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

Meira, S. Sistemas de Informação e Engenharia de Software – Cadê as Escolas? Revista da SBC Engenharia de Software - Qual é o impacto da ES no mercado de Computação e na sociedade como um todo? 1ª. ed. Cap. 1, páginas 11-15. Porto Alegre, Brasil.

Portela, C., Vasconcelos, A. e Oliveira, S. (2015). Análise da Relevância dos Tópicos e da Efetividade das Abordagens para o Ensino de Engenharia de Software: Resultados de um Survey com Professores e Alunos. Em *Anais do VIII Fórum de Educação em Engenharia de Software*, páginas 24-35. Belo Horizonte, Brasil.

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, Junho.

Sowe, S., Stamelos, I. e Deligiannis, I. (2006). A Framework for Teaching Software Testing using F/OSS Methodology. *Open Source Systems.* Chap. 6, pages 261-266.

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.

Jogos Educativos no Ensino da Engenharia de Requisitos

Daniel Negreiros Araujo[1], Maria Lencastre P. de M. Cruz[1], João Henrique Pimentel[1,2], Mariana Duque[1], Fernanda Alencar[1,3]

[1]Universidade de Pernambuco (UPE) – Recife – PE – Brasil
[2]Universidade Federal Rural de Pernambuco (UFRPE) – Recife – PE – Brasil
[3]Universidade Federal de Pernambuco (UFPE) – Recife – PE – Brasil
{dna, mlpm, jhcp, fernandaalenc}@ecomp.poli.br

Resumo. *Este artigo apresenta um conjunto de jogos voltados para o ensino de Engenharia de Requisitos (ER) publicamente disponíveis, avaliando se estes satisfazem um conjunto de requisitos relevantes para softwares educacionais. Os requisitos adotados nesta avaliação são provenientes do EduCatalog4RE, um catálogo de requisitos para o desenvolvimento de softwares educacionais. Dos seis jogos analisados, três mostraram uma boa cobertura dos requisitos, sugerindo que são bons complementos para o ensino de ER. Porém, de um modo geral, os resultados mostraram que os jogos voltados para o ensino de ER podem ser ainda melhor desenvolvidos, contemplando características pedagógicas de ensino, assim como tratando questão de acessibilidade.*

Abstract. *This paper presents a set of games publicly available and an analysis of them evaluating whether these include requirements regarded as required or desirable according to a catalog of requirements, called EduCatalog4RE, turned to assist an educational software development. Of the six games analyzed, three are considered to be good complements to RE teaching. However, in general, the results demonstrated that games that focus on teaching can be developed even better, considering teaching and learning features already consolidated by pedagogy, so as to address the issue of accessibility.*

1. Introdução

Classicamente, é sabido que a Engenharia de Requisitos (ER) tem como objetivo descobrir as reais necessidades dos *stakeholders* que direta ou indiretamente influenciam na determinação dos requisitos do sistema pretendido. Precisa-se entender o problema e seu contexto, elicitar os requisitos necessários, analisá-los, documentá-los e validá-los [Sommerville 2011]. Todavia deficiências na especificação de requisitos são consideradas uma das causas comuns de falhas de projetos na indústria de *software* [Figueiredo et al. 2006], apesar dos inúmeros esforços desenvolvidos e propostos pela comunidade na área de requisitos. As pesquisas continuam a apontar que 85% dos problemas de *software* têm origem na atividade de elicitação de requisitos [Fernandes et al., 2009]. Analisando-se essas afirmações surgem algumas suposições para esse fato: continua-se a não se dedicar tempo à especificação de requisitos; ou, a forma de ensino de ER não está sendo eficaz para os educandos. Acreditamos que há falhas no processo do ensino-aprendizagem de ER, pois, hoje, a forma tradicional de ensino por meio de aulas expositivas e simples exercícios mecânicos, não está sendo efetiva em seu objetivo metodológico. Já, segundo Perrenoud (2000), é necessário o professor adquirir

competências para criar situações desafiadoras, capazes de utilizar recursos didáticos variados que envolvam inclusive *softwares* educacionais. Isso se deve quer por mudança comportamental do educando quer por não ser de fato uma metodologia motivadora diante das inovações tecnológicas e mudanças nas tendências pedagógicas. Existe um amadurecimento na ideia de que os novos dispositivos e mídias eletrônicas podem influenciar no processo educacional de forma positiva. Há um crescente interesse no desenvolvimento de *software* educativo [Peixoto and Silva 2015].

O *software* educativo tem por finalidade o ensino ou autoaprendizagem sendo muito útil e proveitoso no processo de ensino-aprendizagem [Soffa e Alcântara 2008]. Deve-se disponibilizar um ambiente interativo, proporcionando ao aprendiz a oportunidade dele mesmo construir e refinar seu próprio conhecimento, a partir da investigação, do levantamento e do teste de hipóteses [Moura et al. 2013].

Em associação, os *softwares* educativos passam a ser incrementados com a estratégia da gamificação, tema que vem recebendo cada vez mais atenção de pesquisadores e profissionais. Essa estratégia inspira-se na experiência de usuário de jogos e se utiliza de elementos de design de jogos em contextos de não-jogos, como o caso do ensino. Nesse sentido, cria-se um ambiente em que o aprendiz se envolve em desafios abstratos, definidos por regras, interatividade e *feedback* que resulte em produzir uma reação emocional. É a ideia de pensar sobre uma experiência cotidiana, como por exemplo, correr, e converte isso em uma atividade que tenha elementos de concorrência, cooperação, exploração e até enredo em histórias [Duque et al. 2016].

Nesse trabalho buscou-se identificar e avaliar jogos desenvolvidos especificamente para o ensino de Engenharia de Requisitos. Como procedimento metodológico para o desenvolvimento da pesquisa, foram definidas quatro etapas: identificar os jogos existentes na área; baixar e classificar os jogos como pertinentes ou não à área de ER; definir o critério de avaliação dos jogos em termos de ensino; e avaliar os jogos de acordo com os critérios estabelecidos. O critério base para a avaliação dos *softwares* foi o uso do EduCatalog4RE [Henrique, 2015] [Henrique, 2016]. Trata-se de um catálogo de requisitos construído com o objetivo de auxiliar no desenvolvimento de *softwares* educacionais. Como resultado da aplicação desses mecanismos de pesquisa, chegou-se à análise de seis jogos existentes na literatura. Nessa pesquisa, não foram identificados trabalhos relacionados cujo objetivo tenha sido o de avaliar *softwares* educativos gamificados na área de Engenharia de Requisitos.

Este artigo se estrutura da seguinte forma: na Seção 2 apresenta-se o EduCatalog4RE; na Seção 3 descreve-se os procedimentos metodológicos utilizados; na Seção 4, apresenta-se, resumidamente, os jogos avaliados; na Seção 5 são discutidos os resultados da avaliação; e, por fim, na Seção 6 apresenta-se as conclusões e trabalhos futuros.

2. Fundamentação: EduCatalog4RE

O EduCatalog4RE [Henrique, 2016] é um catálogo de requisitos para *softwares* educativos. Ele foi desenvolvido a partir de uma revisão sistemática da literatura que contemplou diversas categorias de *softwares* educativos (Tutoriais, Exercício e prática, Aplicativos, Multimídia e *Internet*, Programação, Simulação e Modelagem, e Jogos).

O EduCatalog4RE possui duas versões. A primeira versão foi avaliada a partir de um questionário avaliativo com diversos docentes do país, assim como com representantes de empresas que produzem *softwares* educativos. A segunda versão foi criada com base no *feedback* dos vinte e nove participantes. Houve uma redução de cento e doze requisitos na primeira versão, para cento e oito na segunda versão, além de

melhorias em relação à usabilidade do catálogo. Para essa segunda versão foi criado um protótipo de uma ferramenta visando auxiliar na organização e busca dos requisitos; nela o usuário pode selecionar sugestões de uso do catálogo, assim como visualizar os requisitos por categoria. O usuário também pode optar por ver os tipos de *softwares* educativos abordados pela ferramenta.

Os requisitos do catálogo estão divididos em requisitos pedagógicos, requisitos de usabilidade e requisitos gerais. Os requisitos pedagógicos são ainda subdivididos de acordo com teorias de aprendizagem.

Considerando que esse catálogo contempla diversos tipos *softwares,* não consideramos todos os seus requisitos. Esta avaliação considera apenas os requisitos do catálogo que se aplicam a todos os tipos de *software* educativo, além daqueles que são voltados especificamente para jogos. Desses, nem todos os requisitos presentes no catálogo são aqui avaliados, pois alguns são estéticos ou opcionais, potencialmente adicionando ruído à avaliação [Araujo, 2016]. Dos 108 requisitos do EduCatalog4RE, foram selecionados 48, divididos em 30 requisitos obrigatórios e 18 requisitos desejáveis. Esses requisitos estão listados nas Tabelas 1 e 2.

Tabela 1. Requisitos obrigatórios avaliados

Requisitos Obrigatórios	Tipo de Requisito
R1- O sistema deve exibir *feedback* construtivo.	Pedagógico
R2- O sistema deve possuir vários desafios.	Pedagógico
R3- O sistema deve propor reflexões críticas ao trabalhar os conteúdos.	Pedagógico
R4- O sistema deve permitir que o usuário realize escolhas durante a interação com o ambiente.	Pedagógico
R5- O sistema deve propor ao usuário construir o máximo de soluções possíveis para cada problema proposto.	Pedagógico
R6- O conteúdo proposto deve se relacionar com os conhecimentos prévios do aluno.	Pedagógico
R7- É preciso criar um ambiente no qual os símbolos e signos do cotidiano dos alunos sejam transcritos visualmente.	Pedagógico
R8- Os objetos do ambiente devem mudar de estado quando o usuário interagir com eles, e devem ser manipuláveis, desta forma o usuário terá uma percepção maior durante a interação e terá uma maior interação no geral.	Pedagógico
R9- O sistema deve perceber as possíveis dificuldades do usuário e oferecer ajuda, através das interações do usuário com outros personagens.	Pedagógico
R10- Os personagens devem interagir entre si e com objetos distribuídos pelo ambiente.	Pedagógico
R11- O sistema deve permitir que o usuário possa interagir com o ambiente de várias formas.	Pedagógico
R12 -Os conteúdos devem ser planejados de forma hierárquica e sequencial.	Pedagógico
R13- O *software* deve penalizar o usuário, que serão consequências das suas ações.	Pedagógico
R14- O sistema deve prover *feedback* imediato para o usuário de acordo com as suas ações.	Pedagógico
R15- O sistema deve beneficiar o usuário caso ele acerte os desafios ou pegue itens importantes no cenário.	Pedagógico
R16- Os conteúdos devem ser divididos em vários níveis de conhecimento, iniciando sempre do menor nível de complexidade.	Pedagógico
R17- O sistema deve ser acessível ao público para o qual foi projetado.	Usabilidade
R18- O sistema deve disponibilizar ajuda através de menu, formulário, ou através de notificações para o usuário.	Usabilidade
R19- A interface do sistema deve ser fácil e bem intuitiva.	Usabilidade
R20- O sistema deve exibir instruções de uso sempre que solicitado pelo usuário.	Usabilidade
R21- O sistema deve disponibilizar um manual/tutorial que possua informações básicas sobre o sistema. Os usuários devem ter acesso ao material.	Usabilidade
R22- As instruções para uso do sistema devem estar visíveis e facilmente encontradas quando necessárias.	Usabilidade
R23- O sistema deve tornar a navegação fácil e eficiente.	Usabilidade
R24- O sistema deve exibir regras sempre que solicitado pelo usuário.	Usabilidade
R25- O sistema deve disponibilizar um *avatar* para facilitar a interação do usuário com o ambiente.	Requisito Geral
R26- O *software* deve possuir várias fases.	Requisito Geral
R27- O ambiente deve ser motivador e atraente, oferecendo ludicidade em sua interface e desafios.	Requisito Geral
R28- O sistema deve possuir vários níveis de dificuldade.	Requisito Geral
R29- O sistema deve exibir a pontuação dos usuários de forma individual e/ou coletiva.	Requisito Geral
R30- O sistema deve exibir o tempo dos desafios e das partidas, se houver.	Requisito Geral

Tabela 2. Requisitos desejáveis avaliados

Requisitos Desejáveis	Tipo de Requisito
D1- Disponibilizar o trabalho em rede (*Internet*)	Pedagógico
D2- O sistema deve permitir que o usuário crie desafios para outros jogadores.	Pedagógico
D3- O sistema deve prover *feedback* sonoro para auxílio aos usuários.	Pedagógico
D4- Organizar e disponibilizar os conteúdos (desafios e aulas) em ordem cronológica.	Pedagógico
D5- O sistema deve propor um ambiente que simule situações reais onde o problema será inserido.	Pedagógico
D6- O sistema deve ser capaz de detectar o perfil do usuário, isso deve acontecer antes dele começar a interagir com o sistema.	Pedagógico
D7- O sistema deve se adaptar as características do usuário.	Pedagógico
D8- O sistema deve levar em consideração a personalidade e/ou sentimentos do usuário, para propor estilos de aprendizagem.	Pedagógico
D9- Prover recursos que permitam que diferentes canais de processamento do usuário venham ser estimulados durante a aprendizagem.	Pedagógico
D10- O sistema deve apresentar de forma adequada os menus que proporcionarão a interação do usuário com o sistema.	Usabilidade
D11- O sistema deve disponibilizar um espaço para exibir os nomes dos desenvolvedores.	Requisito Geral
D12- O personagem (*Player*) e o ambiente devem possuir vários itens e características.	Requisito Geral
D13- O sistema deve possuir vários níveis de interação.	Requisito Geral
D14- O usuário poderá personalizar o *avatar* através da customização de novos elementos.	Requisito Geral
D15- O sistema deve calcular a pontuação dos usuários e gerar um ranking.	Requisito Geral
D16- O sistema deve gerar relatórios com dados estatísticos sobre o desempenho dos alunos.	Requisito Geral
D17- O sistema deve possuir uma versão "*desktop* ou *off-line* no caso do *m-learning*", para os mais diversos sistemas operacionais (com ênfase no *Windows*)	Requisito Geral
D18- O sistema deve propor problemas menores com um curto prazo para resolução.	Requisito Geral

3. Metodologia

Inicialmente foi realizado um levantamento, através de uma revisão bibliográfica, dos principais trabalhos sobre jogos para Engenharia de Requisitos. Essa revisão consistiu no levantamento e no estudo de trabalhos sobre jogos educativos, mais especificamente sobre os que focam em Engenharia de Requisitos.

Na pesquisa de artigos e trabalhos acadêmicos, foram realizadas buscas simples nas bases do Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE) e da *Springer,* não se realizando, de fato uma completa Revisão Sistemática da Literatura (RSL) [Kitchenham 2004]. No entanto, algumas práticas de RSL foram utilizadas, como por exemplo, definição da pergunta de pesquisa; palavras-chaves para a busca às bases digitais; critérios de inclusão e exclusão; *snowballing*; e inclusão manual de artigos com base na consulta a especialistas. Assim, o propósito da busca era o de encontrar os sites de busca digitais *softwares* educacionais gamificados que estivessem sendo utilizados para o ensino de Engenharia de Requisitos. Nesse contexto a única pergunta de pesquisa (QP) a ser respondida foi:

QP. Quais os jogos utilizados para o ensino de Engenharia de Requisitos?

Para responder a essa QP, foram definidas algumas palavras-chave (Tabela 3) e montadas de forma a obter os resultados pretendidos.

Tabela 3. Sequência de Busca

Sequência de Palavras-chave
(((*requirements OR software*) *AND engineering*) *OR* ((engenharia *AND* (requisitos *OR software*))) *AND* (*game OR games OR* jogo) *AND* (*teaching OR learning*).

Além da busca automática às bases, como dito anteriormente, foram obtidos artigos a partir da indicação de especialistas e também através da técnica *snowballing* – ou seja, onde se busca analisar os artigos citados.

Nem todos os jogos encontrados a partir dessa revisão da literatura estavam disponíveis publicamente, já que um dos critérios para a exclusão do trabalho no processo de análise seria a sua disponibilidade. Para ajudar no processo, buscou-se contato com os autores dos jogos. No total, conseguimos acesso a 11 jogos. Desse conjunto de 11 jogos encontrados, apenas 6 foram selecionados para análise, sendo excluídos aqueles que observamos não contemplar tópicos de ER e jogos não funcionais.

Inicialmente, cada jogo foi executado, para observar a área contemplada pelo mesmo. Em seguida, o jogo foi novamente utilizado, em conjunto com o catálogo de requisitos EduCatalog4RE, que lista requisitos importantes que devem estar presentes em ferramentas educativas, para se fazer uma avaliação de cada jogo em termos de: requisitos pedagógicos, requisitos de usabilidade e requisitos gerais.

4. Descrição e Avaliação dos Jogos

Os jogos foram primeiramente analisados, para fins de classificação, de acordo com as seguintes características:

1. Quantidade de jogadores: se o jogo é jogado de forma individual ou com vários jogadores.
2. Plataforma utilizada: *desktop*, *mobile*, *web*, tabuleiro.
3. Área de conhecimento: etapas da engenharia de requisitos, boas práticas de engenharia de requisitos, priorização de requisitos.

4.1 Descrição dos jogos

Na Tabela 4, tem-se uma visão geral dos seis jogos educativos que foram identificados e analisados. Todos os jogos executados em computador são individuais, enquanto os de tabuleiro são multi-jogador. Os jogos "Ilha dos Requisitos" e "RE-O-Poly" apresentam uma maior cobertura de conteúdo, enquanto os demais focam em etapas específicas da ER – a saber: priorização, elicitação, e análise de requisitos.

Tabela 4. Visão geral dos jogos avaliados

Jogo	Quantidade de Jogadores	Ambiente de execução	Área de conhecimento	Atividade de aprendizagem
Ilha dos Requisitos	Individual	Computador com conexão a Internet	Engenharia de Requisitos	O jogador se torna um sobrevivente de queda de avião em uma ilha, e deve resolver desafios referentes a Engenharia de Requisitos para conseguir escapar dela.
EAReq-Game	Individual	Computador em ambiente local	Priorização de requisitos	O jogador assume um papel de analista, e deve coletar requisitos e especificá-los de acordo com sua importância no projeto.
Software Quantum	Individual	Computador com conexão a Internet	Elicitação de requisitos	O jogador realiza interações entre os desejos do cliente com analista, designer e programador, estimulando simulações de comunicação oral entre eles, e documentação dos requisitos.
UbiRE	Individual	Computador em ambiente local	Análise de requisitos	O jogador realiza conexões entre determinados objetos em cômodos da casa, respeitando os pedidos feitos pelo morador, que podem ser ou não detalhados.
i* Game	2-4 jogadores	Jogo de Tabuleiro	Análise de requisitos	Os jogadores realizam atividades de acordo com as cartas e regras fornecidas, de modo a melhorar os modelos disponibilizados de i*.
RE-O-Poly	2-8 jogadores	Jogo de Tabuleiro	Boas práticas de Engenharia de Requisitos	Os jogadores são gerentes de projeto, e devem coletar SSP respondendo perguntas sobre boas práticas de ER e sobre os projetos que o jogo possui.

Nas subseções a seguir, cada jogo é brevemente descrito.

Ilha dos Requisitos

- Jogo *single-player*, acessível pela *web*, que foca nas etapas e processos da engenharia de requisitos. O jogo fornece sete desafios acessíveis sobre áreas relacionadas a ER, separadas por tópicos. Ele "envolve a concepção da sua estória e de seus personagens, assim como a especificação de regras, desafios, e a definição das maneiras como o jogo irá fornecer o feedback ao jogador." [Thiry et al., 2010]

EAReq-Game

- Jogo *single-player*, acessável pelo *desktop*, que possui como alvo a exercitação de priorização de requisitos. Ele simula uma situação ao jogador, "no qual atuará como um engenheiro de requisitos coletando, organizando e priorizando os requisitos em um cenário simulado, buscando progredir nas fases do jogo". [Chiavegatti e Petri, 2014]

Software Quantum

- Jogo *single-player* desenvolvido para plataforma *web*, que procura enfatizar a importância da engenharia de requisitos, *Software Quantum* simula uma interação entre cliente, analista, designer e programador. O cliente dá os requisitos e o código final do programa tem que ser equivalente aos requisitos pedidos, para o jogador atingir a pontuação máxima. [Knauss et al., 2008]

UbiRE

- O UbiRE, jogo *single-player* que foi desenvolvido para *desktop*, tem como foco a área de análise de requisitos. O jogador realiza conexões entre determinados objetos em cômodos da casa, respeitando os pedidos feitos pelo morador, que podem ser ou não detalhados, dependendo do nível de dificuldade escolhido. [Lima et al., 2012]

iStar Game

- Jogo *multiplayer*, iStar Game (*i** Game) é um jogo de tabuleiro não digital focado na área de análise de requisitos que utiliza modelos em i* como tabuleiro, cujo objetivo é melhorar estes modelos. Os jogadores realizam atividades de acordo com as cartas e regras fornecidas, de modo a melhorar os modelos disponibilizados de *i**. [Pimentel et al., 2012]

RE-O-Poly

- RE-O-Poly é um jogo *multiplayer* de tabuleiro não digital baseado em *Monopoly*, trocando o dinheiro pelo SSP (*Stakeholder Satisfaction Points*) e adaptando o tabuleiro e as cartas para ensinar boas práticas de engenharia de requisitos. Os jogadores são gerentes de projeto, e devem coletar SSP respondendo perguntas sobre boas práticas de ER e sobre os projetos que o jogo possui. [Smith e Orlena, 2008]

4.2 Avaliação dos jogos

A avaliação realizada baseou-se considerando os requisitos do EduCatalog4RE [Henrique, 2016]. Os requisitos foram divididos em obrigatórios e desejáveis, seguindo critérios de relevância, de conteúdo e técnicos. Alguns requisitos foram unificados ou removidos por serem ambíguos. São trinta requisitos obrigatórios avaliados e dezoito opcionais. Destes trinta obrigatórios, dezesseis são pedagógicos, oito são de usabilidade

e seis são requisitos gerais. Já dos dezoito opcionais, nove são pedagógicos, um de usabilidade, e oito requisitos gerais. Enfatiza-se que os requisitos são válidos para *Softwares*, em geral; logo, em algumas exceções eles não se aplicam aos jogos de tabuleiros analisados. A Tabela 5 traz a porcentagem de requisitos atendida por cada jogo.

Tabela 5. Porcentagem aproximada de requisitos atendidos

Jogo x Requisito	Requisitos Obrigatórios				Requisitos Desejáveis			
	Atendidos	Não Atendidos	Não se Aplicam	% atendida (aprox.)	Atendidos	Não Atendidos	Não se Aplicam	% atendida (aprox.)
Ilha dos Requisitos	24	6	0	80%	6	12	0	33%
EAReq-Game	9	20	1	31%	4	14	0	22%
Software Quantum	14	15	1	48%	5	13	0	28%
UbiRE	18	11	1	62%	7	11	0	39%
i* Game	22	2	6	92%	7	8	3	47%
RE-O-Poly	24	1	5	96%	8	7	3	53%

Já a Tabela 6 demonstra os resultados de acordo com os tipos de requisitos (pedagógicos, de usabilidade e gerais).

Tabela 6. Porcentagem de requisitos atendidos por tipo de requisito

Jogo x Requisito	Requisitos Obrigatórios			Requisitos Desejáveis		
	% Pedagógicos (aprox.)	% de Usabilidade (aprox.)	% Gerais (aprox.)	% Pedagógicos (aprox.)	% Usabilidade (aprox.)	% Gerais (aprox.)
Ilha dos Requisitos	75%	88%	83%	44%	0%	25%
EAReq-Game	25%	25%	60%*	11%	100%	25%
Software Quantum	44%	50%	60%*	22%	100%	25%
UbiRE	56%	63%	80%*	22%	100%	50%
i* Game	93%*	75%*	100%*	50%*	N/A	43%
RE-O-Poly	100%*	75%*	100%*	50%*	N/A	57%

Legenda: N/A - Não se aplica

* - Excluindo os requisitos que não se aplicam

A listagem de quais requisitos são satisfeitos por cada jogo está apresentada nas Tabelas 7 e 8.

Tabela 7. Satisfação dos Requisitos Obrigatórios

Jogo x Requisito	Ilha dos Requisitos	EAReq-Game	*Software Quantum*	UbiRE	i* Game	RE-O-Poly
R1	Sim	Não	Não	Sim	Sim	Sim
R2	Sim	Não	Sim	Sim	Sim	Sim
R3	Sim	Não	Não	Não	Sim	Sim
R4	Sim	Sim	Sim	Sim	Sim	Sim
R5	Não	Não	Sim	Não	Sim	Sim
R6	Sim	Sim	Não	Sim	Sim	Sim
R7	Sim	Sim	Sim	Sim	Sim	Sim
R8	Sim	Não	Sim	Sim	Sim	Sim
R9	Não	Não	Não	Não	N/A	N/A
R10	Não	Sim	Não	Não	Sim	Sim
R11	Não	Não	Sim	Sim	Sim	Sim
R12	Sim	Não	Não	Não	Não	Sim
R13	Sim	Não	Não	Sim	Sim	Sim
R14	Sim	Não	Sim	Sim	Sim	Sim
R15	Sim	Não	Não	Não	Sim	Sim
R16	Sim	Não	Não	Não	Sim	Sim
R17	Não	Não	Não	Não	Não	Não
R18	Sim	Não	Não	Sim	N/A	N/A
R19	Sim	Sim	Sim	Sim	Sim	Sim
R20	Sim	Não	Não	Não	N/A	N/A
R21	Sim	Não	Sim	Sim	Sim	Sim
R22	Sim	Não	Sim	Sim	N/A	N/A
R23	Sim	Sim	Sim	Sim	Sim	Sim
R24	Sim	Não	Não	Não	N/A	N/A
R25	Sim	Sim	Não	Sim	Sim	Sim
R26	Sim	Não	Não	Sim	Sim	Sim
R27	Sim	Sim	Sim	Sim	Sim	Sim
R28	Não	Não	Sim	Sim	Sim	Sim
R29	Sim	Sim	Sim	Não	Sim	Sim
R30	Sim	N/A	N/A	N/A	N/A	Sim
Legenda: N/A – Não se aplica						

Tabela 8. Satisfação dos Requisitos Desejáveis

Jogo x Requisito	Ilha dos Requisitos	EAReq-Game	*Software Quantum*	UbiRE	i* Game	RE-O-Poly
D1	Sim	Não	Sim	Sim	Não	Não
D2	Não	Não	Não	Não	Sim	Sim
D3	Não	Não	Não	Não	N/A	N/A
D4	Sim	Não	Não	Não	Sim	Sim
D5	Sim	Sim	Sim	Sim	Sim	Sim
D6	Não	Não	Não	Não	Não	Não
D7	Não	Não	Não	Não	Não	Não
D8	Não	Não	Não	Não	Não	Não
D9	Sim	Não	Não	Não	Sim	Sim
D10	Não	Sim	Sim	Sim	N/A	N/A
D11	Não	Não	Não	Sim	Não	Não
D12	Não	Sim	Não	Sim	Sim	Sim
D13	Sim	Não	Sim	Sim	Sim	Sim
D14	Não	Não	Não	Não	Não	Não
D15	Sim	Não	Sim	Não	Sim	Sim
D16	Não	Não	Não	Não	Não	Não
D17	Não	Sim	Não	Sim	N/A	N/A
D18	Não	Não	Não	Não	Não	Sim
Legenda: N/A – Não se aplica						

5. Discussão dos resultados

De acordo com os resultados apresentados, observa-se que os jogos de tabuleiro RE-O-Poly e i* Game, e o Ilha dos Requisitos são aqueles que mais atendem aos requisitos avaliados, superando ou igualando os 80% de requisitos obrigatórios sendo avaliados de forma positiva (eliminando os requisitos que não se aplicam). Por outro lado, o EAReq-Game é um jogo que possuiu uma avaliação muito negativa, correspondendo a menos de 33% dos requisitos obrigatórios atendidos. Os dois jogos restantes atendem na média entre 45% e 65% dos requisitos.

Em relação aos requisitos desejáveis, o RE-O-Poly e o i* Game mais uma vez se sobressaem, porém diminuindo a porcentagem para cerca de 50% de requisitos atendidos.

Observando a relação dos requisitos pedagógicos, dos dezesseis obrigatórios, o Ilha dos Requisitos atende a 75%, o EAReq-Game a apenas 25%, o *Quantum* a aproximadamente 44%, o UbiRE a aproximadamente 56%, o i* Game a 93% (sendo um dos dezesseis requisitos não aplicável), e o RE-O-Poly a 100% (sendo um dos dezesseis não aplicável).

Alguns problemas foram percebidos nos resultados, principalmente em relação à área de acessibilidade, onde nenhum jogo foi capaz de apresentar proposta para tornar-se mais acessível aos usuários com deficiências visuais. Outros problemas frequentes foram que os jogos não percebem as dificuldades dos jogadores e não oferecem ajuda decorrente destas; apenas um, a Ilha dos Requisitos, exibe instruções de uso sempre que solicitado. Nenhum jogo atende ao requisito desejável de ser adaptável ao jogador ou

aos sentimentos dele; todos são feitos de maneira a seguir apenas uma forma única de jogo. Nenhum jogo provê *feedback* sonoro para auxílio aos usuários.

De acordo com os resultados, conclui-se que, apesar de não respeitarem todos os requisitos, os jogos Ilha dos Requisitos, i* Game e RE-O-Poly podem se tornar prováveis bons complementos de ensino. Cada um foca uma diferente área, o que contribui para um maior conhecimento daqueles que utilizam esses jogos.

O fato de a análise ter sido realizada por apenas uma pessoa e a coleta de requisitos e a definição deles em requisitos obrigatórios ou funcionais, que foi realizada pelos autores deste artigo pode influenciar os resultados obtidos de maneira negativa. Pode-se futuramente, com mais profissionais da área, definir com mais exatidão quais os requisitos considerados obrigatórios e desejáveis.

6. Conclusão e trabalhos futuros

Ao final deste artigo, percebeu-se que é viável a utilização de alguns destes jogos, como a Ilha dos Requisitos, i* Game e RE-O-Poly, no ensino de ER. Eles apresentam uma forma complementar ao ensino tradicional, podendo motivar o interesse dos alunos, e fortalecimento do seu conhecimento nas áreas de conceitos de ER (Ilha de Requisitos), melhoria de modelos de requisitos organizacionais (i* Game) e conceitos relacionados a boas práticas de ER (RE-O-Poly).

Na área de definir a importância dos requisitos (se são obrigatórios ou desejáveis) há o jogo EAReq-Game, porém o mesmo se mostrou um jogo insatisfatório em relação aos critérios adotados. O jogo UbiRE mostra uma simulação de situação onde o jogador deve atender aos requisitos do cliente, sendo porém, em relação aos requisitos, um jogo intermediário. E o *Software Quantum*, também um jogo intermediário em relação aos requisitos, tenta fortalecer a importância da área de ER, demonstrando simulações onde o cliente conversa com o analista de projeto, mas sem explicar ou detalhar conceitos sobre ER.

O que pode ser considerado como ameaça a validade do artigo é o fato de pesquisas similares também terem realizado análise dos jogos, tendo como exemplo o trabalho [Wangenhein, 2009]. Além disto, a impossibilidade de utilizar o RE-O-Poly, juntamente com o fato de o Educatalog4RE ter sido construído para *softwares*, não contemplando em suas versões jogos de tabuleiro pode ter resultado em análises não precisas nos seus resultados.

Como recomendações e sugestões de trabalhos futuros, para evoluir a presente pesquisa, pode-se destacar a análise do uso de jogos em sala de aula, atualmente, na disciplina de ER, explorar novos tipos de jogos, como os videogames, e o desenvolvimento de jogos para cada uma das etapas da Engenharia de Requisitos de forma mais completa, abrangendo todo o conteúdo. Estes jogos podem combinar os pontos positivos das avaliações, e também atender tanto os requisitos obrigatórios quanto os desejáveis apresentados neste artigo.

Referências

Araujo, D. N. (2016). “Jogos Educativos na Área de Engenharia de Requisitos”, Monografia em Engenharia da Computação, UPE, Recife.

Chiavegatti, N., Petri, G. (2014) “EAReq-Game: Um Jogo Educacional para o Ensino de Elicitação e Análise de Requisitos”, Anais Do EATI - Encontro Anual de Tecnologia da Informação e Semana Acadêmica De Tecnologia Da Informação, Frederico Westphalen, Brasil.

Duque, M., Alencar, F., Cysneiros, G. e Torres, E. (2016) " Garanhuns Treasure Race: Turismo Educativo Gamificado em Garanhuns-PE". In: Proc. XV Simpósio Brasileiro de Jogos e Entretenimento Digital (SBGames'16), São Paulo: 8-10 Set.

Fernandes, M., et al. (2009) "A Requirements Engineering and Management Training Course for Software Development Professionals." 22th Conference on Software Engineering Education and Training.

Figueiredo, E., et al. (2006) "SimulES: Um Jogo para o Ensino de Engenharia de Software." III Fórum de Educação em Engenharia de Software.

Gil, R., et al. (2009) "Experiential learning approach for requirements engineering education." Requirements Engineering (ISSN: 0947-3602), vol. 14, num. 4, p. 269-287 Springer Verlag.

Henrique, M. S. et al (2015) "Uma Revisão Sistemática da Literatura sobre o uso de Teorias de Aprendizagem em Softwares Educacionais" Revista Renote v. 13, n. 2 Disponível em: http://seer.ufrgs.br/index.php/renote/article/view/61434. Acesso em: jul 2016.

Henrique, M. S. (2016) "Educatalog4re: Um Catálogo de Requisitos para Auxiliar o Desenvolvimento de Softwares Educacionais." Dissertação de Mestrado em Ciência da Computação, área de concentração em Engenharia de Software, UFPE, Recife.

Kitchenham, B. (2004) "Procedures for performing systematic reviews," Keele, UK, Keele Univ., vol. 33, no. 2004, pp. 1–26.

Knauss, E., Schneider, K., Stapel, K. (2008) "A Game for Taking Requirements Engineering More Seriously". Multimidia and Enjoyable Requirements Engineering (MERE)

Lima, T., et al. (2012) "UbiRE: A game for teaching requirements in the context of ubiquitous systems." Informatica (CLEI), XXXVIII Conferencia Latinoamericana En. IEEE, 2012

Moura, S. P. R., Barreto, M. F. T., Teixeira, R. A. G. (2013) "A Expressão de Compreensões a partir se Atividades com Softwares" V Congresso de Fenomenologia da Região Centro-Oeste. Disponível em: https://anaiscongressofenomenologia.fe.ufg.br/up/306/o/SimoneMoura.pdf. Acesso: jul/2016.

Peixoto M. M. and Silva, C. (2015) "Requisitos para Softwares Educacionais Gamificados: Uma Revisão Sistemática de Literatura". In: 18th Workshop on Requirements Engineering (WER 2015). Proc. of the XVIII Ibero-American Conference on Software Engineering. Arequipa: Universidad Católica San Pablo, p. 97-103.

Perrenoud, Ph. (2000). As práticas pedagógicas mudam e de que maneira? Revista Impressão Pedagógica (Curitiba), n° 23, Jul/Ago, pp. 14-15.

Pimentel, J., Santos, E., Pereira, T. (2012) "i*Game".

Smith, R., Orlena G. (2008) "RE-O-Poly: A Customizable Game to Introduce and Reinforce Requirements Engineering Good Practices." Departament of Computer Science, Pace University, New York.

Soffa, M. M. e Alcântara, P. R.C. (2008) "O Uso do Software Educativo: reflexões da prática docente na sala informatizada". Disponível em: http://www.pucpr.edu.br/eventos/educere/educere2008/anais/pdf/335_357.pdf. Acesso em: jul 2016.

Sommerville, I. (2011), Engenharia de Software, Addison-Wesley, 9th ed.

Thiry, M., et al. (2010) "Promovendo a Aprendizagem de Engenharia de Requisitos de Software através de um Jogo Educativo." XXI Simpósio Brasileiro de Informática na Educação.

von Wangenhein, C. G. et al (2009) “Revisão Sistemática sobre Avaliação de Jogos Voltados para Aprendizagem de Engenharia de Software no Brasil”. II Fórum de Educação em Engenharia de Software (FEES).

JoVeTest - Jogo da Velha para Auxiliar no Ensino e Estudo de Teste de Software

Ana Karoline T. Barbosa, Larissa L. E. Neves, Arilo C. Dias-Neto

Instituto de Computação – Universidade Federal do Amazonas
Manaus – AM – Brasil

{aktb,llen,arilo}@icomp.ufam.edu.br

Abstract. *Software testing has become a topic of extreme importance in the training of professionals in the software development area due to increasingly intense demand for software with high quality. However, this topic has a few hours in undergraduate courses and, moreover, it is a complex topic to be taught and absorbed by students. Thus, utilizing the most of the little time devoted to this topic is important, and new teaching resources and methodologies are important to maximize the result. Educational games have recently been explored as an alternative that motivates students to learn without compromising (and sometimes proving better results) the quality of education. This papers presents JoVeTest, an educational game based on the traditional tic-tac-toe game that aims to support the teaching software testing content. The game is currently composed of questions from international certifications in software testing (new questions can be easily included). An evaluation of this game was conducted with 20 students of a Verification and Validation course at UFAM aiming to analyze satisfaction and encouragement that the game offers to the students. The results indicate increased motivation and learning perception of students when using the game, as well as an increase in their grades on mini-tests conducted during the course.*

Resumo. *Teste de software tem se tornado um tópico de extrema importância na formação de profissionais na área de desenvolvimento de software devido à exigência cada vez mais intensa por software com alto índice de qualidade. No entanto, este conteúdo possui pouca carga horária nos cursos de graduação e além disso é um tópico complexo de ser ensinado e absorvido pelos alunos. Assim, aproveitar ao máximo o pouco tempo dedicado a este tópico é importante, e novos recursos didáticos e metodologias são importantes para maximizar o resultado. Jogos educacionais tem sido explorado recentemente como uma alternativa que motiva os alunos a aprenderem sem comprometer (e algumas vezes ampliar) a qualidade do ensino. Este artigo apresenta o JoVeTest, um jogo educacional baseado no tradicional jogo da velha que visa apoiar no ensino do conteúdo da disciplina de teste de software. O jogo é atualmente composto por questões oriundas de provas de cerificações internacionais em teste de software (podendo ser incluídas facilmente novas perguntas). Uma avaliação do jogo foi conduzida com 20 estudantes de uma disciplina de Verificação e Validação da UFAM a fim de analisar a satisfação e incentivo que o jogo oferece aos alunos. Os resultados indicam o aumento da motivação e percepção de ensino dos alunos ao usar o jogo, assim como incremento em suas notas em mini-exames para fixação de conteúdo.*

Palavras-chaves: *Jogo Didático, Jogo da Velha, Teste de Software.*

1. Introdução

Teste de software vem ampliando sua importância no cenário do desenvolvimento de software, deixando de ser apenas uma fase ou etapa do ciclo e tornando-se uma atividade contínua abrangendo todas as fases (DELAMARO et al., 2016). Um software com baixa qualidade ou que não foi suficientemente testado pode gerar prejuízos que vão desde uma simples má reputação adquirida pelos proprietários do software até prejuízos irreversíveis, como perdas financeiras ou até a morte de pessoas. No entanto, para que o software seja adequadamente testado, são necessários profissionais capacitados, e para isso, conceitos de teste e qualidade de software devem ser melhor disseminados ao longo dos cursos de graduação das áreas relacionadas.

De acordo com Beizer (1990), embora testes consumam mais da metade da vida profissional de um programador, menos de 5% da educação de um programador são dedicados a essa atividade. Apesar de antiga, esta referência apresenta um cenário ainda atual nos ambientes acadêmicos na área da computação. A apresentação tardia a conceitos importantes, metodologias de ensino pouco eficientes e atrativas aos alunos e carga horária reduzida para atividade fazem com que o ensino de teste de software seja pouco satisfatório para os profissionais da área (WANGENHEIM e SILVA, 2009). Tomando como exemplo dois cursos de computação da Universidade Federal do Amazonas (UFAM), no curso de Ciência da Computação o discente terá apenas uma disciplina sobre engenharia de software, incluindo testes de software, apenas no 5º de um total de 8 períodos com uma carga horária de 90 horas, o que representa 3,23% da carga horária obrigatória do curso (ICOMP, 2016). Já no curso de Sistemas da Informação, o tema é abordado em duas disciplinas de, respectivamente, 90 e 60 horas no 4º (Introdução à Engenharia de Software) e 5º (Verificação e Validação) períodos de um total de 8, totalizando 5,8% da carga horária obrigatória do curso (ICOMP, 2016).

Considerando ainda a mudança de perfil da população, em 2001 Marc Prensky (2001) cunhou o termo "Nativos Digitais" se referindo aos nascidos depois dos anos 80. Segundo ele, essa geração é capaz de ver TV, ouvir música, usar o celular e o notebook, tudo ao mesmo tempo. Ela é altamente familiarizada com a Internet e computador, e representa hoje 50% da população ativa, mas, segundo (MONTEIRO, 2009), em 2020 serão 80% da população total. Assim, as instituições de ensino e os professores têm um desafio cada vez maior em inovar em metodologias para que o conteúdo histórico seja dado de forma dinâmica. Entre os avanços na área de ensino, está a utilização de jogos educacionais, que são projetados para ensinar determinado assunto e expandir conceitos, reforçar o desenvolvimento, compreender um acontecimento histórico ou cultural, ou auxiliar na aprendizagem de uma habilidade e para estimular a motivação e o interesse do aluno (YEE, 2006). Vários jogos já foram propostos para auxiliar no processo ensino – aprendizagem de teste de software, tais como: O Jogo das 7 Falhas (DINIZ e DAZZI, 2011), *U-Test* (SILVA, 2010), *SimulES* (FIGUEIREDO et al., 2007) e *iTestLearning* (FARIAS et al. 2012; JORGE et al. 2015). Eles possuem diversas características, aspectos positivos e limitações, que são explanados na continuidade deste trabalho.

A partir deste cenário, o objetivo deste trabalho é a elaboração de uma versão do popular Jogo da Velha voltado para auxiliar no processo ensino-aprendizagem da disciplina de testes de software. Com isso, espera-se cooperar com a consolidação de conteúdo explorando o conceito de solidariedade e troca de saberes que, segundo Feldman (2013), são importantes para o processo de aprendizagem. O JoVeTest foi avaliado com 20 alunos de uma disciplina de Verificação e Validação da UFAM e os resultados indicam o aumento da motivação e percepção de ensino dos alunos ao usar o jogo, assim como incremento em suas notas em mini-exames para fixação de conteúdo.

Este trabalho foi organizado da seguinte forma: a Seção 2 apresenta alguns jogos

digitais já propostos para o ensino de testes de software. Na Seção 3 é apresentado o jogo da velha proposto (JoVeTest). A Seção 4 apresenta um comparativo com jogos apresentados na Seção 2. A Seção 5 relata um estudo conduzido com estudantes de uma disciplina de verificação e validação software para avaliação do jogo proposto. Finalmente, a Seção 6 discute as considerações finais e trabalhos futuros.

2. Trabalhos Relacionados

A partir do entendimento de que jogos educacionais são eficientes ferramentas para ensino e consolidação de aprendizagem (YEE, 2006), sugiram várias ideias e versões de jogos para ensino de testes de software. Destes, destacam-se o Jogo das 7 Falhas (DINIZ e DAZZI, 2011), *U-Test* (SILVA, 2010), *SimulES* (FIGUEIREDO et al., 2007) e o *iTestLearning* (JORGE et al., 2015), que serão explicados a seguir.

O Jogo das 7 Falhas (DINIZ e DAZZI, 2011) é um jogo de simulação que reproduz a execução de casos de teste derivados a partir de especificações de requisitos. Destinado a auxiliar no ensino da técnica de testes de caixa-preta, este é um jogo *single-player* no qual o jogador assume o papel de testador com a finalidade de descobrir no menor tempo possível as sete falhas existentes em cada funcionalidade testada, correlacionando as falhas com os critérios de geração de testes funcionais: particionamento em classe de equivalência e análise do valor-limite. Existem dois níveis de complexidade: baixa e média. Cada nível é composto por uma funcionalidade onde existem 7 falhas a serem descobertas. O jogador só passará de nível caso descubra as sete falhas existentes dentro do tempo estimado. Caso o tempo se esgote antes de o jogador identificar as 7 falhas em cada nível, o jogo se encerra. Caso ele descubra todas as falhas, ele é vencedor.

O *U-Test* (SILVA, 2010) é um jogo de simulação com foco específico em teste de unidade, abordando questões teóricas e práticas. O jogador assume o papel de um testador responsável por escrever teste de unidade para funções já escritas de um sistema hipotético. O objetivo do jogo resume-se à aplicação de técnicas para seleção de dados de entrada para o teste de unidade. Novamente, o jogador deverá aplicar critérios de geração de teste como particionamento em classes de equivalência e análise do valor limite. No início do jogo o jogador seleciona um projeto em que deseja trabalhar e na sequência são expostos vários desafios que ele deve superar. Ao final de cada desafio o guru de testes, que representa um especialista em testes, passa a questionar o jogador quanto às respostas fornecidas, a fim de verificar o embasamento teórico para as decisões tomadas. Como resultado de seu desempenho, ao final o jogador será informado de sua posição no ranking geral de jogadores.

O *SimulES* ou Simulador de Engenharia de Software (FIGUEIREDO et al., 2007) trata-se de um jogo de cartas. A dinâmica do jogo inicia com a escolha de uma carta de projeto, a qual contempla as informações sobre o projeto e suas restrições, como por exemplo qualidade e orçamento. Em seguida os participantes do jogo devem utilizar cartas de engenheiros para conceber, construir e testar a solução. Durante o jogo podem surgir cartas problemas, as quais simulam obstáculos comuns em projetos de desenvolvimento de software. O jogo se encerra no momento em que o primeiro jogador conclui o projeto escolhido no início do jogo.

Por fim, o *iTestLearning* (JORGE et al., 2015) é um jogo de simulação composto por três fases: planejamento, projeto e execução de teste de um determinado sistema a partir de descrições do mesmo. Na fase de planejamento, o aluno deverá planejar os testes passando por seis etapas definidas: itens de teste, tipos de testes, níveis de testes, critérios de aceitação, ferramentas e artefatos. Ao longo do progresso nas etapas vão sendo apresentados ao jogador diversos conceitos referentes a teste de

software. Caso o aluno obtenha pontuação mínima na fase de planejamento, ele passa à fase de projeto, que consiste em selecionar os casos de testes válidos de acordo com a descrição de uma especificação. Ao conseguir êxito na fase de projeto, o jogador é direcionado à fase de execução, na qual o sistema apresenta a descrição detalhada de um caso de uso e uma lista de casos de teste no qual o usuário fará uma análise dos casos a serem executados. A execução simulará o comportamento do sistema relacionado aos dados enviados e apresentará ao jogador as respostas sobre os eventos que serão feitas através de alertas ou telas de finalização de processos. O aluno deve observar essas respostas e marcar caso de teste válido quando o sistema apresentar um comportamento correto ou inválido quando apresentar comportamento inesperado.

A próxima seção apresenta o JoVeTest, um jogo digital baseado no famoso Jogo da Velha para apoiar o ensino e estudo de conceitos de teste de software.

3. Jogo da Velha para Ensino e Estudo de Teste de Software

Visando suprir as necessidades relacionadas à formação de novos profissionais e apresentação do conteúdo de forma mais efetiva e divertida, é proposto o JoVeTest – Jogo da Velha para ensino e estudo de teste de software, que será apresentado a seguir.

3.1. Por que o Jogo da Velha?

Há outros jogos com objetivos similares, e outros jogos já conhecidos poderiam ser adaptados para acolher a ideia. No entanto, as principais características para escolha do jogo da velha foram:

- Possui uma dinâmica de jogo de fácil compreensão com objetivo único e direto;
- Não requer cálculos ou esforço estratégico, muitas vezes conta-se com a distração do oponente para vencer;
- É jogado por duas pessoas, promovendo interação entre os oponentes, contribuindo na promoção do relacionamento interpessoal;
- São partidas de curta duração, provê o estímulo à competitividade saudável e é um exercício de atenção, pois em um breve momento de distração de um dos jogadores pode ocasionar sua derrota, portanto estimula o aluno a manter o foco.

3.2. Estrutura do Jogo

O JoVeTest segue a mesma dinâmica do popular jogo da velha. São dois jogadores, cada um possui um símbolo (X ou O), em que cada jogador joga alternadamente. O objetivo do jogo é fazer combinação do seu símbolo em linha, coluna ou diagonal. Ganha quem primeiro atingir o objetivo. O que caracteriza o JoVeTest como ferramenta de ensino é que para o jogador ter o direito de marcar seu símbolo, ele deve responder corretamente a uma questão de múltipla escolha relacionada ao tema Teste de Software.

A priori, as questões integrantes do JoVeTest foram retiradas das provas do CTFL (*Certified Tester Foundation Level*) realizadas nos anos de 2011, 2012, 2013, 2014 e 2015. Este exame é realizado periodicamente pelo ISTQB (*International Software Testing Qualifications Board*) em todo o mundo para certificação de profissionais na área de teste de software. Hoje a ferramenta contém 144 questões. No entanto, questões de qualquer outra fonte podem ser introduzidas na ferramenta de forma simples por meio de um arquivo TXT para incrementar seu conjunto inicial.

Para facilitar a utilização do jogo, o jogo funciona em *standalone* em um arquivo JAR que não necessita de qualquer configuração ou instalação. Basta baixar o arquivo e jogar. As questões já estão embutidas no pacote do jogo. Cada questão é estruturada com as informações descritas na Tabela 1.

Tabela 1. Estrutura das questões.

Campo	Informação armazenada
Pergunta	Enunciado da questão a ser respondida.
Imagem	Uma imagem pode (ou não) ser associada à questão, como parte de seu enunciado.
Opção A	Primeira alternativa de resposta.
Opção B	Segunda alternativa de resposta.
Opção C	Terceira alternativa de resposta.
Opção D	Quarta alternativa de resposta.
Resposta	Equivale à alternativa que contém a resposta correta da pergunta.
Nível	Varia de Fácil, Médio e Difícil, e refere-se ao grau de dificuldade da pergunta.
Tema/Aula	Refere-se ao tema/aula ao qual a questão está associada. Essa informação será usada para agrupar as questões por tema/aula no jogo.

3.3 O Jogo

Ao iniciar o JoVeTest, o usuário se depara com a tela de configurações iniciais, em que é necessário informar os nomes dos jogadores para cada símbolo (O e X) e selecionar o(s) tema(s) sobre o qual(is) deseja responder às questões (Figura 1).

Figura 1. Tela Inicial do JoVeTest.

Inicialmente devem ser informados os nomes dos dois jogadores e pelo menos um tema deve estar marcado para que seja possível iniciar o jogo. Caso os requisitos de configurações iniciais sejam preenchidos, ao clicar em iniciar o usuário é direcionado para a tela do Jogo, que possui as seguintes informações (Figura 2):

1. Indicação de quem é a vez de jogar e qual o símbolo;
2. Informações do jogo: contagem de vitórias de cada jogador e de empates;
3. Controles do Jogo: apresenta 3 botões que possuem as seguintes funções, respectivamente: Reiniciar partida, Instruções/informações sobre o jogo e Sair;
4. Tabuleiro: formado por três linhas e três colunas, onde ocorrerão as marcações dos símbolos dos jogadores;

Figura 2. Tabuleiro do JoVeTest.

Quando o jogador clica na posição que deseja marcar seu símbolo, ocorre o sorteio de uma questão que pertença a algum tema selecionado nas configurações iniciais do sistema, e que ainda não tenha sido sorteada na partida em andamento. Então a pergunta é exibida sobre o tabuleiro. O jogador precisa responder corretamente à questão para adquirir o direito de marcar seu símbolo no local selecionado.

Na tela de exibição da questão sorteada, são apresentadas as seguintes informações (Figura 3 – esquerda):

1. Enunciado da pergunta a ser respondida.
2. Imagem relacionada à questão, se houver, ou imagem padrão, que é a exibida nesse caso da Figura 3.
3. Alternativas de resposta para a questão.
4. Identificação do grau de dificuldade da questão sorteada.
5. Botão para confirmar resposta.

Se o Jogador clicar no botão de confirmar resposta sem marcar uma alternativa, é exibida notificação de que é necessário responder à pergunta. Quando jogador efetua a escolha de alternativa, mas a marcação escolhida não é a resposta certa, é exibida mensagem de resposta incorreta, é emitido som de notificação de erro, a alternativa correta é destacada em verde e não ocorre marcação do símbolo do jogador no local escolhido. Caso o jogador marque alternativa correta, é exibida notificação de que a resposta está correta, a opção marcada é destacada em verde, é emitido um alerta sonoro de aplausos e quando o jogador fecha a tela da questão é efetuada a marcação do símbolo do jogador no lugar escolhido (Figura 3 – direita).

Quando algum dos jogadores atinge o objetivo do jogo, é exibida uma animação parabenizando o jogador vencedor, a contagem de vitórias é atualizada e o tabuleiro é

zerado para nova partida. Caso o jogo termine empatado, o contador de empate é atualizado, exibido notificação de empate e o tabuleiro é reiniciado para nova partida. A qualquer tempo o jogador pode acionar o botão Ajuda. Caso isso ocorra, é exibida a tela com instruções/informações sobre o jogo.

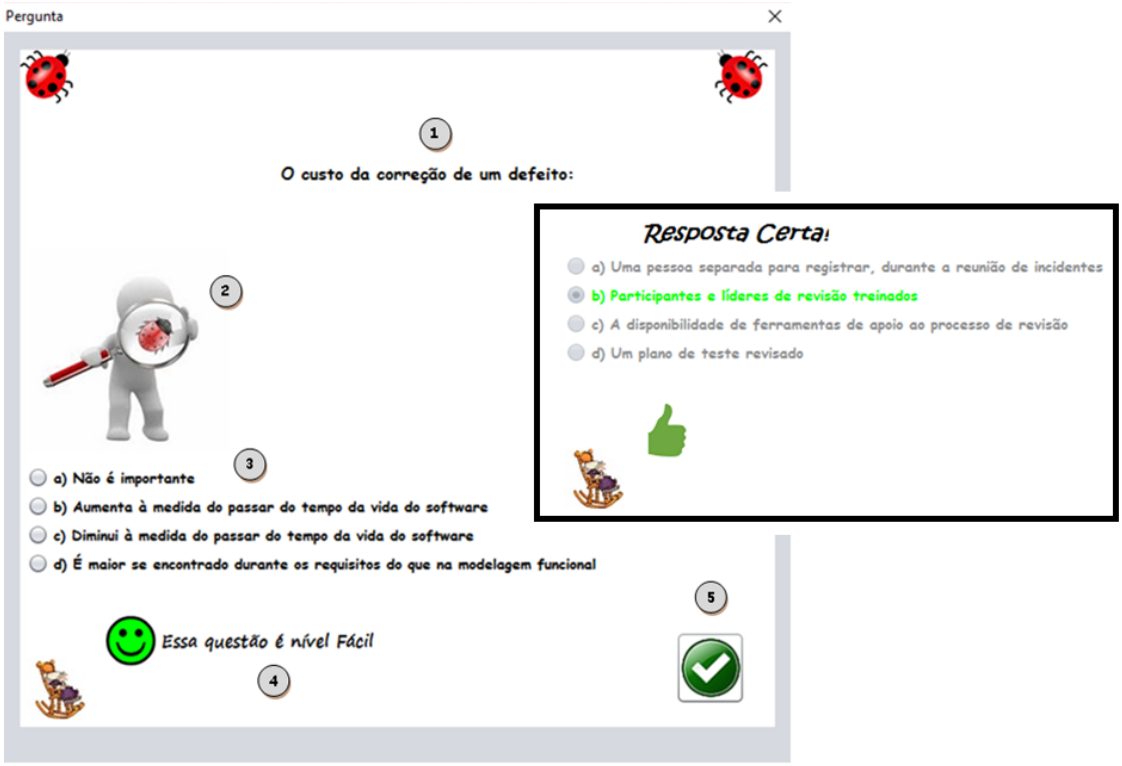


Figura 3. (esquerda) Exibição de pergunta e (direita) indicação de resposta certa.

JoveTest pode ser obtido no endereço http://experts.icomp.ufam.edu.br/jovetest. A próxima seção apresenta uma análise comparativa do JoVeTest com outros jogos de apoio ao ensino/estudo de teste de software.

4. Comparativo com Outros Jogos

Para realizar a análise comparativa do JoVeTest com outros jogos, serão utilizados os seguintes critérios:

1. Gênero do jogo – para esse item será usada a classificação de jogos digitais sugerida por (HERZ, 1997), e (DEMPSEY et al., 2003), que são: ação, aventura, luta, Puzzle, RPG, Simulação, Esporte, Estratégia, Competição, Cooperação, Programação e Gerenciamento e Negócio.
2. Se possui os seis elementos estruturais dos jogos de acordo com (PRENSKY, 2007): Metas e objetivos; Regras; Desfecho e Feedback; Conflitos, Competição e Desafios; Interação; Representação, História ou Narrativa;
3. Se foca no desenvolvimento de habilidades/ensino de teste de software;
4. Se fornece feedback ao usuário quanto às suas ações/ao seu desempenho no jogo;
5. Se pode ser utilizado sem a presença de um instrutor;
6. Se é disponibilizado livremente para uso;
7. Se possui avaliação de efetividade de seu uso;
8. Plataforma utilizada;
9. Classificação do jogo quanto à interação;

A Tabela 2 mostra o comparativo realizado considerando as características definidas acima.

Tabela 2. Comparativo entre jogos.

Característica analisada	**Jogo das 7 Falhas**	***U-Test***	***SimulES***	***iTestLearning***	**JoVeTest**
Gênero	Simulação	Simulação	Simulação	Simulação	Competição
Possui os elementos estruturais dos jogos?	SIM	SIM	SIM	SIM	SIM
Foca no desenvolvimento de habilidades/ ensino de teste de software;	SIM	SIM	Parcialmente	SIM	SIM
Fornece feedback ao usuário quanto a suas ações e desempenho no jogo	SIM	SIM	SIM	SIM	SIM
Pode ser utilizado sem a presença de um instrutor;	SIM	SIM	Parcialmente	SIM	SIM
É disponibilizado livremente para uso;	SIM	NÃO	SIM	SIM	Será
Possui avaliação de efetividade de seu uso	SIM	SIM	Parcialmente	SIM	NÃO
Plataforma	Desktop	WEB	Não digital	WEB	Desktop
Classificação do jogo quanto à interação;	Jogador único	Múltiplos jogadores individuais	Jogador VS. jogador	Jogador único	Dois Jogadores

Ao fim da comparação, nota-se que os jogos analisados possuem diferentes abrangências e focos dentro do universo de Engenharia de Software/Teste de Software:

- Jogo das 7 falhas foca exclusivamente em teste de caixa preta, e não há a possibilidade de inclusão de novos casos de teste. Logo, o jogo não pode ser utilizado em diferentes temas no decorrer da disciplina/curso, pois após algumas partidas o jogador pode decorar as respostas, além de não envolver a parte teórica e conceitual;
- *U-Test* integra a parte teórica e conceitual e apresenta-se como ferramenta promissora, mas possui foco especifico em teste de unidade;
- *SimulES* é uma boa ferramenta para que o aluno possa testar conhecimentos adquiridos com relação à engenharia de software, mas não foca em teste de software;
- *iTestLearning* é a ferramenta mais completa das analisadas: apoia no processo de planejamento, projeto e execução de teste, dispõe de apoio teórico e conceitual, mas foi projetada para ser jogada por apenas um jogador, sem explorar a competição entre os jogadores, e de acordo com o filósofo Thomas Hobbes (2012) a natureza humana é naturalmente competitiva.

Portanto, constata-se que os jogos apresentados são promissores para auxiliar no ensino de teste de software dentro do objetivo para o qual cada um foi desenvolvido. O JoVeTest integra o grupo de ferramentas promissoras porque além das características comuns entre os demais jogos, explora a natureza competitiva do homem com efeito benéfico ao aprendizado e pode ser adaptado e utilizado de acordo com o andamento do curso/disciplina.

A próxima seção descreve um estudo realizado com estudantes de uma disciplina de teste de software para avaliação do nível de aceitação do JoVeTest.

5. Avaliação do Jogo com Estudantes da Disciplina Verificação e Validação

Nesta seção é descrita a avaliação do JoVeTest com estudantes na disciplina Verificação e Validação do curso de Sistemas de Informação da Universidade do Amazonas. Essa avaliação teve como objetivo analisar o incentivo que o JoVeTest oferece aos estudantes e o possível impacto do seu uso na satisfação e eficácia em relação ao apoio à absorção do conteúdo apresentado em sala de aula.

5.1 Planejamento do Estudo

Para o estudo, participaram 20 alunos da disciplina Verificação e Validação. A turma foi dividida aleatoriamente em dois grupos, chamados de Grupo A e B. Foram selecionadas duas aulas da disciplina com tópicos relacionados à qualidade de processo (normas de teste) do dia 09/06/2016 e qualidade do produto (técnicas de teste) do dia 14/06/2016. Assim, foram geradas duas versões do jogo contendo apenas questões referentes a cada aula citada. Nesta divisão em dois grupos, alunos do grupo A utilizaram o JoVeTest para estudar e fortalecer o conteúdo da aula do dia 09/06/2016 (qualidade do processo), enquanto que alunos do grupo B utilizaram o JoVeTest para estudar e fortalecer o conteúdo da aula do dia 14/06/2016 (qualidade do produto). O objetivo foi minimizar o impacto do tópico da aula nos resultados da aplicação do jogo. Assim, os grupos se dividem no uso do jogo em duas aulas do curso.

Os participantes foram caracterizados em relação a alguns itens: Em relação à faixa etária, são estudantes entre 21 e 32 anos. 30% dos alunos são mulheres e 70% são homens. 55% disseram que SIM, costumam revisar o conteúdo periodicamente, 35% disseram que algumas vezes e 10%, apenas quando terá prova. 100% disseram que estudam sozinhos, 15%, em dupla e 25%, em grupo (poderia ser escolhida mais de uma opção). 100% dos alunos participantes nunca usaram anteriormente jogos educacionais. 90% disseram que acreditam que um jogo pode facilitar a consolidação do conteúdo e 10% disseram não saber se há influência. Finalmente, 80% disseram que um jogo motiva a estudar e 20% mostraram-se indiferentes.

Todos os alunos participantes do estudo responderam a um pré-questionário sobre suas preferências de estudo e uso de jogos educacionais na área de computação. Após o uso do jogo na data relacionada ao seu grupo, os alunos responderam a um pós-questionário para que eles opinassem sobre a experiência de usar o jogo como apoio ao ensino de teste de software. Os itens desses questionários serão apresentados na descrição dos resultados do estudo. Além disso, na disciplina, após cada aula teórica os alunos são submetidos na aula seguinte a um mini-exame com cinco questões objetivas sobre o assunto da aula anterior. Todos os alunos da disciplina foram submetidos a este mini-exame independente de ter usado ou não o JoVeTest como apoio ao estudo do conteúdo. O objetivo não é investigar se o jogo possibilita aumento das notas dos alunos, mas sim avaliar o aumento da motivação e entusiasmo pelo conteúdo apresentado.

A próxima subseção apresenta os resultados obtidos a partir da experiência de uso do JoVeTest.

5.2 Análise dos Resultados

A Tabela 3 apresenta as questões analisadas após a utilização do jogo e distribuição de respostas entre os alunos. Pode-se observar que para as cinco questões o índice de

respostas positivas (provavelmente ou definitivamente sim) foi amplamente superior. 95% (55% + 45%) dos participantes entenderam que JoVeTest facilitou a consolidação dos assuntos da aula, 85% acreditam que JoVeTest ajuda na motivação para revisar o assunto das aulas, 90% indicaram que JoVeTest tornou o ensino/aprendizado mais satisfatório e 100% dos alunos acharam o jogo fácil de ser aplicado. Por fim, todos os alunos indicaram que se o jogo se tornasse um aplicativo móvel para ser jogado em dupla pela Internet seria algo que ajudaria no estudo. Isso é um trabalho futuro a este trabalho.

Tabela 3. Questões sobre o uso do JoVeTest.

Questões	Definitivamente Não	Não há diferença	Provavelmente Sim	Definitivamente Sim
1. A utilização do JoVeTest facilitou a consolidação dos assuntos tratados?	0%	5%	55%	40%
2. O JoVeTest traz diferença quanto a motivação em revisar o assunto?	0%	15%	40%	45%
3. A utilização do JoVeTest torna o ensino/estudo mais satisfatório?	0%	10%	40%	50%
4. O jogo aplicado é de fácil compreensão?	0%	0%	25%	75%
5. Se existisse um aplicativo móvel com o jogo para ser jogado em dupla pela internet, ajudaria no estudo?	0%	0%	25%	75%

Os alunos também deram uma **nota** de 1 a 10 para o JoVeTest e então calculamos a média das notas, que ficou em **8**. Apenas um aluno deu nota baixa (3), argumentando que jogos educacionais mais inovadores poderiam ser providos e o JoVeTest seria algum mais tradicional e simples

Também foram analisadas as notas dos alunos nos mini-exames citados anteriormente, separando as notas entre quem usou o jogo e não usou o jogo. A Tabela 4 apresenta esses resultados analisando a média e desvio padrão por grupo e no geral. Os dados indicam que em todos os grupos e também na análise geral que o uso de JoVeTest, em média, resultou em uma maior nota na avaliação dos mini-exames de fixação de conteúdo. O desvio padrão das notas entre os alunos que usaram JoVeTest foi igual ou mais baixo que dos alunos que não usaram em uma dada aula, indicando que o JoVeTest contribuiu um pouco para a melhoria no ensino e tornou o grupo mais homogêneo, reduzindo a distância entre os alunos na avaliação.

Tabela 4. Análise das notas dos mini-exames com e sem o uso do JoVeTest.

Grupo	Sem JoVeTest	Com JoVeTest
Grupo A	Média: 6,8 | DP: 2,42	Média: 7,1 | DP: 2,42
Grupo B	Média: 5,5 | DP: 2,56	Média: 8,35 | DP: 1,33
GERAL	Média: 6,13 | DP: 2,21	Média: 7,76 | DP: 2,10

Os alunos também foram perguntados se desejavam que o jogo fosse aplicado em outros conteúdos do curso, escolhendo algumas razões (em caso positivo ou negativo). A Tabela 5 apresenta a distribuição das respostas entre as razões apresentadas. É possível perceber que um jogo como recurso didático diverte o aluno durante as atividades de ensino, trazendo novidades para o ambiente de sala de aula, mas que o aluno não entende isso como uma substituição às aulas “tradicionais”, o que seria de fato o objetivo do jogo (contribuir, não substituir). Nenhum aluno indicou que não gostaria de usar JoVeTest na continuidade do curso.

Tabela 5. Razões para continuar ou não o do JoVeTest no curso.

RESPOSTAS POSITIVA	100%	RESPOSTAS NEGATIVA	0%
RAZÕES			
O jogo é divertido, agradável, descontrai e distrai	35%	Porque aula é aula	0%
Aprendemos com o colega de uma forma diferenciada	25%		
É mais fácil de aprender, fixar o conteúdo e torna o assunto mais divertido	70%	Acho idas ao laboratório mais proveitosas	0%
É mais legal que ter aula	15%		
As vezes é necessário fazer atividades diferentes para sair do tradicional	50%		
Porque nos ajuda a relacionar o conteúdo	30%		

Assim, este estudo é finalizado e a próxima seção apresenta as conclusões e trabalhos futuros que darão continuidade a este trabalho.

6. Conclusões e Trabalhos Futuros

A área de Teste de software ainda sofre com a carência de capacitação e tempo resumido para o ensino, como observa- na grade curricular de cursos da área de computação da UFAM. Assim, alternativas que visem facilitar o processo ensino–aprendizado são sempre bem-vindas e utilizadas de acordo com necessidade de cada turma e curso.

O JoVeTest – Jogo da Velha para ensino e estudo de teste de Software – apresenta-se como uma ferramenta promissora para dar suporte aos professores/tutores na tarefa de formar bons profissionais de testes, tornar aulas mais dinâmicas e alinhar o teórico ao lúdico, proporcionando estudos de forma dinâmica e flexível. Neste jogo, cada jogador busca formar combinações do seu símbolo, mas para adquirir o direito de marcar no local escolhido deve responder corretamente a uma pergunta relacionada a teste de software. Vale ressaltar que jogos educacionais são eficientes ferramentas para ensino e consolidação de aprendizagem (YEE, 2006), não como método único e padrão, mas associado a outras metodologias, como aulas expositivas.

O jogo foi avaliado com estudantes de um curso de graduação da UFAM, que representam o público-alvo do jogo. Os resultados indicam que os participantes concordam que o jogo aumenta a motivação para o estudo do conteúdo, que o jogo facilita a consolidação do conteúdo e facilita o ensino, além de ser fácil utilização. Em média, os participantes deram nota 8 para o jogo. Dentre as justificativas para continuar o seu uso, a introdução de uma abordagem mais divertida, lúdica e descontraída, diferente dos métodos convencionais, foi apontada pela maioria dos participantes. O jogo ainda contribuiu para um melhor aproveitamento do conteúdo das aulas, o que foi medido por meio de mini-exames (tradicionais no curso) nos quais o grupo que usou o jogo obteve um desempenho melhor e mais homogêneo nas avaliações.

Como trabalhos futuros, espera-se ampliar o conjunto de questões disponíveis no jogo, corrigir dificuldades pontuais relatadas pelos alunos que usaram o jogo e estender a ferramenta para um aplicativo móvel na plataforma Android, permitindo que duas pessoas joguem/estudem de forma online. Além disso, estudos experimentais mais aprofundados e com análises estatísticas mais abrangentes estão previstos a fim de avaliar o impacto do jogo no ensino de teste de software.

7. Agradecimentos

Os autores agradecem á FAPEAM pelo apoio para a condução desta pesquisa.

Referências

BEIZER, B. (1990). Software Testing Techniques. 2 ed. New York: Van Nostrand Reinhold.

DELAMARO, M. E.; MALDONADO, J. C.; JINO, M. (2016). Em: Introdução ao teste de software. 1ed.Rio de Janeiro - RJ: Editora Campus. 2016.v. 2.

DEMPSEY, J. LUCASSEN, B. GILLEY, W. RASMUSSEN, K. (2003). Since Malone's theory of intrinsically motivating instruction: Whats the score in the gaming literature? Journal of Educational Technology Systems, 22(2), 173-183.

DINIZ, L.L., DAZZI, R. L. S. (2011) Jogo Digital para o Apoio ao Ensino de Teste de Caixa-Preta. In: X Simpósio Brasileiro de Qualidade de Software (SBQS) Curitiba.

FARIAS, F., MOREIRA, C., COUTINHO, E., SANTOS, I. S. (2012). *iTestLearning*: Um Jogo para o ensino do planejamento de testes de software. In: V Fórum de Educação em Engenharia de Software (FEES 2012). Natal.

FELDMAN, L. (2013): Qual a melhor forma de estudar: Sozinho ou em grupo? Reportagem do Jornal UOL de 21/07/2013 disponível em: http://ne10.uol.com.br/canal/vestibular-2013/noticia/2012/10/24/qual-a-melhor-forma-de-estudar-sozinho-ou-em-grupo-376272.php, acessado em 08/03/2016

FIGUEIREDO, E.; LOBATO, C., DIAS, K.; LEITE, J.; LUCENA, C. (2007). Um jogo para o ensino de engenharia de software centrado na perspective da evolução. Anais do XV Workshop sobre educação em computação (WEI), pp. 37-46. Rio de Janeiro.

HERZ, J. C. (1997). Joystick Nation: how videogames ate our quarters, won our hearts, and rewired our minds. Little, Brown & Company.

HOBBES, T. (2012). Locke – Rosseau, disponível no endereço: https://studyabroaders.wordpress.com/2012/10/19/hobbes-locke-rousseau/. Acessado em 11/05/2016.

ICOMP (2016), Instituto de Computação UFAM: www.icomp.ufam.edu.br, visitado em 07/06/2016;

JORGE, F. de F.; BEZERRA, C. I. M.; COUTINHO, E. F.; MONTEIRO, J.M.; ANDRADE, R. M. C. (2015). A Evolução do Jogo *iTestLearning* para o ensino das atividades de execução de testes de software. Em: Conferência Internacional sobre Informática na Educação (TISE), Santiago, Chile.

MONTEIRO, E. (2009). Nativos Digitais já estão dominando o mundo e transformando a forma como o ser humano se comunica. Reportagem do O Globo de 18/05/2009. Disponível em: http://fndc.org.br/clipping/nativos-digitais-ja-estao-dominando-o-mundo-e-transformando-a-forma-como-o-ser-humano-se-comunica-379162/. Acesso em 08/06/2016.

PRENSKY, M. (2007). Digital Game-based learning. Saint Paul: Paragon house.

PRENSKY, M. (2001). Nativos digitais, Imigrantes digitais. On the Horizon. Outubro.

SILVA, A. C. (2010) Jogo Educacional para apoiar o ensino de técnicas para elaboração de teste de unidade. Dissertação de Mestrado. Univali São José;

WANGENHEIM, C. G.; SILVA, D. A. (2009). Qual conhecimento de engenharia de software é importante para um profissional de software? In Anais do Fórum de Educação em Engenharia de Software, Fortaleza.

YEE, N. (2006). The labor of fun: how video games blur the boundaries of work and play. Games and Culture, v.1, p.68-71.

Obsolescência profissional em engenheiros de software: Uma revisão sistemática da literatura

Bruno do S. Rocha[1], César França[2]

[1]Centro de Estudos e Sistemas Avançados do Recife (CESAR)
R. do Brum, 77 - Recife, PE, 50030-260

[2]Departamento de Informática e Estatística (DEINFO)
Universidade Federal Rural de Pernambuco.

bdsr74@gmail.com, cesar@franssa.com

Abstract. *This paper aims to present a systematic review of the literature about the obsolescence in information technology (IT) professionals. After a research in 438 articles, we extracted 20 relevant studies of which we collected factors that contribute to professional obsolescence and identified its antecedents and consequences in the organization and individuals perspective; how individuals react and handle with the threat of obsolescence; individual and joint actions to avoid the professional obsolescence.*

Resumo. *Neste trabalho foi apresentada uma revisão sistemática da literatura sobre a obsolescência em profissionais de tecnologia da informação (TI). Após uma busca em 438 artigos, foram identificados 20 estudos relevantes para o tema. Como resultado da pesquisa, foram coletados fatores que contribuem para a obsolescência e os estados que a antecedem; suas consequências, na visão da organização e do indivíduo; reações do indivíduo frente ao risco da obsolescência e como ele lida com a mesma; e por fim, ações individuais e conjuntas para mitigar o risco da obsolescência profissional.*

1. Introdução

Um dos maiores desafios do profissional de tecnologia da informação (TI) é ter a capacidade de se adaptar às constantes mudanças impostas pela indústria. Diferente de outras profissões, onde o conhecimento e habilidades se dissolvem mais lentamente, estima-se que o tempo de vida útil do domínio técnico de um profissional de TI seja de apenas dois anos [Joseph 2001]. Por este motivo, o risco da obsolescência é um dos principais fatores de stress em profissionais de tecnologia da informação em todo o mundo [Rajeswari & Anantharaman 2003].

A defasagem profissional ou obsolescência é descrita como o grau onde os profissionais de uma organização não têm o conhecimento ou habilidades necessárias para manter um desempenho eficaz tanto em seus papéis de trabalho atuais, quanto em futuros [Dubin 1972]. Ela pode ser dividida em dois tipos: econômica e técnica. A primeira diz respeito à perda de valor do mercado de trabalho por fatores relacionados a mudanças na estrutura do setor, a profissão em si ou devido a uma reorganização ou mesmo fechamento de uma empresa. Já a obsolescência técnica diz respeito ao

indivíduo e ocorre por desgaste (idade, doença, lesão, etc) ou atrofia, atribuída à pouca utilização das habilidades do profissional [De Grip 2006]. Neste trabalho daremos ênfase à segunda abordagem.

Neste estudo, foi realizado um esforço para mapear o conhecimento atual sobre a obsolescência técnica, suas principais causas e ações que ajudem o indivíduo a manter-se apto a atender às exigências do mercado de trabalho, em especial para a área de engenharia de software. Nas seções seguintes serão discutidos, respectivamente, conceitos gerais relacionados com a obsolescência profissional, a metodologia aplicada, resultados e discussões, e por fim, as conclusões sobre o estudo.

2. Obsolescência em profissionais de TI

Entende-se por conhecimento o resultado da assimilação de informações através do aprendizado, ou em outras palavras, o acervo de fatos, princípios, teorias e práticas relacionadas a um campo de trabalho ou estudo. É o resultado do processo de compreensão, entendimento e aprendizado [Swigon 2011].

Quando o indivíduo ingressa no mercado de trabalho, ele conclui a etapa de preparação para a vida profissional e passa a administrar a própria base de conhecimentos [Miller & Dettori 2008]. A gestão do conhecimento é um processo de criação, captura e utilização do saber para a melhoria da performance individual ou organizacional [Jefferson 2006]. No âmbito empresarial, significa manter-se preparado para as constantes mudanças impostas pelo mercado e tomá-las como vantagem competitiva para se colocar em uma posição privilegiada frente aos seus concorrentes [Alavi & Leidner 2001].

É correto afirmar que um profissional alcançou a obsolescência, quando já não possui mais o conhecimento ou habilidades necessárias para manter um desempenho eficaz em seu papel de trabalho atual ou em futuros [Dubin 1972]. Uma das consequências da obsolescência técnica é a incidência de profissional com um vasto conhecimento em áreas pouco usuais e possivelmente com habilidades pessoais bem desenvolvidas, porém, carente de conhecimento técnico atual. Fu (2011) relaciona esse fenômeno com o grau de comprometimento do indivíduo com a sua carreira. De Grip (2006) coloca a busca pela atualização como uma corrida, onde os vencedores recebem como premiação a oportunidade de permanecerem exatamente onde estão. Por esse motivo, o risco de obsolescência profissional está presente entre os maiores causadores de stress em profissionais de TI [Joseph 2001; Joseph et al. 2011; Rajeswari & Anantharaman 2003; Setor et al. 2015].

3. Metodologia

Pesquisadores utilizam revisões sistemáticas da literatura como forma de identificar, avaliar e interpretar estudos empíricos realizados sobre um tema, uma pergunta de pesquisa ou fenômeno de interesse comum [Kitchenham & Charters 2007]. Essa seção descreve a metodologia utilizada para executar a revisão sistemática da literatura materializada neste estudo, a qual foi baseada no guia para desenvolvimento de revisões sistemáticas em engenharia de software, descrito por Kitchenham [Kitchenham & Charters 2007] e no modelo implementado por Ghapanchi, no artigo "*Antecedents to IT*

personnel's intentions to leave: A systematic literature review" [Ghapanchi & Aurum 2011].

Kitchenham considera três fases principais para uma revisão sistemática: Planejamento da avaliação, realização da revisão e construção de um relatório sobre a revisão [Kitchenham & Charters 2007]. Neste trabalho, dividimos tais fases em seis passos: (1) Identificação dos recursos, (2) Definição dos critérios de inclusão/exclusão, (3) busca por estudos relevantes, (4) extração dos dados, (5) síntese dos dados e (6) escrita do estudo no formato de um relatório. O objetivo desta revisão é identificar estudos que abordem a defasagem (ou obsolescência) de profissionais de tecnologia da informação, especialmente engenheiros de software. Os estudos preliminares foram conduzidos pelas seguintes questões:

- *O que um profissional da engenharia de software faz para evitar tornar-se obsoleto?*

A partir desta pergunta central, outras três questões secundárias foram desenvolvidas, para ajudar na compreensão do problema:

- *O que leva à obsolescência do profissional da engenharia de software?*
- *Como o profissional da engenharia de software lida com o risco da obsolescência?*
- *Como o profissional da engenharia de software lida com a obsolescência?*

3.1. Critérios de inclusão e exclusão

Para esta revisão, foram considerados artigos acadêmicos que analisam o fenômeno "profissionais obsoletos para o mercado, ainda em idade produtiva". Respeitando a volatilidade da área de tecnologia da informação, serão considerados apenas os trabalhos publicados a partir de janeiro de 2000. Serão excluídos também:

- Estudos publicados em outro idioma que não o inglês
- Estudos não disponíveis para consulta online
- Chamadas para congressos, prefácios, anais de conferências e entrevistas
- Estudos que falam sobre obsolescência do software e não do profissional
- Trabalhos que se baseiam meramente em opiniões de especialistas
- Trabalhos com inconsistências na metodologia ou coleta dos dados

3.2. Estratégia de pesquisa

As fontes de dados utilizadas neste estudo foram:

- ACM Digital Library
- IEEE Xplore
- ScienceDirect
- Scopus

A combinação de termos relacionados ao tema foi idealizada para evitar que estudos relevantes fossem descartados indevidamente, mas com o cuidado de rejeitar trabalhos sem conexão com o tema abordado. Como resultado, as palavras-chave para esta pesquisa foram "Software engineering", "IT", "Obsolescence" e "Half-life".

A partir destas palavras-chave, as seguintes expressões de busca foram criadas e utilizadas nas bases selecionadas:

1. "Software engineering" AND Obsolescence
2. "Software engineering" AND "Half-life"
3. IT AND Obsolescence
4. IT AND "Half-life"

A pesquisa foi executada no mês de abril de 2016. O resultado de cada busca em separado foi agrupado por fonte de dados. Como todas as fontes de dados selecionadas são ricas nos mais variados estudos, as buscas foram limitadas às áreas de ciências da computação, psicologia, gestão empresarial e tecnologia da informação e comunicação. Desse modo, foi possível descartar de imediato publicações relacionadas às ciências gerais, genética, saúde e matemática, entre outras que não são o foco deste estudo. Como resultado tivemos:

Tabela 1. Quantidade de estudos encontrados em cada banco de dados

Banco de dados	Número de estudos
ACM Digital Library	60
IEEE Xplore	215
ScienceDirect	163
Scopus	28

3.3. Seleção dos trabalhos

Esta seção descreve todo o processo de seleção dos estudos. O primeiro passo foi eliminar as redundâncias. Para tal, utilizou-se a ferramenta Mendeley. A pesquisa inicial trouxe 466 trabalhos. Após eliminar as duplicidades (publicações presentes em mais de uma base de dados), restaram 438 publicações. O segundo passo foi analisar o título de cada trabalho selecionado, para determinar a relevância dos mesmos para esta revisão. Foram eliminados todos os trabalhos não relacionados a aspectos humanos em tecnologia da informação ou defasagem computacional, restando 63 estudos. O terceiro passo foi ler o resumo (*abstract*) das publicações restantes, para então remover aquelas que não estão relacionadas, de forma estreita ou geral, com a obsolescência técnica ou profissional. Nesse processo foram eliminadas 39 publicações, restando, de um total de 438 trabalhos, 24 estudos relacionados com o objeto desta pesquisa, conforme demonstrativo abaixo:

Tabela 2. Quantidade de estudos filtrados por etapa de seleção

Fase do processo de seleção	Número de estudos
1. Pesquisa nas bases de dados	466
2. Eliminação das redundâncias	438
3. Análise dos títulos	63
4. Análise dos resumos	24

3.4. Avaliação dos trabalhos

Após a identificação da lista primária dos estudos, análise dos títulos e análise dos resumos, foi necessário ainda um quarto passo, que previa a exclusão de trabalhos, com base no texto integral da publicação.

Para tal, foram formuladas questões para avaliar a qualidade do material produzido e verificar se poderiam integrar o estudo proposto. Desse modo foram realizadas 5 questões preliminares:

1. O estudo examina a obsolescência com foco no profissional?
2. O estudo é baseado em uma pesquisa (não apenas opinião de especialistas)?
3. O estudo propõe algum modelo ou processo que ajude a solucionar o problema estudado?
4. Os métodos utilizados para coleta dos dados foram adequados?
5. A conclusão do estudo está coerente com a coleta e análise dos dados?

Após responder às perguntas propostas, dos 24 estudos remanescentes, apenas 11 passaram para a fase seguinte.

E por fim, para garantir uma cobertura e coerência maior ao estudo proposto, foi realizada uma busca nas referências dos trabalhos pesquisados, para detectar estudos em comum e que não haviam sido identificados na busca inicial, aplicando aos mesmos as análises previstas nos passos anteriores (análise do título, resumo e texto integral). Nesta etapa foram adicionadas ao acervo 9 publicações. A execução de todas as etapas do processo pode ser observada na Figura 1.

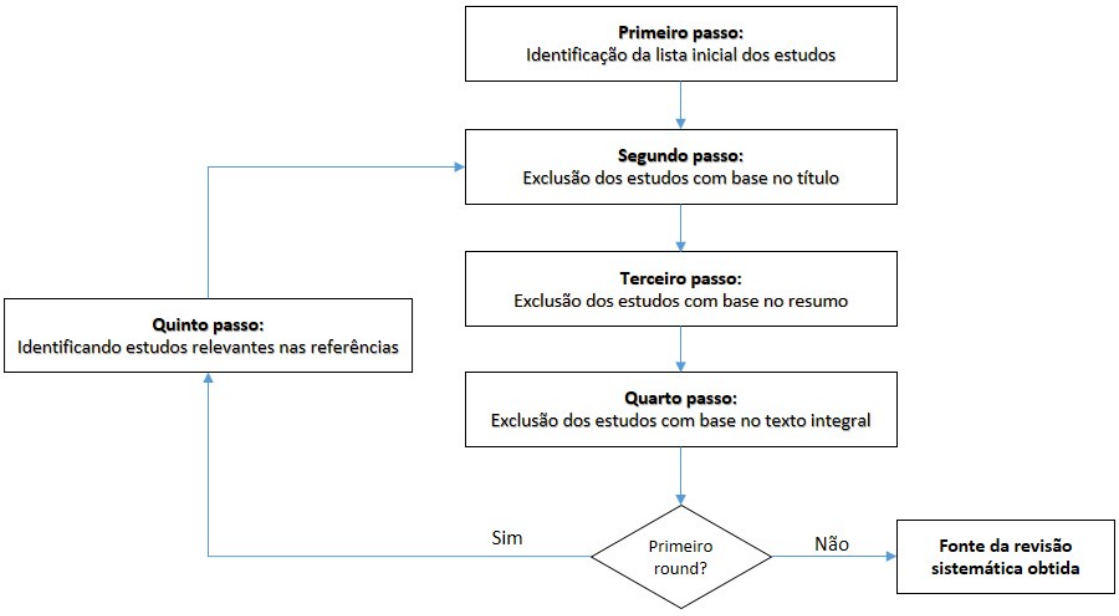


Figura 1. Processo de seleção e avaliação das publicações

Todo o processo foi executado por dois pesquisadores, sendo a pesquisa e seleção das publicações realizadas pelo primeiro e a validação dos resultados pelo segundo. Uma análise comparativa entre as publicações, no que diz respeito à qualidade

dos trabalhos, não foi realizada pelos autores e pode configurar uma limitação à validade dos resultados.

4. Resultados

O processo de pesquisa resultou em 20 estudos preliminares, escritos por 35 autores, ligados a 22 instituições distintas, espalhadas por 7 países, em 4 continentes. A maioria dos estudos foi realizada por universidades dos Estados Unidos e Singapura. Com exceção de duas publicações datadas da década de 90, ambas largamente referenciadas por vários dos estudos selecionados na busca inicial, todos os demais trabalhos foram publicados entre os anos de 2000 e 2015. Os temas mais abordados nas publicações selecionadas foram: Empregabilidade (8), auto-gestão do conhecimento (9), obsolescência profissional (14), desafios do profissional de TI (17), antecedentes para a obsolescência (4) e ações para mitigar o risco da obsolescência (11).

4.1. Sobre o conceito de obsolescência

A obsolescência é o estado em que os profissionais já não têm o conhecimento ou habilidades necessárias para desempenhar bem suas atividades atuais ou futuras; é a distância entre as habilidades do profissional e as exigências do mercado [3,8-12,17-19]. Na área de tecnologia da informação, este é um risco em potencial, considerando a velocidade com que as ferramentas, plataformas e metodologias de trabalho evoluem e o impacto que elas exercem na empregabilidade dos indivíduos [2,5,14,15,19].

4.2. O que antecede a obsolescência

Existem três fatores atuantes na obsolescência do profissional [19]: fatores pessoais, que dizem respeito ao indivíduo (idade, personalidade, aptidão, etc.); fatores relacionados com o trabalho, como a complexidade e desafios enfrentados pelo profissional; e por fim, fatores organizacionais, como por exemplo a variedade das atividades exercidas, ambiente inovador e cultura de aprendizado contínuo.

Fatores individuais: A atualização contínua consome tempo e energia [2,14]. A não ser que o indivíduo veja nisso algo prazeroso ou extremamente necessário, cedo ou tarde a negligenciará. A auto eficácia, estágio da vida profissional e compromissos pessoais determinarão para mais ou para menos a inclinação para a obsolescência [4,7].

Fatores associados ao trabalho: O indivíduo está muito mais familiarizado com a realidade da instituição onde atua do que propriamente com a tecnologia, plataforma ou metodologias que conhece [2,18]. Ele dificilmente perceberá a necessidade de aprender coisas que não estejam diretamente ligadas com o seu dia-a-dia [13,15,16]. Logo, a complexidade e os desafios diários serão igualmente determinantes para obsolescência profissional.

Fatores organizacionais: Por questões competitivas, algumas organizações promovem um clima de incentivo ao aprendizado contínuo, como forma de ter o seu corpo funcional em sintonia com o mercado e preparado para assumir diferentes papeis dentro da instituição [11,19]. Profissionais que atuam em empresas contrárias a essa filosofia estão mais expostos ao risco se tornarem defasados.

4.3. Lidando com o risco da obsolescência

É uma ironia pensar que justamente os profissionais da área de TI, responsáveis por promover constantes inovações tecnológicas, encontrem-se imensamente vulneráveis por estas mudanças [1,4,5,14]. Alguns autores apontam as ações preventivas ou reativas ao risco da obsolescência em quatro categorias [8,17-19]:

Resolução do problema: Deliberar esforços com foco no problema a ser resolvido. Esta categoria prevê quatro ações distintas: Especializar-se em algo (conhecer muito sobre um pouco); estar familiarizado com uma vasta lista de tecnologias e plataformas (saber um pouco sobre muita coisa); manter-se a par do mercado (acompanhar o mercado e suas tendências); e fazer ciclos de duas fases (aprendizado e trabalho), tratando-os como coisas distintas.

Redefinição da situação: Também dividido em quatro grupos, tenta avaliar o problema de um ponto de vista que o faça parecer mais tolerável: Tirar o foco de si e colocar no grupo (a responsabilidade de estar atualizado seria do grupo); estamos todos no mesmo bote (a obsolescência é um problema do grupo); bolhas temporariamente imutáveis (o domínio atual já é o suficiente para hoje - amanhã será um novo dia); e evolução gradual (a expansão do conhecimento é lenta e incremental).

Negação ou distanciamento: Basicamente negar a existência do problema, baseando-se na experiência do profissional (estar atualizado é acumular experiência) e que estar atualizado não é um ingrediente fundamental para o sucesso profissional.

Aprender por lazer: Criar um laço positivo com o ato de aprender algo novo, relacionando-o com recompensas prazerosas ou fazer do aprender algo prazeroso.

4.4. Lidando com a obsolescência

Alguns autores relacionam a obsolescência com a intenção de mudar de emprego ou mudar de profissão [5,9,10,17,20] e afirmam que uma vez obsoleto, resta ao indivíduo se envolver em áreas onde a pressão por renovação seja cada vez menor ou arcar com a perda do grau empregável [8-10]. O grau de comprometimento e a satisfação com a carreira enquanto profissional de TI determinarão se o indivíduo estará disposto a recomeçar em outra profissão, onde a sua base de conhecimento e habilidades sejam mais estáveis e sofram menos pressão da indústria, ou se enfrentará a perda do grau empregável e buscará a renovação da sua base de conhecimentos [1,3,6,8-11,17,20].

5. Discussões

Os frequentes avanços tecnológicos exigem cada vez mais que profissionais de tecnologia da informação busquem o aprendizado contínuo [1,2,4-6,8-11]. Não conseguir manter um bom desempenho profissional, por questões técnicas, é um dos principais receios dos indivíduos que atuam com TI [5,16,19]. Isso porque o tempo de vida estimado de uma publicação de teor tecnológico é de aproximadamente cinco anos [3]. Já o tempo estimado para que a base de conhecimentos de um profissional de TI se torne obsoleta é de apenas dois [5-11,16-20].

A obsolescência pode ser dividida em estado da arte, que diz respeito a estudos e pesquisas acadêmicas, e estado da prática, que se refere ao que a indústria já absorveu e pratica [2,3,15,16]. Como o natural é estar familiarizado apenas com o universo da

empresa onde atua [2,18], é possível ver profissionais mudarem de organização para absorver conhecimento em outros locais ao invés de acompanhar a academia [4,5,9,10].

A literatura aponta três antecedentes para a obsolescência profissional: fatores individuais (idade, aptidão, personalidade, motivação), fatores relacionados ao trabalho (complexidade, desafios) e fatores organizacionais (variedade de distribuição de trabalho). Tais elementos se confundem com fatores empregáveis[Jefferson 2006; Miller & Dettori 2008; Swigon 2011]. Isso porque pode-se afirmar que a obsolescência antecede a perda de grau empregável [8-10,12], levando o indivíduo a buscar trabalhos compatíveis com as suas habilidade e conhecimentos, dentro ou fora da sua área de atuação original, dentro ou fora da organização onde está empregado [10,11,18].

Alguns autores observam o fenômeno com uma visão simplista, atribuindo à defasagem profissional a falta de uma leitura constante sobre assuntos relacionados ao mercado [2,13,16] ou apontando como solução para a rotina intensa e desgastante, a utilização de mídias com conteúdo prático, como por exemplo *podcasts* [14]. Contudo a quantidade de tecnologias emergentes, linguagens de programação e técnicas aplicadas a processos ou formas de construção é imensa [15].

No âmbito educacional, Miller & Dettori (2008) indicam que programas de formação técnica desenvolvam habilidades que não estejam diretamente ligadas a uma tecnologia ou plataforma. Eles esperam que tal conhecimento se dissolva numa velocidade mais lenta, mesmo com as variações tecnológicas, comumente observadas na área de TI como um todo, justamente por estarem desconectadas a elas. Swingon (2011) por sua vez aponta a importância de aprender a aprender e coloca o processo de gestão do conhecimento pessoal como um dos pilares da gestão e suporte à vida profissional, indicando inclusive o comprometimento com a manutenção da própria base de conhecimentos como um fator natural de mitigação de risco de obsolescência técnica ou profissional.

No âmbito profissional, como forma de evitar a obsolescência técnica, pode-se buscar atribuir a responsabilidade de estar atualizado ao grupo de trabalho, acompanhando as tendências do mercado e mantendo uma evolução gradual [19]. Desse modo, espera-se reduzir a carga de stress e pressão exercida sobre o indivíduo, comuns na área de TI [Joseph 2001; Joseph et al. 2011; Rajeswari & Anantharaman 2003; Setor et al. 2015]. Não é recomendada a negação do fenômeno ou puramente a avaliação simplista sobre a sua empregabilidade, dando à experiência profissional um peso maior do que de fato exerce no mercado de trabalho [8,19], pois este seria um atalho para alcançar a obsolescência.

Por fim, uma vez obsoleto, o profissional se encontra numa bifurcação. De um lado, a aposta na profissão, onde ele já não consegue mais acompanhar as mudanças impostas pelo mercado, e do outro, atuar em áreas onde o seu conhecimento atual sirva como vantagem competitiva, aceitando que mesmo sendo um usuário avançado (na maior parte das vezes), carregará o peso da inexperiência no campo das suas novas atribuições [4,8,19].

6. Conclusão

O objetivo deste trabalho foi levantar estudos e soluções para o risco da obsolescência profissional, através da identificação dos fatores que levam à defasagem técnica e profissional. Na fase inicial da pesquisa, 438 trabalhos foram encontrados, onde após aplicar o protocolo definido nesta revisão, 20 foram selecionados para os estudos preliminares. As publicações foram classificadas de acordo com a definição dos termos empregabilidade e obsolescência técnica ou profissional, tempo de obsolescência previsto, avanços tecnológicos e desafios do profissional de TI, antecedentes para a obsolescência e ações para mitigar seus riscos ou revertê-la.

Apesar da tentativa de foco na área de engenharia de software, muitos dos fatores identificados não se restringem a esta área, sendo de fato muito mais abrangentes à TI, de uma forma geral.

A partir dos estudos selecionados, entende-se que ações individuais ou coletivas em prol do aprendizado contínuo ainda são a melhor ferramenta para se manter em sintonia com as exigências do mercado. Ações tomadas por instituições para criar e manter um clima organizacional que estimule a constante renovação da base de conhecimentos e variedade da execução das atividades do seu quadro funcional ajudam a reter bons talentos e estimulam que soluções inovadoras e modernas sejam disseminadas no grupo. É importante destacar também a importância do ensino de habilidades profissionais não ligadas diretamente a uma tecnologia ou plataforma, como forma de fazer com que tal conhecimento se dissolva mais lentamente. Os esforços em prol de destacar a importância da manutenção da própria base de conhecimento são igualmente relevantes, considerando que ao ingressar no mercado de trabalho o indivíduo será senhor de si e terá por responsabilidade gerir a própria carreira profissional.

Assumindo a consumação do risco da obsolescência, o indivíduo deverá arcar com os prejuízos relacionados à sua empregabilidade e optar pela reciclagem da sua base de conhecimentos, como forma de recuperar o grau empregável, ou por exercer outra atividade que não exerça tanta pressão pelo aprendizado contínuo e onde suas habilidades atuais o coloquem numa posição ainda diferenciada frente aos seus concorrentes, dentro ou fora da organização onde trabalha, na mesma área ou em funções correlatas.

7. Referências

Alavi, M. and Leidner, D. (2001). Review: Knowledge Management and Knowledge Management Systems : Conceptual Foundations and Research Issues. *MIS Quarterly*, v. 25, n. 15, p. 107–136.

De Grip, Andries. Evaluating human capital obsolescence. Researchcentrum voor Onderwijs en Arbeidsmarkt, *Faculteit der Economische Wetenschappen en Bedrijfskunde*, Universiteit Maastricht, 2006.

Dubin, S. S. (1972). Obsolescence or Lifelong Education: A Choice for the Professional. *American Psychologist*, v. Vol 27(5), p. 486–498.

Fu, J. R. (2011). Understanding career commitment of IT professionals: Perspectives of push-pull-mooring framework and investment model. *International Journal of Information Management*, v. 31, n. 3, p. 278–293.

Ghapanchi, A. H. and Aurum, A. (2011). Antecedents to IT personnel's intentions to leave: A systematic literature review. *Journal of Systems and Software*, v. 84, n. 2, p. 238–249.

Jefferson, T. L. (2006). Taking it personally: personal knowledge management. *VINE*, v. 36, n. 1, p. 35–37.

Joseph, D. (2001). The Threat-Rigidity Model of Professional Obsolescence and Its Impact on Occupational Mobility Behaviors of IT Professionals THE THREAT-RIGIDITY MODEL OF.

Joseph, D., Tan, M. L. and Ang, S. (2011). Is Updating Play or Work? *International Journal of Social and Organizational Dynamics in IT*, v. 1, n. 4, p. 37–47.

Kaufman, H. G. (2014). Obsolescence and Professional Career Development . by H . G . Kaufman Review by : Douglas T . Hall. v. 20, n. 3, p. 474–477.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. *Engineering*, v. 2, p. 1051.

McGrath, S. (2009). What is Employability? *Learning to support employability project*, p. 1–15.

McQuaid, R. W. and Lindsay, C. D. (2005). The concept of employability. *Urban Studies*, v. 42, n. 2, p. 197–219.

Miller, C. S. and Dettori, L. (2008). Employers' perspectives on it learning outcomes. *Proceedings of the 9th ACM SIGITE conference on Information technology education - SIGITE '08*, p. 213.

Rajeswari, K. and Anantharaman, R. (2003). Development of an instrument to measure stress among software professionals: Factor analytic study. *and diversity in the IT workforce*, p. 34–43.

Setor, T., Joseph, D. and Srivastava, S. C. (2015). Professional Obsolescence in IT: The Relationships Between the Threat of Professional Obsolescence, Coping and Psychological Strain. *Proceedings of the 2015 ACM SIGMIS Conference on Computers and People Research*, p. 117–122.

Swigon, M. (2011). Personal Knowledge Management (PKM) and Personal Employability Management (PEM) - Concepts Based on Competences. *Proceedings of the 3rd European Conference on Intellectual Capital*, n. APRIL 2011, p. 432–438.

7.1. Publicações selecionadas

[1] Chilton, M., Hardgrave, B. and Armstrong, D. (2010). Performance and strain levels of it workers engaged in rapidly changing environments: a person-job fit perspective. *ACM SIGMIS Database*, v. 41, n. 1, p. 8–35.

[2] Coulson, C. (2014). ICT skills shortage: a clear and present danger. *iStart technology in business magazine*, n. 48, p. 18–23.

[3] Fu, J. and Chen, J. H. F. (2015). Career commitment of information technology professionals : The investment model perspective. *Information & Management*, v. 52, n. 5, p. 537–549.

[4] Datta, S. (2015). Discovering the Rise and Fall of Software Engineering Ideas from Scholarly Publication Data. *Www 2015*, p. 585–590.

[5] Fu, J. R. (2011). Understanding career commitment of IT professionals: Perspectives of push-pull-mooring framework and investment model. *International Journal of Information Management*, v. 31, n. 3, p. 278–293.

[6] Ghapanchi, A. H. and Aurum, A. (2011). Antecedents to IT personnel's intentions to leave: A systematic literature review. *Journal of Systems and Software*, v. 84, n. 2, p. 238–249.

[7] Glass, R. L. (2000). On Personal Technical Obsolescence. *Communications of the ACM*, v. 43, n. 7, p. 15–17.

[8] It, E. (1992). Threat of Professional Obsolescence: How Do Professionals at Different Career. v. 29, n. 3, p. 251–269.

[9] Joseph, D. (2001). The Threat-Rigidity Model of Professional Obsolescence and Its Impact on Occupational Mobility Behaviors of IT Professionals THE THREAT-RIGIDITY MODEL OF.

[10] Joseph, D. (2010). Threat of Professional Obsolescence and Mobility Intentions : The Mediating Role of Coping Strategies Threat of Professional Obsolescence and Mobility Intentions : The Mediating Role of Coping Strategies.

[11] Joseph, D., Siew, C. and Koh, K. (2011). Organization Support as a Moderator in Coping with the Threat of Professional Obsolescence Organization Support as a Moderator in Coping with the Threat of Professional Obsolescence.

[12] Kaufman, H. G. (2014). Obsolescence and Professional Career Development . by H . G . Kaufman Review by : Douglas T . Hall. v. 20, n. 3, p. 474–477.

[13] Olav, F. and Dingsøyr, T. (2008). Knowledge management in software engineering : A systematic review of studied concepts , findings and research methods used.

[14] Purvis, J. R. (2006). Tip the work-life balance back in your favor with a podcast. *IEEE International Engineering Management Conference*, p. 24–27.

[15] Roberts, E. (2004). The dream of a common language. *Proceedings of the 35th SIGCSE technical symposium on Computer science education - SIGCSE '04*, v. 36, n. 1, p. 115.

[16] Rong, G. and Grover, V. (2009). Keeping up-to-date with information technology : Testing a model of technological knowledge renewal effectiveness for IT professionals. v. 46, p. 376–387.

[17] Setor, T., Joseph, D. and Srivastava, S. C. (2015). Professional Obsolescence in IT: The Relationships Between the Threat of Professional Obsolescence, Coping and Psychological Strain. *Proceedings of the 2015 ACM SIGMIS Conference on Computers and People Research*, p. 117–122.

[18] Siew, C., Koh, K., Chi, A. and Foo, H. (2010). Sustainable it-specific human capital: Coping with the threat of professional obsolescence.

[19] Tsai, H. P., Compeau, D. and Haggerty, N. (2004). A Cognitive View of How IT Professionals Update Their Technical Skills. *SIGMIS'04 acm*, p. 70–73.

[20] Zhang, X., Ryan, S. D., Prybutok, V. R. and Kappelman, L. (2012). Perceived Obsolescence, Organizational and Turnover of IT Workers: An Empirical Study. *ACM SIGMIS Database*, v. 43, n. 4, p. 12–32.

Scrum in Practice: Evaluating an Activity to Support Agile Software Development Learning

Cleiton Tavares[1], **Fischer Ferreira**[1,2], **Eduardo Fernandes**[2], **Johnatan Oliveira**[2]

[1]University of Itaúna (UIT)
Itaúna – MG – Brazil

[2]Software Engineering Laboratory (LabSoft) – Department of Computer Science
Federal University of Minas Gerais (UFMG)
Belo Horizonte – MG – Brazil

cleitonsilvatavares@gmail.com,
{fischerjf, eduardofernandes, johnatan.si}@dcc.ufmg.br

Abstract. *Practical activities have been largely used to support learning of subjects related to Software Engineering, such as software development and project management. However, we lack a study to assess the effectiveness of practical activities in the improvement of students' performance to learn the Scrum agile method. In this paper, we present a one-and-a-half hour empirical study with 77 undergraduate students from 2 institutions. We divided participants in controller and experiment group. The controller group engaged in a face-to-face class on Scrum with support of slide presentation. The experiment group engaged in a practical activity that exercises Scrum concepts. After, we applied a questionnaire to assess whether students' performance is improved by participating in a practical activity when compared with face-to-face class. As a result, participants from the experiment group obtained a mean of 73% of correctness in the questionnaire, against 78.9% from the controlled group. In this context, our data suggests that practical activities may support face-to-face classes without replacing it. With respect to participant feedback, up to 84.8% of participants point that practical activity is the most interesting type of class.*

1. Introduction

Software Engineering is one of the foundations of Computer Science and related courses [ACM/IEEE 2013, Braude and Bernstein 2016]. It encompasses several subjects that have applications in both academia and industry, such as software development and maintenance, software quality, and project management [Laird 2016]. With respect to the applications of Software Engineering in industrial settings, it is important that educators are able to provide sufficient knowledge and experience that support the formation of practitioners [Scott et al. 2016]. Therefore, the learning process may focus not exclusively in theoretical basis, but also in application of concepts in classroom through practical activities [Rouvrais et al. 2010], for instance.

Many studies propose the use of different practical activities to support learning of Software Engineering subjects [Battistella and von Wangenheim 2016, Braude and Bernstein 2016, Laird and Yang 2016]. In addition, some studies investigate

the effectiveness of applying practical activities to improve students' performance in terms of score grades [Hwang et al. 2012, Rodriguez et al. 2015]. However, to the best of our knowledge, we did not find a study to assess specifically the effectiveness of practical activities in the context of agile methods with focus on Scrum. Such a study may be useful because Scrum is of the most largely adopted agile method in the industrial context [Yang et al. 2016]. Therefore, to instruct students on the subject, in a practical way, may improve the formation of practitioners. We also applied a feedback form to assess the subjective opinion of participants regarding face-to-face class and practical activities.

In this paper, we present a one-and-a-half hour empirical study with 77 undergraduate students from 2 superior education institutions: University of Itaúna (UIT) and School of Pará de Minas (FAPAM). We divided participants in a controller group and an experiment group. The controller group engaged in a face-to-face class on Scrum with support of slide presentation. The experiment group engaged in a practical activity that exercises Scrum concepts. After, we applied a study questionnaire with 10 questions on Scrum. We aimed to assess whether students' performance is improved by participating in practical activities when compared with face-to-face class, based on the percentage of correct answers in the questionnaire.

As a result, participants from the practical activity obtained a mean of 73% of correctness in the questionnaire, against 78.9% from the face-to-face class. In this context, our data suggests that practical activities may effectively support face-to-face classes without replacing it. With respect to participant feedback, up to 84.8% of participants point that practical activity is the most interesting type of class when compared with traditional class (i.e., oral exposition) and slide presentation. Therefore, we concluded that, ion addition to its effectiveness, practical activities motivate the students' learning.

The remainder of this paper is organized as follows. Section 2 provides background information the support the comprehension of the study. Section 3 describes the design of an empirical study that compares students' performance in face-to-face class with practical activity in class. Section 4 presents the study results. Section 5 discusses related work. Section 6 presents some threats to the study validity. Section 7 concludes the paper and suggests future work.

2. Background

This section provides background information to support the comprehension of our study. Section 2.1 presents agile software development and provides some examples of techniques. Section 2.2 discusses the need of practical approaches to support the formation of software engineers. Section 2.3 presents the Scrum agile method.

2.1. Agile Software Development

The agile software development is a set of techniques to support development and management of software systems in a fast and practical way [Cohen et al. 2004]. It has been proposed as a new development approach, that is less strict and more efficient than traditional, sequential models [Cohen et al. 2004, Fowler and Highsmith 2001]. The principles of agile development are compiled in The Agile Manifesto[1]. These principles cover,

[1] agilemanifesto.org/principles.html

basically, four key-statements: individuals instead of processes and tools, functional software over extensive documentation, collaboration between stakeholders and development team, and responding to change.

Many agile methods have been proposed in the literature [Beck 2000, Janzen and Saiedian 2008, Mellor et al. 2003], such as Extreme Programming to guide software development with sufficient product quality, fast delivery, and team collaboration [Beck 2000], Model-Driven Development to support software reuse based on abstract models of a software system [Mellor et al. 2003], and Test-Driven Development to support highly-tested and reliable software systems [Janzen and Saiedian 2008]. In general, the agile methods focus on acceptable software quality [Fowler and Highsmith 2001], sufficient documentation [Cockburn 2004], and short time-to-market [Ambler and Lines 2012].

2.2. Software Engineer Formation

In order to provide formation of software engineers that are appropriately prepared to the industry of software development, courses needs to develop some key skills of the student. Such skills encompass ability to work in group, i.e., collaboration with other developers [Giraldo et al. 2010], creativity to solve problems [Highsmith and Cockburn 2001], ability in the abstraction of problems [Pressman 2005], and communication with the team [Pressman 2005]. To support the development of these skills in students, Software Engineering courses covers several technical subjects such as programming, software modeling, requirement elicitation, and software processes.

However, classes with exclusive focus on technical and theoretical aspects may not be sufficient to provide the required improvement of skills for students [Giraldo et al. 2010]. In this context, practical, ludic activities may be an effective pedagogical strategy to improve students' skills that are not covered by theoretical learning [Shaw 2000]. That is, by using practical activities in class, professors may stimulate students with respect to relevant aspects of software engineering, that require practical experience to be developed.

2.3. The Scrum Agile Method

Scrum is one of the most applied agile methods in industrial settings [Cohen et al. 2004]. It consists of a technique to support software management and project planning, for instance. Many studies have been investigating the use of Scrum to support organizations [Bellomo et al. 2013, Jie 2016, Maximini 2015] as effective for the intended purpose. Moreover, it is interesting to teach Scrum in academia because this technique is largely applied in industry. Therefore, Scrum is important to the formation of practitioners, for instance [Jalali and Wohlin 2010, Rodriguez et al. 2015].

By using Scrum, software development projects are divided in well-defined cycles called *sprints*. In general, each sprint has a duration between two weeks and a month. In Scrum, daily meetings (called *daily Scrum*) are promoted to support tracking of issues and development of tasks. One of the main goals of daily meetings is to provide communication among development team members [Cohen et al. 2004]. Scrum prescribes a framework for the daily meetings composed of three basic questions that cover (i) what did the developer do in the previous working day, (ii) what do the devel-

oper intend for the current working day, and (iii) what difficulties hinder the developer to reach its goals [Cohen et al. 2004].

3. Experiment Design

This section presents the design of an empirical study. Section 3.1 discusses the study goal and research questions that support our work. Section 3.2 presents the artifacts to support the study execution. Section 3.3 presents the steps of the empirical study. Section 3.4 provides an overview of the participant set.

3.1. Goal and Research Questions

In this study, we aim to assess the effectiveness of applying a practical activity in class to support learning of the Scrum agile method. For this purpose, we compare students' performance when engaged in a face-to-face class with a practical activity in class to provide practical learning. This comparison is provided by the analysis of a questionnaire that covers Scrum main concepts, such as definition and application in software development.

The study goal, based on the Goal-Question-Metric (GQM) method [Wohlin et al. 2012], is: *Analyze a practical activity to support Scrum agile method learning, from the purpose of students' performance evaluation through a questionnaire on Scrum when compared with face-to-face class, with respect to the percentage of correct answers in the questionnaire, from the point of view of Software Engineering students, in the context of Brazilian undergraduate students*.

To support the proposed study, we designed the following research questions.

RQ1. Did participants from the practical activity present better performance in questionnaire when compared with participants from face-to-face classes?
RQ2. Did participants from the practical activity agree the experiment may support Scrum learning?

To investigate both RQ1 and RQ2, we conducted an empirical study to compare two types of class: face-to-face class with support of slide presentation and practical activity. In this context, participants of the study are divided in two groups for comparison: (i) a *controller group*, composed by participants of a face-to-face class and (ii) an *experiment group*, composed by participants of a practical activity selected by the authors with support of students. We discuss the selection of our practical activity for analysis as follows. We also discuss a pilot-study for refinement of the study design.

Selection of the practical activity. To select the practical activity for assessment in this study, we selected a Software Engineering class with 12 undergraduate students from UIT. These students had previously studied concepts related to agile methods and Scrum. We ask participants to search on the Web and in the literature for practical activities that encompass the Scrum subject. Such activities could be educational games (based on cards, boards, or simulation, for instance) or any activity applicable to classrooms in a one-and-a-half hour duration.

After collecting proposals of practical activities for the study, we conducted a seminar to discuss each proposal and chose the most appropriate one. Among the various

practical activities from the students, we selected a paper airplane production line. This activity, collected from the Web[2], was considered feasible to provide sufficient knowledge on Scrum to participants. Therefore, we expected that the application of the chosen activity is easy and appropriate to be compared with face-to-face classes.

Pilot-study. To evaluate the feasibility of the selected practical activity in the academic context, we performed a pilot-study. All the 12 students involved in the selection of the activity participated in this process. Finally, we concluded that the selected activity is feasible and motivating to be applied in our empirical study.

3.2. Experiment Artifacts

We designed the following artifacts to support the execution of our empirical study.

- *Commitment Form*: this formal document consists of an authorization for us to collect, analyze, and anonymously publish experiment data,
- *Experiment Questionnaire*: this document contains 10 questions to be answered during the study execution. All questions from the questionnaire are specific to the subject of Scrum agile method. Data collected from the Study Questionnaire provided us means to assess students' performance regarding the percentage of correct answers for both controller and experiment group, and
- *Feedback Form*: this document consists of 4 questions to assess the subjective perception of participants with respect to the performed experiment (face-to-face class or practical activity).

3.3. Experiment Steps

The steps of our empirical study are different according to the group in which a participant is allocated. We discuss the steps for each group as follows.

Controller group: face-to-face class. In the controller group, participants have engaged in a face-to-face class conducted by the same course instructor with support of slide presentation regarding the Scrum agile method. In a one-and-a-half hour class, the instructor presented the main Scrum concepts. The slide presentation encompasses agile methods, definition of Scrum, main Scrum phases, software development supported by Scrum, and the relationship between developers and stakeholders.

Experiment group: practical activity. In the experiment group, participants have engaged in a practical activity in class. The practical activity performed by the students consists of producing a paper airplane by applying Scrum concepts. To produce the airplane, participants have to perform the activity in a product line approach.

We discuss the steps of the activity as follows. First, each class was divided in teams with from 5 to 9 participants. A team is independent from the others and responsible for its decision-making and management. Second, teams are asked to produce a paper airplane in a product line approach. That is, the plane production is started by a participant

[2]http://agileway.com.br/2009/08/18/dinamica-fabrica-de-avioes-2-0/#more-351 (in portuguese)

and finished by other participant in a sequential fashion. In addition, each team has only 3 sprints with 3 minutes length, plus 3 minutes to estimate the airplane production, as in a *daily Scrum*, and complete the activity.

Other details of the activity are explained as follows. The paper airplane produced by the teams has to follow a well-defined specification. Each airplane has to contain 12 windows, a cabin, a corporation logo, and it must to be able to thrown in the air and fly. With respect to the roles of participants in the activity, per team we have (i) a *Scrum master* to motivate the achievement of goals and handle with problems during the airplane production, (ii) a *weak link*, i.e., a participant that hinders the satisfactory production of the airplane, and (iii) *team members* that actually produce the airplane and report eventual issues to the Scrum master. Team planning has to be documented in a spreadsheet. In the end of the activity, the winner team is chosen by the instructor in accordance to the airplane requirements. Finally, recap on the activity is provided with lessons learned.

3.4. Participant Set

Table 1 presents the distribution of participants in the study groups per educational institution, i.e., FAPAM and UIT. For more details, including the artifacts of the study described in Section 3.2, consult the research website[3].

Table 1. Distribution of participants in groups per institution

Institution	Face-to-Face Class	Practical Activity	Sum
FAPAM	16	16	32
UIT	28	17	45
Total	**44**	**33**	**77**

4. Results and Discussion

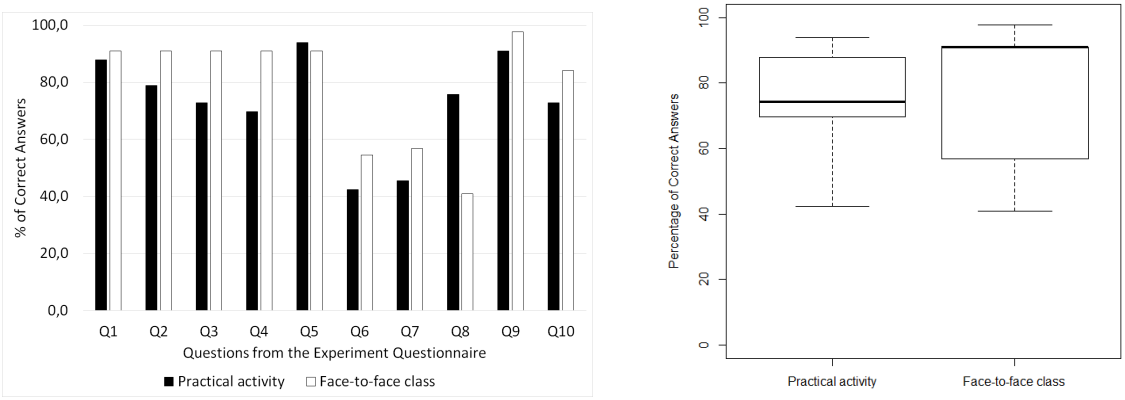
This section presents and discusses the results we obtained through our empirical study with 77 undergraduate students. Section 4.1 presents the analysis of results from the Study Questionnaire. Section 4.2 presents the analysis of participant feedback with respect to the conduction of face-to-face class and the practical activity. Section 4.3 discusses the main lessons learned.

4.1. Analysis of Questionnaires

Figure 1(a) presents the percentage of correct answers with respect to the Study Questionnaire. Q1 to Q10 refer to the 10 questions provided to the participants. In this figure, we compare both experiment group, i.e., students that participated in the practical activity, and controller group, i.e. students that engaged in the face-to-face class. Note that participants from the practical activity presented higher percentage of correct answers for only 2 of the 10 questions (20% of them), namely, Q5 and Q8. This result was expected, since our practical activity does not aim to exercise terminology and dense theoretical aspects of Scrum, but to provide a practical overview on the subject. For instance, we may expect a student that performed the Scrum Master role to present a more solid learning regarding the performed role than other Scrum roles.

[3]http://goo.gl/Zh9IF1

Figure 1(b) presents the distribution of the percentage of correct answers for the two groups. Table 2 provides the respective descriptive analysis. Note that Q3 for both groups are close (with a slight different around 5%), as Q1 (around 2%), suggesting that the distribution of values for the groups is similar. Moreover, the mean percentage for students of practical activities and face-to-face class present a minimum difference around 6%. Note that the practical activity does not cover theoretical aspects of Scrum as in the face-to-face class. Therefore, there is an evidence that the practical activity provided sufficient knowledge for students to answer questions regarding Scrum.



(a) Percentage of correct answers per question of the Study Questionnaire

(b) Distribution of the percentage of correct answers per group

Figure 1. Correct answers for the Study Questionnaire

Table 2. Descriptive analysis of Figure 1(b)

Group	Q1	Median	Q3	Mean
Practical activity	70.45%	74.25%	85.625%	73%
Face-to-face class	63.625%	90.9%	90.9%	78.9%

4.2. Analysis of Participant Feedback

Figure 2 presents the percentage of participants that prefer each of the following types of class: traditional (i.e., classes based on oral exposition only), practical activity, and slide presentation. For this analysis, we separated the concepts of traditional class, that encompasses oral exposition, from slide presentation, that may be also conducted with oral exposition. We provided this separation for students to report when they do prefer oral exposition class without the support of slide presentation. Note that most of the participants prefer practical activities. We may assume that the main reason for that is students tend to be motivated by innovation, dynamism, and entertainment in classroom.

Since the practical activity presented sufficient positive impact on the students' performance (see Section 4.1, and the participants reported a positive feedback on the its application in class, we have an evidence that practical activities may support face-to-face agile method classes without replacing it.

4.3. Lessons Learned

After conducting our empirical study, we have observed some informal feedback from the participants discussed as follows.

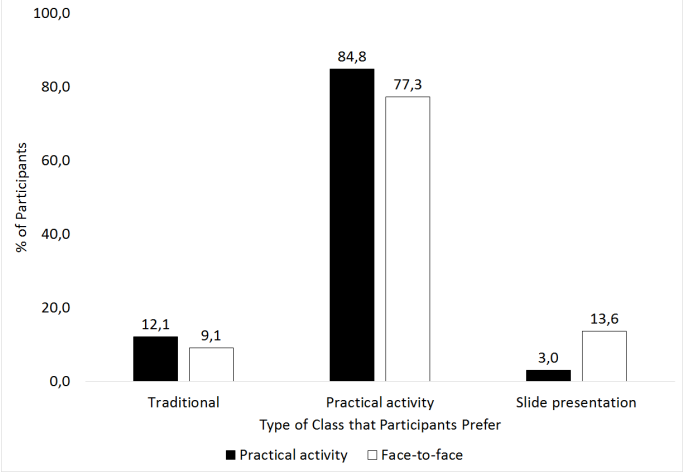


Figure 2. Percentage of type of class that participants prefer

- In general, participants reported a positive feedback with respect to the practical activity. However, some students reported that, from their viewpoint, this type of activity may not replace face-to-face classes,
- Students presented significant motivation to perform the practical activity. For instance, many of them followed properly the roles established by the Scrum agile method. In such cases, participants presented better results in the questionnaire, from the viewpoint of the authors,
- Among the students that performed the pilot-study (see Section 3.1), some of them presented interest in deeper knowledge regarding Scrum and its applications, and
- Some students with professional experience reported a growing interested in applying Scrum in the industrial settings they are inserted.

Therefore, we conclude that the application of our practical activity with students provided a significant positive impact in classroom, with respect to pedagogical aspects that may support the Scrum agile method learning.

5. Related Work

Oliveira et al. (2013) present a systematic mapping study [Petersen et al. 2008] on studies that report experiments regarding the use of learning approaches based on real-life situations. The authors observe that, although Computing courses are massively theoretical, the learning approaches with real application appeal have been successfully applied in education. These results are evidenced for Software Engineering and Programming subjects. The study presents student and professor reports on learning improvements related to skills for solving practical problems. It is discussed that professors may effectively simulate real, industrial problems in class.

Zorzo et al. (2013) investigate an approach that applies Scrum concepts and principles in a Web development specialization course. In the proposed approach, the course structure is designed based on Scrum main aspects and activities, such as roles and sprints. The study motivation is to provide collaboration and parallel execution of the course activities inspired by the philosophy of agile methods. The authors guided students to perform the activities of the course. They observe some drawbacks of the proposed approach, such as the difficulty to distribute subjects along the course by using sprints.

In this study, we selected a simple and motivating practical activity for assessment: the production of paper airplanes inspired by Scrum. The selected activity covers many aspects of the Scrum agile method, such as division of teams, meetings, and management of tasks. The main difference of our study when compared to previous work is that we chose a shorter practical activity that may be performed in a one-and-a-half hour duration class by undergraduate students.

6. Threats to Validity

We designed and conducted carefully our empirical study as described in Section 3. As an example, we designed our study before its execution. Moreover, we defined the research questions and ways to assess them based on previous work. However, some threats may invalidate our research findings. In this context, we discuss each of the four types of threats presented by Wohlin et al. (2012), with respective treatments, as follows: internal, conclusion, construct, and external validity.

Construct Validity. We designed our study to be applied in different education institutions without significant adaptations. We conducted a pilot-study with volunteers to define development tasks that could be adequately performed by students considering time and space constraints in the academic context. To provide impartiality of participants with respect to our research questions and experiment hypotheses, we omitted this information to all participants. Therefore, we expected an experiment execution with minimization of biases. Finally, we proposed a baseline for assessment of correctness to the study questionnaire.

We carefully selected participants that did not engage in agile method courses before study execution. All participants of our empirical study are undergraduate students in the primary stages of Computer Science and related courses. In this context, we assume that participants have similar background and, therefore, we do not consider background analysis in this study. Finally, our participant set is not optimally balanced in terms of the number of participants. We minimize this issue by computing the percentage of correct answers relative to the number of participants in each group.

Internal Validity. We carefully collected data from participants through printed forms that were analyzed and converted to electronic spreadsheets. However, one of the issues that may affect the data collection is the subjective comprehension of participants with respect to (i) the questions in the Study Questionnaire and (ii) the design of the practical activity, in case of participants from the experiment group. To minimize this issue, we (i) trained participants with respect to form and questionnaire filling and (ii) carefully designed the practical activity to be motivating and easy to understand and participate.

In addition, the authors controlled the empirical study execution in terms of time and space. We also provided support to participants during the study execution in case of doubts. We instructed all participants and assisted them in every step of the study to assure the success of the study. Through the aforementioned treatments, we expect the execution of our study was appropriately performed and, then, data collected from the study is reliable for analysis. Finally, our data collection may have been affected

by problems in the design of the Study Questionnaire, with respect to the analysis of the practical activity. In this context, we designed simple questions without excessive technical terms, for instance.

Conclusion Validity. We conducted a careful data analysis to draw meaningful conclusions regarding our empirical study. As an example, we based our data analysis, and also the descriptive analysis to be applied in the study, on previous work. In addition, we double-checked the analyzed data to avoid missing data, incorrect discard of participants, and other factors that prevent a correct data analysis. Finally, we provide all study artifacts, including spreadsheets with the collected data and data analysis, in the research website (see Section 3.4 for more details).

External Validity. There are some factors that prevent the generalization of our findings to any academic context. As an example, the 77 participants of our study are undergraduate students from only 2 Brazilian educational institutions. Therefore, they may not represent appropriately all Brazilian students because of different background, cultural aspects, and professional experience, for instance. To minimize this issue, we conducted our study in different classes from the 2 institutions (see Section 3.4 for details).

Another issue is the time constraints of our empirical study (a one-and-a-half hour study execution). In this context, we designed (i) a succinct, sufficient slide presentation to be used in the face-to-face class, from the authors' viewpoint, and (ii) a practical activity that fits the time constraints with sufficient exercising of the Scrum main concepts, such as roles of developers, sprints, and so on. Therefore, we expect that our study has appropriate length to provide the assessment of our research questions defined in Section 3.1.

7. Conclusion

In this paper, we assess the effectiveness of practical activities in learning the Scrum agile method. For this purpose, we discuss a one-and-a-half hour empirical study with 77 undergraduate students from 2 institutions: FAPAM and UIT. We divided participants in a controller and an experiment group. The controller group engaged in a face-to-face class on Scrum with support of slide presentation. The experiment group engaged in a practical activity that exercises Scrum concepts. After, we applied a study questionnaire with 10 questions on Scrum to participants of both groups. We aimed to assess whether students' performance is improved by participating in practical activities when compared with face-to-face class, based on the percentage of correct answers in the questionnaire.

As a result, participants from the practical activity obtained a mean of 73% of correctness in the questionnaire, a slightly smaller percentage when compared to the 78.9% for participants from the face-to-face class. We observe, then, that practical activities may effectively support face-to-face classes without replacing it. In turn, with respect to the feedback of participants, up to 84.8% of students point that practical activity is the most interesting type of class when compared with traditional class and face-to face class with support of slide presentation. Therefore, we concluded that practical activities also motivate the students' learning.

As future work, we suggest the replication of our study in other educational institutions than FAPAM and UIT, with more participants. We also suggest the conduction of a practical activity that simulates software development with support of Scrum, by using a real software system. In addition, we suggest the application of practical activities that exercise other agile methods such as Extreme Programming. Finally, we suggest the application of different practical activities in class to support learning of other Software Engineering subjects with evident industrial appeal, such as software maintenance, modeling, and requirement elicitation.

References

ACM/IEEE (2013). *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. ACM. 999133.

Ambler, S. and Lines, M. (2012). *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*. IBM Press.

Battistella, P. E. and von Wangenheim, C. (2016). Games for Teaching Computing in Higher Education: A Systematic Review. *Technology and Engineering Education (ITEE)*, 9(1):8–30.

Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional.

Bellomo, S., Nord, R., and Ozkaya, I. (2013). A Study of Enabling Factors for Rapid Fielding Combined Practices to Balance Speed and Stability. In *Proceedings of the 35th International Conference on Software Engineering (ICSE)*, pages 982–991.

Braude, E. and Bernstein, M. (2016). *Software Engineering: Modern Approaches*. Waveland Press.

Cockburn, A. (2004). *Crystal Clear: A Human-Powered Methodology for Small Teams*. Pearson Education.

Cohen, D., Lindvall, M., and Costa, P. (2004). An Introduction to Agile Methods. *Advances in Computers*, 62:1–66.

Fowler, M. and Highsmith, J. (2001). The Agile Manifesto. *Software Development*, 9(8):28–35.

Giraldo, F. D., Collazos, C. A., Ochoa, S. F., Zapata, S., and de Clunie, G. T. (2010). Teaching Software Engineering from a Collaborative Perspective: Some Latin-American Experiences. In *Proceedings of the 21st International Workshop on Database and Expert Systems Applications (DEXA)*, pages 97–101.

Highsmith, J. and Cockburn, A. (2001). Agile Software Eevelopment: The Business of Innovation. *IEEE Computer*, 34(9):120–127.

Hwang, G.-J., Wu, P.-H., and Chen, C.-C. (2012). An Online Game Approach for Improving Students' Learning Performance in Web-based Problem-Solving Activities. *Computers & Education*, 59(4):1246–1256.

Jalali, S. and Wohlin, C. (2010). Agile Practices in Global Software Engineering: A Systematic Map. In *Proceedings of the 5th International Conference on Global Software Engineering (ICGSE)*, pages 45–54.

Janzen, D. and Saiedian, H. (2008). Does Test-Driven Development Really Improve Software Design Quality? *IEEE Software*, 25(2):77–84.

Jie, J. L. H. (2016). Industrial Case Study of Transition from V-Model into Agile SCRUM in Embedded Software Testing Industries. *Software Engineering Notes*, 41(2):1–3.

Laird, L. (2016). Strengthening the "Engineering" in Software Engineering Education: A Software Engineering Bachelor of Engineering Program for the 21st Century. In *Proceedings of the 29th International Conference on Software Engineering Education and Training (CSEET)*, pages 128–131.

Laird, L. and Yang, Y. (2016). Engaging Software Estimation Education using LEGOs: A Case Study. In *Proceedings of the 38th International Conference on Software Engineering (ICSE)*, pages 511–517.

Maximini, D. (2015). *The Scrum Culture: Introducing Agile Methods in Organizations*. Springer.

Mellor, S., Clark, T., and Futagami, T. (2003). Model-Driven Development: Guest Editors' Introduction. *IEEE Software*, 20(5):14–18.

Oliveira, A. M., Santos, S., and Garcia, V. (2013). PBL in Teaching Computing: An Overview of the Last 15 Years. In *Proceedings of the 43rd Frontiers in Education Conference (FIE)*, pages 267–272.

Petersen, K., Feldt, R., Mujtaba, S., and Mattsson, M. (2008). Systematic Mapping Studies in Software Engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, volume 17.

Pressman, R. (2005). *Software Engineering: A Practitioner's Approach*. Palgrave Macmillan.

Rodriguez, G., Soria, Á., and Campo, M. (2015). Virtual Scrum: A Teaching aid to Introduce Undergraduate Software Engineering Students to Scrum. *Computer Applications in Engineering Education*, 23(1):147–156.

Rouvrais, S., Mallet, J., and Vinouze, B. (2010). A Starter Activity Design Process to Deepen Student's Understanding of Outcome-related Project Learning Objectives. In *Proceedings of the Frontiers in Education Conference (FIE)*, pages T1E–1.

Scott, E., Rodríguez, G., Soria, Á., and Campo, M. (2016). Towards better Scrum Learning using Learning Styles. *Journal of Systems and Software (JSS)*, 111:242–253.

Shaw, M. (2000). Software Engineering Education: A Roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, pages 371–380.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., and Wesslén, A. (2012). *Experimentation in Software Engineering*. Springer Science & Business Media.

Yang, C., Liang, P., and Avgeriou, P. (2016). A Systematic Mapping Study on the Combination of Software Architecture and Agile Development. *Journal of Systems and Software (JSS)*, 111:157–184.

Zorzo, S., Ponte, L., and Lucrédio, D. (2013). Using Scrum to Teach Software Engineering: A Case Study. In *Proceedings of the 43rd Frontiers in Education Conference (FIE)*, pages 455–461.

Soft Skills Required! Uma Análise da Demanda por Competências Não-Técnicas de Profissionais para a Indústria de Software e Serviços

César França[1], Diego Mellet[2]

[1]Departamento de Estatística e Informática da Universidade Federal Rural de Pernambuco (DEINFO/UFRPE) – Recife – PE – Brasil

[2]Faculdade Boa Viagem / Devry – Recife – PE – Brasil

cesar@franssa.com, diegovasquesster@gmail.com

Abstract. *Software development companies look for professionals not only with high technical knowledge, but also with several soft skills. In order to map out the most required soft skills by software companies, we analyzed ads for 420 job opportunities in Porto Digital, in Recife/PE. Our data evidence the high demand for soft skills such as English proficiency, teamwork and proactivity. In a deeper analysis, we detail the specific requirements for four different functional roles. Overall, our results are consistent with previously reported studies, which reinforces even more the importance of some soft skills for the software industry.*

Resumo. *Empresas de software buscam profissionais não apenas com conhecimento técnico mas também com habilidades não-técnicas (conhecidas como soft-skills). Com o objetivo de mapear as* soft skills *mais requeridas pelo mercado, analisamos 420 anúncios de empregos em empresas localizadas no Porto Digital, em Recife/PE. Nossos dados evidenciam a alta demanda por soft skills tais como fluência em língua inglesa, trabalho em equipe e proatividade. Uma análise mais aprofundada revela os requisitos mais demandados para diferentes perfis funcionais. Os resultados consistentes com outros estudos previamente reportados na literatura reforçam ainda mais a importância de alguns destes* soft skills *para a indústria de software.*

1. Introdução

Nos últimos anos, a indústria de software e a academia vêm discutindo estratégias para que o capital humano formado nas faculdades, universidades e cursos técnicos atenda de forma mais efetiva às demandas do mercado, e o Fórum de Educação em Engenharia de Software (FEES) tem sido um relevante espaço de interação para facilitar esta convergência de interesses. Porém, as discussões em torno desta questão são frequentemente centradas em, e muitas vezes limitadas à, aspectos pertencentes puramente à dimensão técnica da formação do capital humano [Portela et al. 2015].

A indústria de software vem se descobrindo como uma atividade que envolve um significativo esforço de interação social e humano, refletido claramente por trabalhos recentes que buscam aprofundar o conhecimento científico sobre aspectos

humanos no trabalho na engenharia de software [Guinan et al. 2008, Dutra e Prikladnicki, 2014]. Sendo assim, o desenvolvimento de habilidades técnicas, apenas, não é suficiente para assegurar o sucesso de um profissional ou otimizar o sucesso dos projetos nos quais eles irão atuar [Ahmed et al. 2012]

O próprio *Rational Unified Process* (RUP) [Kruchten 2003] define seus papéis funcionais em termos não apenas de responsabilidades técnicas no processo de desenvolvimento, mas também de algumas habilidades necessárias para os profissionais desempenharem com sucesso determinados papéis funcionais. Para o papel de Analista de sistemas, por exemplo, o RUP sugere que o profissional deve ter facilidade de expressão e comunicação, facilidade em construir relacionamentos; facilidade de adaptação a mudanças, iniciativa na solução de problemas e desenvolvimento de alternativas criativas, tolerância a pressão, presteza e iniciativa, organização, proatividade e objetividade. O Manifesto Ágil (http://agilemanifesto.org), por sua vez, também demanda uma série de *soft skills* que parecem ser largamente responsáveis pelo sucesso dos projetos neste paradigma, tais como comunicação e adaptabilidade.

No entanto, os cursos de computação brasileiros ainda não trabalham adequadamente *soft skills* na formação dos profissionais. Como é possível notar em uma crítica recente de Nunes [2012], que refere-se à comunidade científica de computação do brasil utilizando o termo "Casa de ferreiro, espeto é de pau" pois, segundo ele:

> *"Na produção de software, os primeiros e mais importantes passos estão na definição de requisitos, nas fases finais na implementação. Entretanto, quando da criação/reforma de um curso de Engenharia de Software (ou outro qualquer), os gestores acadêmicos começam pela grade curricular (implementação) e não pela especificação do profissional."* [Nunes 2015, p.18]

Na mesma edição da Revista da Sociedade Brasileira de Computação, Gimenes [2012] aponta que:

> *"os professores* [de engenharia de software] *são conteudistas, isto implica que não trabalham as soft-skills como comunicação, liderança, resolução de conflitos e dinâmica de grupo. Essas habilidades são imprescindíveis para engenheiros de software."* [Gimenes, 2012, p. 24]

Sendo assim, este artigo tem o intuito de ampliar a discussão sobre as características de profissionais demandados pelo mercado de trabalho, focando em particular nas habilidades não-técnicas (*soft skills*) requeridas pela indústria [Gotel 2009], como um primeiro passo no sentido de orientar a adequada formação de profissionais. S*oft skills* são atributos pessoais que permitem ao indivíduo desempenhar uma boa interação com o mundo ao seu redor, incluindo colegas de trabalho e o próprio trabalho em si [Pereira 2012]. *Soft skills* são consideradas tão ou ainda mais importantes do que habilidades técnicas para o profissional [Schulz 2008, Kumar e Sreehari 2011].

Para tanto, conduzimos um levantamento com o objetivo de responder a seguinte questão: "Quais são os *soft skills* mais demandados por empresas de tecnologia da informação localizadas no Porto Digital em Recife?". O Porto Digital foi escolhido

pela sua relevância no cenário da indústria de software nacional, aliado à conveniência dos pesquisadores. Buscando então responder a esta pergunta, seguimos um método similar ao utilizado por Ahmed et al. (2012) e Maturro (2013), e analisamos 420 ofertas de emprego referentes ao período de um ano, a partir das quais conjuntos de *soft skills* foram identificados e contabilizados.

Os resultados apontam para fluência em inglês e habilidade para trabalhar em equipe como os dois principais *soft skills* requeridos pela indústria. Embora nossa pesquisa tenha sido limitada ao ambiente do Porto Digital, os resultados são amplamente consistentes com pesquisas realizadas em outros locais [Ahmed et al. 2012, Mautrro 2013]. Ao longo do texto, aprofundamos ainda a discussão em quatro perfis funcionais específicos: desenvolvedores, analistas de qualidade, designers de software e analistas de requisitos. Esperamos que os *soft skills* relatados sirvam para provocar e guiar a reflexão sobre a eficácia de programas de formação de mão de obra sob o ponto de vista principalmente de disciplinas não-técnicas.

O restante deste trabalho está organizado nas seguintes seções: Seção 2 contém definições básicas dos termos utilizados, a Seção 3 descreve as etapas de levantamento de dados e análise dos resultados, a Seção 4 apresenta os resultados e uma discussão comparativa com estudos relacionados, e finalmente a Seção 5 apresenta considerações finais e diversos direcionamentos para pesquisas futuras.

2. Revisão Literária

Soft skills são conhecidos popularmente como atributos individuais que permitem ao indivíduo manter boas interações com outros, e com o mundo ao seu redor (Pereira, 2012). Estes atributos têm como característica serem capacidades não técnicas ou específicas para um posto de trabalho, sendo úteis portanto em qualquer área profissional. Matturro (2013) define (em tradução livre) que "*soft skills* ou habilidades não técnicas são conjuntos vastos de componentes como habilidades, atitudes, hábitos, boas práticas que quando corretamente combinadas tendem a maximizar o trabalho de uma pessoa".

Na literatura existem estudos que, como este, também identificam quais os *soft skills* mais demandados por empresas de tecnologia da informação para contratação de seus profissionais, por meio da análise de anúncios de emprego publicados em jornais. Fernandez-Sanz (2010), por exemplo, analisou durante oito anos (do início de 2001 até o fim de 2008) mais de 3000 anúncios de emprego relacionados a engenharia de software, no principal jornal de circulação nacional da Espanha, identificando os soft skills mais desejados para os mais diversos cargos e atividades funcionais.

Em 2012, Ahmed et al. (2012) reportaram uma análise de 500 ofertas de emprego para atividade de tecnologia da informação na América do Norte, Europa, Asia e Austrália, e identificaram 9 grupos de *soft skills* com um alto nível de abstração: boa comunicação, habilidades interpessoais, pensamento analítico, trabalho em equipe, habilidades organizacionais, aprendizagem rápida, autogerenciamento, inovação e adaptabilidade. Os dados coletados evidenciaram que boa comunicação se destaca nos quatro perfis funcionais estudados: analistas de sistemas, designers, programadores e testadores. Habilidades interpessoais, trabalho em equipe, pensamento analítico, e habilidades organizacionais foram os outros *soft skill* mais presentes nos seus dados,

como resumidos na Tabela 1. Quando analisados de forma transcultural, as habilidades de comunicação, trabalho em equipe e interpessoais se mantêm no topo da lista.

Tabela 1 – Distribuição de *soft skills* por função no mundo [Ahmed et al. 2012]

Desenvolvedores 28% (140/500)		**Testadores 28% (140/500)**		**Designer de software 21% (105/500)**		**Analista de Sistemas 23% (115/500)**	
Boa comunicação	90% (126/140)	Boa comunicação	79% (110/140)	Boa comunicação	92% (97/105)	Boa comunicação	92% (106/115)
Habilidades interpessoais	65% (91/140)	Pensamento analítico	43% (60/140)	Habilidades interpessoais	92% (97/105)	Pensamento analítico	66% (76/115)
Trabalho em equipe	62% (87/140)	Habilidades organizacionais	37% (52/140)	Trabalho em equipe	49% (51/105)	Habilidades organizacionais	37% (42/115)
Pensamento analítico	52% (72/140)	Trabalho em equipe	27% (38/140)	Pensamento analítico	40% (42/105)	Auto-gerenciamento	27% (31/115)
Habilidades organizacionais	37% (52/140)	Habilidades interpessoais	22% (31/140)	Habilidades organizacionais	16% (17/105)	Habilidades interpessoais	26% (30/115)

Mais recentemente, Matturro (2013), analisou por um período de 15 meses (de Outubro de 2011 até Dezembro de 2012) um total de 678 anúncios de empregos, coletados a partir de um jornal de circulação nacional do Uruguai e a partir da base de dados mantido pela Universidad ORT Uruguay. Destes, 488 ofertas possuíam pelo menos um soft skill requerida, e 17 *soft skills* diferentes foram identificados. A Tabela 2 descreve a distribuição de frequência encontrada por Maturro (2013). Observe que enquanto a língua estrangeira não é mencionada no estudo de Ahmed et. al (2012), este trata-se do principal fator no estudo de Maturro (2013).

Em seu estudo, Maturro sugere não existir nenhuma surpresa com o fato da fluência da língua inglesa ser o *soft skill* mais frequente nos seus dados, uma vez que Uruguai tem o Espanhol como língua principal e a indústria de software trabalha para o mercado externo.

Tabela 2 – Distribuição de *soft skills* por função, no Uruguai [Maturro 2013]

Desenvolvedores 81% (395/488)		**Analista de Qualidade 9% (43/488)**		**Designer de Software 3% (15/488)**		**Analista de Requisitos 7% (35/488)**	
Fluência em Inglês	66% (263/395)	Fluência em Inglês	65% (28/43)	Fluência em Inglês	67% (10/15)	Fluência em Inglês	46% (16/35)
Proatividade	54% (217/395)	Trabalhar em Equipe	63% (27/43)	Boa comunicação	53% (8/15)	Trabalhar em Equipe	37% (13/35)
Trabalhar em Equipe	46% (184/395)	Proatividade	46% (20/43)	Trabalhar em Equipe	40% (6/15)	Proatividade	31% (11/35)
Comprome-timento/ Responsa-bilidade	25% (99/395)	Analítico / Resolução de Pro-blemas	28% (12/43)	Proatividade	33% (5/15)	Comprome-timento/ Responsa-bilidade	26% (9/35)
Vontade de aprender	18% (72/395)	Metódico	23% (10/43)	Vontade de Aprender	20% (3/15)	Relacionamento interpessoal	26% (9/35)

3. Aspectos Metodológicos

Inspirados pelos estudos realizados por Ahmed et al. (2012) e Maturro (2013), decidimos conduzir um levantamento para mapear os *soft skills* mais frequentemente exigidos por empresas de software localizadas no Porto Digital, em Recife – PE. O Porto Digital é um dos principais parques tecnológicos e ambientes de inovação do Brasil. Localizado no Recife, sua atuação se dá nos eixos de software e serviços de Tecnologia da Informação e Comunicação (TIC) e Economia Criativa (EC), com ênfase nos segmentos de games, multimídia, cine-vídeo-animação, música, fotografia e design

[Porto Digital 2015]. Atualmente, o Porto Digital abriga 250 empresas, organizações de fomento e órgãos de Governo e cerca de 7.100 trabalhadores. O Porto Digital foi considerado pela Associação Nacional de Promotoras de Empreendimentos Inovadores (Anprotec), em 2007 e 2011, o melhor parque tecnológico do Brasil. Em 2005 o ambiente foi considerado o maior do País pela A.T. Kearney.

Para identificarmos os *soft skills* mais demandados por empresas de software no âmbito do Porto Digital, consultamos os arquivos do informativo semanal de ofertas de emprego disponibilizados pela assessoria de comunicação do Centro de Informática da UFPE (CIn) por e-mail para os alunos de graduação. O CIn-UFPE é o maior centro de formação de mão-de-obra especializada em tecnologia da informação da região nordeste, e possui uma estreita relação institucional com o Núcleo de Gestão do Porto Digital, bem como com diversas outras empresas relevantes do ecossistema de inovação instalado em Pernambuco.

Poderíamos, alternativamente, ter optado por consultar redes sociais especializadas em relações profissionais, como *Linkedin* (https://br.linkedin.com/) ou *Glassdoor* (https://www.glassdoor.com/), para fazer esta pesquisa. No entanto, tais ferramentas não disponibilizam acesso gratuito a este tipo de análise detalhada. Além disso, como o foco principal era na indústria local, entendeu-se que o canal consultado era suficientemente representativo.

Analisamos, então, um a um, todos os anúncios de um determinado período de um ano, e selecionamos aqueles que atendiam aos seguintes requisitos: (1) referiam-se a ofertas de empregos abertas no âmbito do Porto Digital ou redondezas geográficas; (2) a oferta de emprego referia-se a alguma tarefa relacionada à engenharia de software, desde que inserida no escopo do SWEBOK, e (3) continham a descrição de ao menos um requisito não-técnico (considerados aqui como sinônimos para *soft skill*) como requisito para contratação. Não fizemos distinção entre chamadas para empregos fixos ou estágios temporários, partindo do pressuposto que ambos refletem de forma semelhante as demandas da indústria.

Em seguida, os termos utilizados para cada descrever cada *soft skill* foi extraído dos anúncios e categorizado em grupos por similaridade semântica, semelhante ao recomendado num processo de análise temática padrão [Cruzes and Dyba 2011]. Em seguida, analisamos a frequência com que cada grupo aparece mencionado dentre os anúncios válidos, representando a sua relevância para o mercado.

Seguindo um processo semelhante ao descrito na análise temática, classificamos os anúncios em quatro funções específicas dentro de um processo de engenharia de software, para as quais são apresentadas análises mais aprofundadas:

- **engenheiros de software:** anúncios com atividades relacionadas a “construção detalhada de um software, através de atividades como codificação, verificação, testes unitários, testes de integração e depuração“ [Abran and Moore 2004]
- **analistas de qualidade:** anúncios que diziam respeito a atividades relacionadas a “garantir que produtos e processos de software, dentro do ciclo de vida de um projeto, estejam de acordo com os requisitos previamente especificados durante o planejamento” [Abran and Moore 2004]

- **designers de software:** anúncios relacionados com o design de interfaces e experiência de usuário [Liddle 1996], e também com a definição de alto nível de arquiteturas e sistemas [Abran and Moore 2004]
- **analista de requisitos:** anúncios relacionados com as atividades de "identificar o propósito de sistemas, através da identificação de *stakeholders* e suas necessidades, documentar esses requisitos de forma adequada para análise, comunicação e posterior implementação da solução" [Nuseibeh and Easterbrook 2000]

Por fim, contrastamos os nossos resultados com estudos previamente relatados, discutindo algumas semelhanças e diferenças notáveis por perfil funcional.

4. Resultados e Discussões

Esta pesquisa foi realizada no primeiro semestre de 2015. No período de 12 meses compreendido entre Fevereiro de 2014 até Janeiro de 2015 foram identificados um total de 420 anúncios de ofertas de emprego, o que corresponde à uma média de 35 anúncios de emprego por mês. Dentre estes, 51% (213) atenderam aos três critérios de seleção determinados na Seção anterior, isto é, (1) tratavam de vagas no Porto Digital, (2) para atividades relacionadas à engenharia de software, e (3) mencionavam explicitamente algum requisito não-técnico.

Em seguida, ao executarmos a análise temática dos *soft skills* mencionados nos 213 anúncios válidos, foram encontrados 20 grupos de fatores, detalhados com as respectivas frequências na Tabela 3. De acordo com esta tabela, as empresas de software demandam prioritariamente o domínio da língua inglesa (46%, 98/213), a qual obteve a maior frequência de menções dentre os aspectos não técnicos identificados. Outros níveis e línguas também são demandadas, tais como Inglês básico (21%), Espanhol (3%) e coreano (.5%), mas estes últimos com uma frequência significativamente menor. Em seguida, destacam-se aspectos relacionados ao comportamento organizacional e individual dos profissionais, tais como trabalhar em equipe (40%, 86/213), proatividade (32%, 69/213) e boa comunicação (31%, 66/213).

Ao separarmos os anúncios por perfil técnico, observamos que as ofertas de emprego para desenvolvedores de software foram responsáveis por 83 dos 213 anúncios identificados que possuíam pelo menos um *soft skill*, correspondendo a 38.96% do total. Os anúncios de oportunidades para analistas de qualidade foram responsáveis por 45 das 213 ofertas identificadas, sendo responsável por 21% do total, 17 vagas de emprego foram ofertadas para designers de software, sendo responsável por 8% do total de anúncios, e 44 oportunidades de emprego para analista de requisitos, representando 20.65% do total de vagas.

A Tabela 4 lista os cinco *soft skills* mais frequentes de cada uma das funções. Juntas, estas quatro funções dão conta de 89% do total dos anúncios. Os 24 anúncios restantes referiam-se a funções diversas, tais como gerentes, consultores, e suporte técnico, de modo que nenhuma delas se repetiu mais do que 10 vezes.

Fluência em inglês e habilidade para trabalhar em equipe aparecem entre os *soft skills* mais frequentes independente da função. Embora o mercado interno brasileiro seja significante, aparentemente as empresas concentradas na localidade estudada tem

um forte foco em exportação de software, e por isso exigem a fluência na língua inglesa. A larga presença de multinacionais neste contexto também pode ser responsável por este resultado. Um fator que reforça este argumento é a presença do fator disponibilidade para viajar e morar fora. No entanto, outro fator que pode levar à esta alta exigência pela língua inglesa é o fato das linguagens de programação terem uma larga documentação em inglês disponível na internet, de modo que programadores que dominam a língua têm naturalmente mais acesso a material técnico e a tecnologias mais atuais.Já a exigência pela habilidade de trabalho em equipe também é justificável, uma vez que a engenharia de software é uma atividade que combina fortemente atividades técnicas e intelectuais com atividades sociais, sendo definida comumente como uma atividade sócio-técnica. Como Guinam et al. (1998) define "na prática, é muito difícil desconectar a forma como as pessoas fazem as coisas, dos métodos, técnicas e tecnologias de computação que elas utilizam".

Tabela 3 - Frequência de menção a *Soft skills*

Soft skill	**Frequencia**		**Soft skill**	**Frequencia**	
Fluência em Inglês	46%	(98/213)	Autodidata	8%	(17/213)
Trabalhar em Equipe	40%	(86/213)	Organizado	8%	(16/213)
Proatividade	32%	(69/213)	Capacidade de Liderança	7%	(15/213)
Boa Comunicação / Capacidade para Ministrar Treinamentos	31%	(66/213)	Adaptável a Mudanças	7%	(15/213)
Disponibilidade para Viajar / Morar Fora	28%	(60/213)	Orientação para Resultados	4%	(9/213)
Inglês Básico	21%	(45/213)	Bom Relacionamento Interpessoal	3%	(8/213)
Gostar de Desafios	21%	(45/213)	Focado	3%	(8/213)
Comprometimento / Responsabilidade	19%	(40/213)	Produtivo	3%	(8/213)
Interessado em Novas Tecnologias	17%	(37/213)	Fluência em Espanhol	2%	(4/213)
Capacidade Investigativa	12%	(26/213)	Fluência em Coreano	.5%	(1/213)

Tabela 4 - *Soft skills* por função técnica, no âmbito do Porto Digital

Desenvolvedores 39% (83/213)		**Analista de Qualidade 21% (45/213)**		**Designer de Software 8% (17/213)**		**Analista de Requisitos 20% (44/213)**	
Fluência em Inglês	67% (57/83)	Trabalhar em Equipe	33% (15/45)	Trabalhar em Equipe	29% (5/17)	Fluência em Inglês	32% (14/44)
Trabalhar em Equipe	57% (47/83)	Capacidade Investigativa	31% (14/45)	Fluência em Inglês	23% (4/17)	Boa Comunicação	23% (10/44)
Capacidade Investigativa	49% (41/83)	Proatividade	29% (13/45)	Proatividade	23% (4/17)	Disp. para viajar / morar fora	20% (9/44)
Proatividade	42% (35/83)	Disp. para viajar / morar fora	27% (12/45)	Boa Comunicação	18% (3/17)	Trabalhar em Equipe	16% (7/44)
Disp. para viajar / morar fora	30% (25/83)	Fluência em Inglês	18% (8/45)	Criatividade, Capacidade de discutir Ideias, Curiosidade, Envolvimento com a comunidade de TI	12% (2/17)	Experiência em ministrar treinamentos	11% (5/44)

Por fim, notamos que enquanto as atividades mais técnicas como desenvolvedores de software e Analista de Qualidade exigem capacidade investigativa e pro atividade com frequência, as atividades mais estratégicas como Designer de Software e Analistas de Requisitos exigem boa comunicação e habilidades de lidar com pessoas, sejam elas colegas de trabalho, integrantes da comunidade de software, clientes ou usuários.

4.1 Demanda por *Soft Skills*: comparativo entre Porto Digital e outros estudos

Para Desenvolvedores de Software

Para desenvolvedores, o *soft skill* de maior frequência é fluência em Inglês. Assim como no Uruguai, este foi o *soft skill* mais mencionado por empresas de software. Embora em Ahmed et al. (2012) não apareça a demanda pela língua, este aspecto pode estar incluído na habilidade de comunicação, que é o *soft skill* mais frequente no seu estudo. Outros dois *soft skills* identificados por Maturro (2013) em sua pesquisa, que assim como nesta, fazem parte do perfil desejado por empresas de software são pro atividade e habilidade para trabalhar em equipe. A capacidade investigativa mencionada nos anúncios do Porto Digital também poderia intuitivamente ser relacionada com a habilidade de pensamento analítico como descrita em Ahmed (2012), no entanto não temos dados suficientes para assegurar esta afirmação. Já as habilidades organizacionais de Ahmed et al. (2012) podem referir-se a comprometimento e responsabilidade de Maturro (2013), e mesmo a disponibilidade de viajar no nosso estudo.

Para Analistas de Qualidade

Como verificado, o perfil que empresas de tecnologia da informação procuram para um analista de qualidade, é um pouco diferente do perfil identificado para um desenvolvedor de software. O *soft skill* mais frequente neste caso é capacidade de trabalhar em equipe. Em contrapartida, saber ler, falar e escrever em Inglês aparece na quinta posição como *soft skill* de maior frequência para analistas de qualidade. Um *soft skill* atrativo para empresas de software para analistas de qualidade, possivelmente pela natureza de suas atividades, é a capacidade investigativa. Esta emergiu com um pouco mais de frequência relativa em analistas de qualidade do que em desenvolvedores.

A fluência em inglês aparece no final da lista, o que contrasta com o resultado encontrado no estudo do Uruguai que, assim como para desenvolvedores, fluência em Inglês foi o *soft skill* mais requisitado. Ahmed et al. (2012) coletou dados referentes a ofertas de trabalho para testadores, o que seria o mais parecido com a nossa categoria de analista de qualidade. Embora também não possamos fazer comparações diretas, aspectos como capacidade investigativa / pensamento analítico, e boa comunicação são amplamente consistentes entre estes dois grupos.

Para Designer de Software

Para um designer de software, o perfil mais procurado pelas empresas de software é o de pessoas que saibam trabalhar em equipe. Um diferencial identificado para um design de software, se comparado com as áreas já analisadas, é a quantidade de *soft skills* que foram mencionados 2 vezes cada, e totalizou 5.88% dos anúncios. São elas: Criatividade, produtividade, capacidade de discutir ideias, pessoas curiosas e que tenham um bom envolvimento com a comunidade de T.I. Vários outros *soft skills* foram mencionados apenas uma vez, como: Comprometimento, gostar de inovação, saber ler, falar e escrever em Espanhol fluentemente, entre outras.

O perfil procurado para os profissionais de designer de software no estudo uruguaio é parecido com o procurado no Porto Digital. Com exceção de vontade de aprender, quinta colocada no estudo de Maturro (2013), os outros *soft skills* são

coincidentes. Já no estudo de Ahmed et al. (2012), os aspectos mais demandados para designers são exatamente os mesmos de desenvolvedores, o que não foi replicado em nosso estudo.

Para Analista de Requisitos

Para um analista de requisitos, o *soft skill* com maior frequência foi novamente fluência em Inglês. As empresas que procuram um profissional deste tipo, também mostraram interesse em que o candidato tivesse experiência em ministrar treinamentos. Além disso, o candidato que deverá ficar em constante contato com o cliente também deve saber se comunicar bem.

Para analista de requisitos, assim como em nosso estudo saber ler, falar e escrever em Inglês também foi o *soft skill* mais desejada pelas empresas de software do Uruguai. No geral, o perfil do analista de requisitos identificados nos três estudos, são muito parecidos, com exceção do pensamento analítico, que é demandado em Ahmed et al. (2012) e não aparece entre os mais frequentes dos outros dois estudos.

5. Discussão e Considerações Finais

Neste trabalho, analisamos 420 anúncios de empregos relacionados a áreas da engenharia de software, a fim de mapear os *soft skills* mais demandados por empresas de software localizadas no Porto Digital. Como vimos, *soft skills* são explícitos em grande parte dos anúncios de emprego analisados, com isso, podemos concluir que as empresas software veem que estes requisitos não técnicos são tão importantes quanto as características técnicas do profissional, e podem desempenhar um papel decisivo na contratação ou não de um candidato à uma vaga.

Os dados utilizados nos permitiram identificar os 20 *soft skills* que são requisitados com maior frequência por estas empresas. Dentre os mais mencionados estão fluência em Inglês, habilidade de trabalhar em equipe, pro atividade, boa comunicação, e disponibilidade para viajar ou morar fora. Observe-se que o perfil de empresas no local (Porto Digital) e a atividade (Engenharia de Software) devem ser grandes influenciadores dos resultados desta pesquisa. Para localidades com perfis de empresas ligeiramente diferentes, ou para uma atividade radicalmente diferente, não esperaríamos que os resultados fossem necessariamente similares aos alcançados nesta pesquisa.

No entanto, neste trabalho fizemos também uma comparação com estudos conduzidos em outros países e pudemos verificar que o perfil desejado nas empresas do Porto Digital, por profissionais das quatro principais áreas da engenharia de software são bastante semelhantes a outras localidades. Isto nos leva a refletir que, uma vez que a cultura é um fator que afeta fortemente os *soft skills*, é sensato concluir que estes fatores são demandas fortemente atreladas à natureza do trabalho em Engenharia de Software, ao invés de meras particularidades locais. Portanto, esperamos que os *soft skills* identificados neste trabalho possam ser, por exemplo, considerados na revisão da matriz de competências e habilidades definida para engenheiros de software no Brasil, pela Sociedade Brasileira de Computação, no FEES.

Esperamos que os resultados apresentados neste estudo levem responsáveis pela elaboração de currículos em cursos superiores da área de computação a refletirem se estão formando a mão de obra de forma adequada para as demandas do mercado [Andrews e Higson, 2008]. Uma questão a ser investigada, por exemplo, seria sobre a importância atribuída nestes currículos ao ensino da língua inglesa nos cursos de computação. Além disso, este mapeamento também pode servir para orientar professores no planejamento didático de seus cursos, para proporcionarem o desenvolvimento, de forma intencional e apropriada, de competências não-técnicas, habilidades e atitudes que sejam complementares aos conhecimentos naturalmente necessários para os alunos atingirem o sucesso em suas disciplinas específicas [da Silva, 2012].

Além disso, é comum encontrarmos na literatura trabalhos que reportam sugestões ou experiências de formação que utilizam dinâmicas que contribuem para o desenvolvimento de *soft skills* como trabalho em equipe e comunicação oral, mas raramente encontramos questionamentos sobre a eficácia destas práticas para o desenvolvimento efetivo das habilidades demandadas pela indústria.

Para fins de simplificação da análise, não fazemos distinção entre contratação efetiva ou para um estágio temporário, assim como não fazemos distinção entre requisitos básicos e desejáveis. Todos são considerados igualmente relevantes. No entanto, pesquisas futuras devem questionar estes pontos e investigar essas nuances dos anúncios de emprego. Um ponto digno de nota é que o fato de um anúncio não mencionar explicitamente os *soft skills* requeridos não significa que os analisados aqui são considerados irrelevantes para eles. Um indicativo disto é que em nenhum dos 420 anúncios analisados, os *soft skills* coincidem exatamente com todos os descritos no RUP (por exemplo). No entanto, é mais provável que isto aconteça pela empresa acreditar que aquele conjunto de requisitos não-técnicos já estão tacitamente distribuídos na cultura daquela sociedade, do que por acreditarem que aquelas características não são importantes.

Outra questão relevante a ser abordada em pesquisas futuras diz respeito a como as empresas avaliam os *soft skills* no momento da seleção dos profissionais. Vide a infinidade de termos técnicos existentes na literatura do comportamento organizacional, em particular das áreas da psicometria e da seleção de profissionais, um levantamento poderia revelar se as empresas de software utilizam sistematicamente alguma técnica fundamentada para tal, ou se avaliam as características de maneira puramente arbitrária [Heckman e Kautz 2012].

Neste último caso, é importante notar que o significado dos *soft skills* pode mudar radicalmente de acordo com os avaliadores [França et al. 2014]. O termo comprometimento, para citar como exemplo, aparece na literatura com significados bastante variados [Meyer et al. 1988]. Desse modo, o que é considerado como "boa comunicação" em uma situação, pode não ter o mesmo julgamento em outra situação análoga. Sendo assim, este levantamento poderia ainda ser útil para documentar parâmetros que podem ser considerados como "bons" ou "ruins" para cada *soft skill* na indústria de software. Seria útil ainda a existência de um modelo, ou taxonomia, que fosse capaz de atribuir uma maior validade e comparabilidade entre pesquisas desta natureza.

Da mesma maneira, é de esperar-se que alguns *soft skills* tenham um peso maior do que outros em um processo seletivo, inclusive podendo variar de acordo com aspectos metodológicos ou culturais da empresa em questão. Portanto, pesquisas futuras devem abordar esta priorização de *soft skills*, o que não foi permitido avaliar no escopo deste artigo uma vez que os anúncios de emprego não carregam tal informação. Seria de igual relevância a condução de pesquisas longitudinais que pudessem, além de fazer uma análise histórica, acompanhar a evolução da demanda industrial por estes *soft skills* no Brasil e no Mundo.

Referências

Abran, L. and Moore, J. (2004). "Guide to the software engineering body of knowledge". IEEE.

Ahmed, F.; Capretz, L. F.; Campbell, P. (2012) Evaluating the Demand for Soft Skills in Software Development. IT Pro January/February

Andrews, J.; Higson, H. (2008) Graduate Employability, 'Soft Skills' Versus 'Hard' Business Knowledge: A European Study. Higher Education in Europe, Vol. 33, No. 4, December 2008

Cruzes, D. S.; Dybå, T. (2011) Recommended Steps for Thematic Synthesis in Software Engineering. International Symposium on Empirical Software Engineering and Measurement - ESEM 2011.

Da Silva, W. C. M.; de Sales, A. B.; Santos, G. A.; Neri, H. R.; Laranjeira, L. A.; Ribeiro Jr, L. C. M. (2012) Anais do V Fórum de Educação em Engenharia de Software (FEES 2012).

Dutra, A. C. S.; Prikladnicki, R. (2014) Formação de Equipes de Alto Desempenho Para Desenvolvimento de Software. Anais do VII Fórum De Educação Em Engenharia De Software (FEES 2014).

Fernandez-Sanz, L. (2010). "Analysis of non technical skills for ICT profiles". 5th Iberian Conference on Information Systems and Technologies.

França, C.; Sharp, H.; da Silva, F. Q. B. (2014) Motivated software engineers are engaged and focused, while satisfied ones are happy. Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement.

Gimenes, I. (2012) "Os dilemas didáticos da Engenharia de Software". Em: Sociedade Brasileira de Computação (2012) "Computação Brasil – Revista da Sociedade Brasileira de Computação", Vol. 28, ed Especial 03/2015. Porto Alegre/RS

Gotel, O; Kulkarni, V; Say, M.; Scharff, C. (2009) A Global and Competition-Based Model for Fostering Technical and Soft Skills in Software Engineering Education. 22nd Conference on Software Engineering Education and Training.

Guinan, P. J.; Cooprider, J. G.; Faraj, S. Enabling Software Development Team Performance Dureing Requirements Definition: A Behavioral Versus Technical Approach. Information Systems Research, v. 9, n. 2, p. 101-125, June 1998.

Heckman, J. J. ; Kautza, T (2012) Hard evidence on soft skills. Labour Econ. 2012 Aug 1; 19(4): 451–464. doi: 10.1016/j.labeco.2012.05.014

Kruchten, P. (2003) Introdução ao RUP – Rational Unified Process, Rio de Janeiro: Editora Ciência Moderna Ltda

Kumar, E. and Sreehari, P. (2010). "Communication skills and soft skills. An integrated approach". Pearson; 1 edition

Liddle, D. (1996). "Design of The Conceptual Model. In Bringing design to software", ACM, New York.

Matturro, G. (2013). "Soft skills in software engineering: A study of its demand by software companies in Uruguay". 6th International Workshop on Cooperative and Human Aspects of Software Engineering, pages 133 – 136.

Meyer, J. P., Paunonen, S. V., Gellatly, I. R., Jackson, D. N. (1988) Organizational commitment and job performance: It's the nature of the commitment that counts. Article in Journal of Applied Psychology 74(1) · December 1988. DOI: 10.1037//0021-9010.74.1.152

Nunes, D. (2012) "Competências do Engenheiro de Software". Em: Sociedade Brasileira de Computação (2012) "Computação Brasil – Revista da Sociedade Brasileira de Computação", Vol. 28, ed Especial 03/2015. Porto Alegre/RS

Nuseibeh, B.; Easterbrook, S. (2000) Requirements engineering: a roadmap. Proceedings of the Conference on The Future of Software Engineering - ICSE '00. doi>10.1145/336512.336523

Pereira, F. (2012). "Afinal, o que são soft skills?". Disponível em <http://mexxer.pt/afinal-o-que-sao-soft-skills/>.Último acesso: 07/06/2016.

Portela, C.S.; Vasconcelos, A. M. L.; Oliveira, S. R. B. (2015) Análise da Relevância dos Tópicos e da Efetividade das Abordagens para o Ensino de Engenharia de Software. Anais do VII Fórum de Educação em Engenharia de Software / Congresso Brasileiro de Software. ISSN: 2175-9677

Porto Digital. (2015). "O que é o Porto Digital". Disponível em < /h/ttp://www.portodigital.org/ >. Último acesso: 07/06/2016.

Schulz, B (2008) The Importance of Soft Skills: Education beyond academic knowledge. Journal of Language and Communication, June 2008

Uma Abordagem de Ensino para o Controle Estatístico do Processo nos cursos de Ciência da Computação

Julio Cezar Costa Furtado[1,2], Sandro Ronaldo Bezerra Oliveira[2]

[1]Departamento de Ciência Exatas e Tecnológicas – Universidade Federal do Amapá (UNIFAP) – Rodovia Juscelino Kubitschek, KM 02 – Macapá – AP – Brasil

[2]Programa de Pós-Graduação em Ciência da Computação – Universidade Federal do Pará (UFPA) – Rua Augusto Corrêa, 01 – Guamá – Belém - PA – Brasil

furtado@unifap.br, srbo@ufpa.br

Abstract. *The Statistical Process Control (SPC) was originally developed in the manufacturing area, to support the implementation of continuous improvement programs in production lines and the use of SPC in the processes improvement is not new to the industry in general. In the context of software organizations, the statistical control can be considered a subject relatively recent, there are still many doubts about its application. Thus, this initiative aims to supply for software development organization the computer science professionals fully able to perform this task.*

Resumo. *O Controle Estatístico de Processos (CEP) foi originalmente desenvolvido na área de manufatura para apoiar a implementação de programas de melhoria contínua em linhas de produção e o uso de CEP na melhoria de processos não é novo para a indústria em geral. No contexto das organizações de software, o controle estatístico pode ser considerado um assunto relativamente recente e ainda há muitas dúvidas sobre a sua aplicação. Assim, esta iniciativa tem o objetivo de fornecer para as organizações de desenvolvimento de software profissionais de ciência da computação plenamente capaz de executar esta tarefa.*

1. Introdução

Um processo considerado sob controle estatístico deve ser estável e repetível (Montgomery, 2007). Assim, o Controle Estatístico do Processo (CEP) é um conjunto de técnicas para garantir que esse objetivo seja atingido. Apesar do uso de CEP nos processos de melhoria não ser novo para a indústria em geral, no contexto das organizações de software o controle estatístico pode ser considerado algo relativamente recente (Alhassan e Jawawi, 2014) e ainda existem muitas dúvidas sobre a sua real aplicação (Boffoli *et al*., 2008).

No entanto, a importância do CEP para a indústria de software tem crescido nos últimos anos, principalmente devido ao uso de modelos de qualidade reconhecidos internacionalmente (Fernández-Corrales *et al*., 2013), como o *Capability Maturity Model Integration* (CMMI) que, em seu nível de maturidade 4, há a necessidade da organização gerenciar quantitativamente a execução de seus processos e buscar a sua otimização contínua.

Assim, neste contexto este trabalho irá: Examinar as práticas de controle de processos estatísticos relevantes para a indústria de software, onde esta análise irá identificar o conhecimento sobre CEP necessário para os profissionais da área desenvolverem suas atividades com o máximo de desempenho; e identificar as práticas do CEP existentes em cursos de Computação. O resultado dessa análise será usado para definir uma nova abordagem de ensino do CEP em cursos de Computação, sendo alinhada com as recomendações do CMMI e as necessidades da indústria de software.

Além desta seção introdutória, e identificação do problema, este artigo está estruturado da seguinte forma: na Seção 2 será exposto o escopo da pesquisa, assim como a sua motivação; na Seção 3 serão descritos a fundamentação teórica deste trabalho; Na Seção 4, as questões de pesquisa e hipóteses serão apresentadas; a Seção 5 cobrirá os métodos de pesquisa utilizados para responder às questões de pesquisa e para testar a hipótese; e as contribuições e resultados esperados são apresentados na Seção 6.

2. Motivação e Escopo da Pesquisa

O uso do CEP em organizações de desenvolvimento de software se mostra complexa, pois as técnicas existentes no contexto não consideram as particularidades presentes em um processo de desenvolvimento de software (Fernández-Corrales *et al*., 2013).

Em relação à qualidade do processo de desenvolvimento e ao modelo CMMI (SEI, 2010), nos primeiros níveis desse programa de melhoria, as organizações adotam uma medida que simplesmente consiste na coleta de dados, a partir da execução do projeto, e os compara com os valores planejados. Apesar desta ser uma abordagem suficiente, não é adequada para as organizações que procuram pela alta maturidade, avaliar e evoluir seus processos. Nessas organizações é necessário executar o controle estatístico de processos de software para conhecer o seu real comportamento, determinar o seu desempenho em execuções anteriores e prever seu desempenho em projetos atuais e futuros, certificando-se que eles são capazes de atingir as metas estabelecidas e identificar ações corretivas e de melhoria quando for o caso (Barcellos *et al*., 2010).

A indústria costuma reclamar que os cursos de graduação não ensinam as habilidades necessárias para os alunos trabalharem de forma eficiente (Wangenheim e Silva, 2010). Assim, esta dificuldade é particularmente acentuada ao lidar com as atividades mais complexas de uma organização com um alto nível de maturidade em seus processos. As empresas de software têm de complementar o conhecimento de recém-licenciados com formação e têm de fornecer competências técnicas e não técnicas relacionadas com o processo de desenvolvimento de software (Sargent, 2004). Em geral, a indústria de software sofre com a falta de profissionais qualificados para trabalhar em atividades que envolvem o processo de desenvolvimento de software (ABES, 2014).

De acordo com Lethbridge *et al*. (2007), esta deficiência na formação de graduados na área de ES é o resultado de uma formação inadequada. Este achado pode ser reforçado com a investigação realizada por (Sargent, 2004), o que revela que: apenas 40% dos profissionais de TI (Tecnologia da Informação) Estados Unidos têm formação nesta área; e apenas 40% destes profissionais estão cientes dos principais campos de Engenharia de Software, tais como requisitos, arquitetura, testes, fatores humanos e gestão de projetos. Embora não tenha sido encontrado dados estatísticos específicos em

relação ao CEP, é fácil de inferir que a realidade dos profissionais ES neste domínio específico não deve ser diferente do cenário observado pelos autores citados.

3. *Background* do Ensino da Engenharia de Software

De acordo com a ACM / IEEE (2015), a ES é uma disciplina interessada na aplicação de teoria, conhecimento e prática para o desenvolvimento eficaz e eficiente de sistemas de software que atendam aos requisitos dos usuários. Os profissionais de ES devem ter a capacidade de compreender o desenvolvimento de software como um processo e assegurar prazos, custos e qualidade do produto desenvolvido (Portela *et al*., 2015).

Uma pesquisa realizada (Wangenheim e Silva, 2010) buscou descobrir a opinião dos profissionais da área de engenharia de software sobre a relevância dos temas abordados nos cursos de Ciência da Computação. Como resultado, a pesquisa indica que existe uma falta de atenção para alguns tópicos da ES. Para determinado tema, foi possível identificar até mesmo uma completa falta de consideração de professores e alunos. Por exemplo, o tema "Gerenciamento de Configuração de Software", o que na prática é considerado como uma base essencial não só para engenheiros de software, mas também para qualquer profissional de software (Wangenheim e Silva, 2010).

Por outro lado, apesar da importância destes conhecimentos relativos às atividades da ES, Lethbridge (2000) verificou-se que os profissionais aprendem mais sobre estas atividades durante o seu trabalho do que durante a sua formação. Isto pode ocorrer pelo simples fato de que, se tivermos em conta a sugestão de um total de pelo menos 2800 horas para um curso de Ciência da Computação (SBC, 2005), a alocação de cerca de 36 horas para o tópico de ES não corresponde com a percepção da importância deste tema, consequentemente, não há tempo suficiente para ser gasto em tópicos importantes. Gary *et al*. (2013) propõem um modelo pedagógico para o ensino de ES nos cursos de graduação em computação. Os estudantes assistem a palestras e praticam os conceitos aprendidos em sessões de laboratório realizados em cada semana do curso. Esta proposta combina a sala de aula tradicional com o *Problem Based Learning* (PBL).

4. Perguntas de Pesquisa e Hipóteses

O principal objetivo deste trabalho é propor uma abordagem de ensino de Controle Estatístico do Processos em cursos de computação, sendo esta abordagem baseada em padrões de qualidade e nas práticas de mercado identificadas. Para este fim, as seguintes questões de pesquisa (RQ) devem ser respondidas:

1. RQ1. Quais são as práticas de controle estatístico do processo mais utilizados pela indústria de software?
2. RQ2. Quais são os tópicos sobre controle estatístico do processo abordados nas diretrizes curriculares dos cursos de computação?
3. RQ3. Quais são os tópicos sobre controle estatístico de processo abordados nos currículos dos cursos de computação?
4. RQ4. Quais são os tópicos sobre controle estatístico do processo efetivamente aprendidos pelos alunos de computação?
5. RQ5. Quais são as habilidades do controle estatístico do processo exigidas pela indústria de software e quais delas foram adquiridos nos cursos de computação?

As questões de pesquisa foram definidas para tentar refutar a seguinte hipótese nula: H0. A atual abordagem para o ensino do Controle Estatístico do Processo atende às necessidades da indústria de software.

Se a hipótese nula for refutada, irá ser testada a seguinte hipótese alternativa: H1. A abordagem atual para o ensino do Controle Estatístico do Processo não atende a indústria de software devido ao desalinhamento entre o currículo da disciplina de Engenharia de Software e as reais necessidades da indústria.

5. Método de Pesquisa e Progresso

A realização deste trabalho conta com a participação de três alunos de graduação em Ciência da Computação, um aluno de mestrado em Ciência da Computação e um aluno de doutorado em Ciência da Computação. Além desses alunos, o projeto também conta com a participação de dois consultores de melhoria de processos.

Para responder a RQ1 e ajudar a RQ5, será realizada uma revisão sistemática (Kitchenham, 2007)(Dyba *et al*., 2007), com o objetivo de identificar quais práticas de CEP são usadas pela indústria do software. Isto irá permitir a criação de um catálogo com as principais práticas selecionadas a partir desta revisão. Este catálogo então irá compor parte da análise que para apoiar ou refutar a hipótese H0, fornecendo as informações sobre as necessidades reais da indústria de software. Atualmente a pesquisa está nesta etapa, já elaborando o relatório final da revisão sistemática junto a um catálogo e manual de uso de ferramentas de software que implementem o CEP alinhado aos modelos de qualidade.

Para responder a RQ2, será realizada uma revisão da literatura sobre as orientações curriculares da ACM/IEEE (2013) e SBC (Sociedade Brasileira de Computação) (2005) com o objetivo de identificar quais tópicos do CEP são contemplados nas orientações. O resultado desta avaliação pode apoiar ou refutar a hipótese H0, dando evidências de que as atividades sugeridas nas diretrizes curriculares atendem às demandas da indústria de software. Para responder a RQ3, uma pesquisa será realizada com os professores dos cursos de graduação em computação. O objetivo deste estudo é analisar quais as atividades de CEP identificadas na revisão da literatura estão incluídas nas disciplinas curriculares ES. Estes resultados também podem validar a hipótese H0.

Então, na RQ4 uma outra pesquisa será realizada, desta vez com os alunos que concluíram as disciplinas de Engenharia de Software. Esta pesquisa tem como objetivo avaliar se os alunos estão realmente aprendendo as atividades da CEP, se estas forem contemplado pelas disciplinas ES. Os resultados também podem validar H0, dando provas de que o problema pode estar na abordagem de ensino adotada nas salas de aula. Ambas as pesquisas serão aplicadas para cursos de graduação em Ciência da Computação de universidades públicas e privadas no Brasil e vai seguir as orientações de Kitchenham e Pfleeger (2008).

Após as cinco questões de pesquisa serem respondidas, se terão os seguintes resultados: o catálogo de práticas de CEP utilizados pela indústria de software; as recomendações das diretrizes curriculares para CEP; as abordagens de ensino para CEP existentes em cursos de computação; e quais os tópicos são considerados importantes pela indústria.

Estes resultados vão orientar o desenvolvimento da abordagem de ensino CEP. A abordagem vai adotar o Aprendizagem Baseada em Problemas como método de ensino (Bessa *et al*., 2012). A partir dos resultados obtidos em (Portela *et al*., 2015), observa-se que os alunos estão mais interessados na realização de atividades práticas, tais como projetos de desenvolvimento em laboratórios que simulam situações próximas às que serão encontradas no mercado. Para avaliar e validar a proposta de abordagem de ensino, será realizado um experimento controlado em uma disciplina de Engenharia de Software de um curso de graduação em Ciência da Computação. Inicialmente, este experimento será realizado em duas turmas de computação de duas universidades públicas brasileiras. Este experimento seguirá as orientações propostas por Wohlin (2000).

6. Contribuições Esperadas e Resultados

Esta pesquisa tem como objetivo identificar práticas de Controle Estatístico de Processos e sua relevância para a indústria de software atual, identificando possíveis problemas na abordagem de ensino em engenharia de software, especificamente sobre CEP e definir uma nova abordagem de ensino que assegure que os estudantes de graduação de cursos de computação terão em sua formação de básica os tópicos de CEP exigidos pelo mercado.

O cenário atual da educação mostra que determinados temas são considerados menos relevantes pelos professores e, portanto, têm um baixo nível de aprendizagem dos alunos. Acontece que esses tópicos consomem uma significante carga horária da disciplina de engenharia de software, enquanto que alguns temas considerados mais relevantes, têm uma menor carga. Isto talvez se de ao fato de que não há tempo suficiente para ensinar de forma eficaz todos os tópicos dessas unidades relevantes.

Referências

ABES, “Brazilian software market: scenario and trends,” in Brazilian Software Market and Services 2014, 1st ed. São Paulo, Brazil: Brazilian Association of Software Companies, 2014.

ACM/IEEE. Computer science curricula 2013. Curriculum guidelines for undergraduate degree programs in Computer Science. December 20, 2013.

Alhassan, M. A. e Jawawi, D. N., “Sequential Strategy for Software Process Measurement that Uses Statistical Process Control”, in: 8th Malaysian Software Engineering Conference (MySEC), pp. 37-42, 2014.

Barcellos, M., Falbo, R. e Rocha, A., “Establishing a Well-Founded Conceptualization about Software Measurement and High Maturity Levels”, in: 2010 7th International Conference on the Quality of Information and Communication Technology, pp. 467-472, 2010.

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

Boffoli, N., Bruno, G., Caiavano, D. e Mastelloni, G., “Statistical Process Control for Software: a Systematic Approach”, Proceedings of the Ninth European Conference on Software Maintenance and Reengineering, pp. 288-293, 2008.

Dyba, T., Dingsoyr, T. e G. Hanssen, “Applying Systematic Reviews to Diverse Study Types: An Experience Report”, First International Sysmposion on Empirical Software Engineering and Measurement, pp. 225-234, 2007.

Fernández-Corrales, C., Jenkins, M. e Villegas, J., “Application of Statistical Process Control to Software Defect Metrics: an Industry Experience Report”, in: 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 323-331, 2013.

Gary, K., Lindquist, T., Bansal, S. e Ghazarian, A., “A Project Spine for Software Engineering Curricular Design”. In Proceedings of 26th Conference on Software Engineering Education and Training (CSEET), pp. 299-303, 2013.

LethBridge, T., “What knowledge is important to a software professional?”, Journal Computer, 33(5), IEEE Computer Society Press, USA, pp 44-50, 2000.

Lethbridge, T., Diaz-Herrera, J., LeBlanc, R. e Thompson, J., “Improving software practice through education: Challenges and future trends”, Conference Future of Software Engineering, Minneapolis, MN, pp.12-28, 2007.

Kitchenham, B., Guidelines for performing Systematic Literature Reviews in Software Engineering, Technical Report EBSE - 2007-01, Departament of Computer Science Keele University, Keele, pp. 19-28, 2007.

Kitchenham, B. e. Pfleeger, S., Personal Opinion Surveys, in Guide to Advanced Empirical Software Engineering, Springer, 2008.

Montgomery, D., “Introduction to Statistical Quality Control”. John Wiley & Sons, 2007.

Portela, C. S., Vasconcelos, A. e Oliveira, S. B., “How to Develop Competencies and Abilities Professional in Software Engineering in Undergraduate Students?”, in Int'l Conf. Frontiers in Education: CS and CE - FECS'15, pp. 91-94, 2015.

Sargent, J., “An overview of past and projected employment changes in the professional IT occupations”, Computing Research News, vol. 16, no. 3, pp. 1-21, 2004.

SBC. Currículo de Referência para cursos de graduação em Bacharelado em Ciência da Computação e Engenharia da Computação, 2005.

Software Engineering Institute, "CMMI® for Development, Version 1.3, Improving processes for developing better products and services," No.CMU/SEI-2010-TR-033. Carnegie Mellon University, Pittsburgh, PA, 2010.

Wangenheim, C. e Silva, D., “Qual o conhecimento da Engenharia de Software é importante para um profissional de software”, II Fórum de Educação em Engenharia de Software, Fortaleza, Brasil, 2009.

Wohlin, C., et al., “Experimentation in software engineering: an introduction”, in Kluwer Academic Publishers, Norwell, MA, 2000.

Uma Investigação sobre Estilos de Aprendizagem e Hábitos de Estudo de Engenheiros de Software

César França[1], José Adson Cunha[2], Dayan Adjarde[3], Fagner Alan[3]

[1]Departamento de Informática e Estatística (DEINFO)
Universidade Federal Rural de Pernambuco.

[2]Centro de Informática (CIn)
Universidade Federal de Pernambuco

[3]Centro de Estudos e Sistemas Avançados do Recife (CESAR)
R. do Brum, 77 - Recife, PE, 50030-260

cesar@franssa.com, jaogc@cin.ufpe.br, dayan@insideweb.com.br, fagneralan@outlook.com

Abstract. *This article is aimed at understanding how professional software engineers differ in terms of their learning styles, and how these learning styles relate to their autonomous habits and practices of studying. Using the VARK Questionnaire, we conducted a Survey with 119 engineers in Brazil, and documented that there is a tendency among the participants to prefer knowledge sources, respectively, knestesics (K), Auditive (A), Visuals (V) and Textuals (R). Moreover, the engineers reported that planning and focus are two fundamental elements to enable a good practice of learning at the working environment.*

Resumo. *Este artigo busca entender como engenheiros de software profissionais diferem em seus estilos de aprendizagem, e como estes estilos de aprendizagem se relacionam com os seus hábitos e práticas de estudo autônomo. Utilizando o VARK Questionnaire, realizamos um survey com 119 engenheiros no Brasil, e relatamos que existe uma tendência entre os participantes de preferirem fontes de conhecimento, respectivamente, cinestésicas (K), auriculares (A), visuais (V) e textuais (R). Além disso, os próprios engenheiros relatam que planejamento e concentração são elementos fundamentais para habilitar a aprendizagem no contexto industrial.*

1. Introdução

São desafiadoras as experiências relacionadas ao estilo de aprendizagem e hábitos de estudo dos engenheiros de software (ES), um campo que busca a compreensão de quais meios facilitarão a evolução do conhecimento dos profissionais da área. Tendo em vista as fontes relacionadas ao objeto do estudo, encontram-se nos artigos estudados as premissas base para o padrão de aprendizagem do profissional de engenharia de software.

Segundo Guerra [2014], os processos de aprendizagem precisam ser avaliados com cuidado, porque cada indivíduo processa as informações de forma diferente. Ao nos aprofundarmos com a particularidade de cada profissional, conseguiremos absorver uma visão geral dos hábitos e estilo de aprendizagem de cada um. Piaget em seu

construtivismo psicogenético defende que os estímulos oferecidos pelo ambiente dão ao indivíduo a capacidade de desenvolver seu intelecto [LAKOMY, 2008, p. 30]

Sendo assim, estudos sobre estilos de aprendizado e cognição podem trazer diversos benefícios tanto na área de educação [Riding; Cheema, 1991; Felder-Silverman, 1988; Witkin et al., 1975; Saeed; Yang; Sinnappan, 2009; Franzoni; Assar, 2009], como ganhos de desempenho na área profissional [Richards et al., 2011].

Neste artigo, buscamos aprofundar o conhecimento sobre a aprendizagem de engenheiros de software em dois estados brasileiros: Pernambuco e Paraíba. Em uma direção mais generalista, buscamos mapear a distribuição de perfis de aprendizagem entre os engenheiros pesquisados, e detectamos que há uma tendência destes engenheiros de expressarem uma maior habilidade de aprender através de estímulos respectivamente cinestésicos, auditivos, visuais e, por último, textuais. Em uma direção mais pragmática, buscamos dados no campo para compreender hábitos práticos de estudo dos engenheiros de software na indústria. Além de obtermos uma lista genérica de fontes de aprendizado consultadas por engenheiros profissionais, os dados indicam que planejamento e concentração são dois elementos fundamentais para condicionar a situação de aprendizagem de engenheiros de software, sendo transversais aos estilos de aprendizagem e às suas práticas de estudo.

Na seção seguinte, descrevemos uma breve revisão bibliográfica para delimitar o framework conceitual que orientou a presente pesquisa, e descrevemos trabalhos recentes relacionados ao tema. Na Seção 3, descrevemos os métodos e instrumentos utilizados para coleta e análise de dados da pesquisa. Na Seção 4, descrevemos e discutimos os resultados gerados a partir da análise. Por fim, na Seção 5 deste artigo, apresentamos algumas considerações finais sobre limites à generalização e oportunidades para investigações futuras.

2. Revisão Bibliográfica

2.1. Estilos de Aprendizagem e o modelo VARK

Segundo Riding e Cheema [1991] o termo estilo cognitivo foi usado pela primeira vez por Allport em 1937, que foi descrito como o modo típico e habitual de uma pessoa resolver problemas, pensar, perceber e lembrar. A partir daí diversos pesquisadores tem trabalhado e criaram diversas definições [De Bello, 1990], nomenclaturas próprias para os modelos desenvolvidos, resultando numa quantidade variada de títulos para estilos de aprendizado e cognitivos em muitos casos semelhantes [Dunn et al., 1981; Adán-Coelho et al., 2008]. Coffield et. al [2004] em sua pesquisa que avaliou os modelos desenvolvidos entre 1909 a 2003, afirma que há mais de 70 diferentes. Para Moazeni [2013], os instrumentos mais comuns para identificar os estilos de aprendizado e de cognitivos são: Indicador de Estilo de Aprendizado de Kolb [1984], Inventário VARK [Fleming; Mills, 1992], Índice de Felder –Silverman [Felder; Silverman; Solomon, 2000], Delineador de Estilos de Gregorc e a Pesquisa de Preferência Ambiental de Dun e Dun [1979].

Buscando operacionalizar o conceito de estilos de interação através da preferência individual por diferentes tipos de estímulos sensoriais, Fleming e Mills [1992] propuseram o modelo VARK que é o acrônimo de Visual, Auditory, Read/Writing e Kinesthetic (em português; Visual, Auricular, Textual e Cinestésico) que são categorias sensoriais dos seres humanos empregados no aprendizado e processamento de

informações utilizados em seu modelo. A partir destas categorias, o VARK se propõe a identificar o comportamento de aprendizagem de estudantes, e recomendar técnicas personalizadas para melhorar a efetividade do aprendizado para cada um dos perfis. A **Tabela 1** sintetiza os quatro estilos e as respectivas práticas de estudo consideradas como efetivas para si.

O *VARK Questionnaire* é o instrumento proposto por Fleming e Mills [1992] para identificação de estilos de aprendizagem de acordo com o modelo VARK. Ele tem sido utilizado em pesquisas em diversas áreas para subsidiar a identificação e mapeamento dessas categorias, tais como educação [Brown; Cosgriff; French, 2008; Marcy, 2001; Robertson et al., 2011], tecnologia [Katsioloudis; Fantz, 2012], saúde [Boyde et al., 2009], esportes [Dunn, 2009; Braakhuis et al., 2015], entre outros, obtendo resultados representativos e confiáveis para o público pesquisado.

Tabela 1 - Dimensões do VARK (elaborado com base nas informações disponíveis em http://vark-learn.com)

Estilo de Aprendizagem	Descrição	Estratégias de Aquisição de conhecimento adequadas
(V) Visual	Preferência por gráficos e formas simbólicas de representação de informação	Use sublinhados, cores diferentes, destaques; Fluxogramas; Imagens, vídeos, cartazes, slides; Docentes que utilizem gestos e linguagem pictórica; Livros didáticos com diagramas e fotografias; Gráficos; Símbolos; Espaços em branco (lacunas)
(A) Auricular	Preferência por informação audível	Assista atentamente às aulas; Participe em debates e explicações; Debata os temas com os outros; Discuta os assuntos com os professores; Explique as novas ideias que adquiriu a outras pessoas; Use um gravador de áudio; Recorde exemplos interessantes, histórias, piadas; Descreva os aspectos gerais, imagens e outros recursos visuais para alguém que não tenha estado presente e não esteja dentro do assunto; Deixe espaços em branco nas suas anotações para mais tarde recordar e os 'preencher'.
(R) Textual	Preferência por informações impressas e textos	Listas; Tópicos; Dicionários; Glossários; Definições; Fotocópias; Livros didáticos; Leituras – biblioteca; Apontamentos (muitas vezes literalmente transcritos); Os professores que são bons falantes fornecem muita informação através das suas afirmações e explicações.; Ensaios; Manuais (de informática, de práticas laboratoriais)
(K) Cinestésico	Preferência por métodos práticos experimentais, simulações, dramatizações	Todos os seus sentidos – visão, tato, paladar, olfato, audição; Laboratórios; Viagens ao campo; Visitas de estudo; Exemplos de princípios; Docentes que utilizam exemplos da vida real; Aplicações; Abordagens mãos-na-massa (computadores); Tentativa e erro; Coleções de tipos de rochas, plantas, cascas, ervas; Exposições, amostras, fotografias; Receitas – soluções para os problemas, testes de exames anteriores

Segundo Fleming e Mills [1992], o mais comum é que indivíduos apresentem múltiplas preferências, com estilos diferentes que se sobressaem em situações de aprendizagem diferentes, configurando-se num perfil chamado de Multimodal. Os indivíduos multimodais têm a habilidade de absorver as informações transmitidas de acordo com os seus diferentes perfis de preferência com proficiência semelhante, adaptando-se melhor, por exemplo, a diferentes professores, materiais, cursos, etc.

À luz teórica de Fleming e Mills [1992], Guerra [2014] evidencia que quanto mais assertivo o estímulo, maior o será o aprendizado, uma vez que a base de conhecimento é adquirida conforme a(s) principal(is) habilidade(s) de cada aprendiz.

Na área de engenharia de software, no entanto, o mais comum é encontrar pesquisas que considerem estilos de aprendizagem em aplicações educacionais pois, segundo Valente [1989], um bom uso do software educacional se faz quando a estratégia de desenvolvimento do conhecimento combina com a expectativa do aprendiz/usuário. Mais recentemente, os estilos de aprendizagem vêm sendo explorados para subsidiar a construção de Ambientes Virtuais de Aprendizagem mais adequados a determinados conjuntos de usuários, como em Cury [2000] e Geller [2004].

No entanto, ainda são raros os trabalhos que investigam estilos cognitivos no contexto da aprendizagem da Engenharia de Software como disciplina técnica. No contexto acadêmico, Thomas et. al [2002] demonstrou que existem diferenças de desempenho significativas entre estudantes de programação que tiveram contato com um conteúdo único, indicando que os métodos e técnicas de ensino utilizados poderiam estar beneficiando um sub conjunto dos estudantes e prejudicando outro, de acordo com os estilos de aprendizagem deles. Na linha da proposta do presente trabalho, Capretz [2006] apresenta um mapeamento de traços de personalidade de 68 estudantes de graduação em computação canadenses, utilizando o MBTI – um instrumento largamente conhecido e referenciado na psicologia, como forma de subsidiar professores para utilizarem diferentes estratégias de ensino em sala de aula. Utilizando buscas não sistemáticas em respeitadas bases de dados científicas e no *Google Scholar* ®, não encontramos trabalhos que tratassem especificamente da realidade de engenheiros de softwares profissionais.

3. Procedimentos Metodológicos

Objetivando mapear especificamente a distribuição de estilos de aprendizagem entre engenheiros de software profissionais, e a relação destes estilos com práticas e hábitos de estudo, adotamos uma abordagem metodológica indutiva, e conduzimos um levantamento transversal com engenheiros de software, coletando dados quantitativos e qualitativos [Marconi e Lakatos, 2003].

Sobre os estilos de aprendizagem, instanciamos uma versão em português do *VARK Questionnaire*[1]. O referido questionário é composto por 16 questões contendo 4 alternativas. Cada uma das alternativas reflete a preferência por um dos quatro estilos de aprendizagem. A análise do resultado do questionário aponta então para os estímulos sensoriais preferidos dos indivíduos. O questionário foi disponibilizado em papel, distribuído à rede de relacionamentos dos autores, e, posteriormente, consolidado em uma planilha para fins de análise. As respostas obtidas, portanto, não estão livres de problemas originários de amostragens com auto seleção. Por razões de propriedade, o questionário não pode ser reproduzido neste artigo, mas as questões podem ser integralmente consultadas no link fornecido.

Como explicado anteriormente, as preferências não são mutualmente exclusivas, portanto os respondentes podem marcar múltiplas respostas. Dessa maneira, as quatro variáveis (V, A, R, K) são independentes, e a escala de pontuação pode variar entre 0 e 16. Os valores absolutos, no entanto, não carregam significado, e servem apenas para

[1] http://vark-learn.com/questionario/

determinar a ordenação das dimensões preferidas. Para analisar o padrão de preferências entre os engenheiros de software, comparamos as diferenças entre as ordens de preferência resultante de cada indivíduo, utilizando o teste de Wilcoxon (1945), que avalia diferenças de postos, sobre dados pareados de uma amostra única.

Além dos estilos de aprendizagem coletados com o questionário VARK, coletamos dados adicionais em um segundo questionário, com um conjunto de perguntas dirigidas a coletar opiniões e experiências dos participantes sobre seus hábitos de estudo. A pergunta aberta foi então analisada seguindo a técnica de análise temática, agrupando as menções dos participantes em códigos e categorias geradas a partir dos próprios dados [Cruzes e Dyba, 2011]. Os resultados das duas análises encontram-se descritos na seção seguinte.

4. Resultados e Discussão

A coleta de dados foi realizada entre os meses de Abril e Maio de 2016, atingindo um total de 119 respondentes. Dentre estes, 81% correspondiam ao sexo masculino, enquanto 19%, ao sexo feminino. A concentração dos respondentes foi nos estados de Pernambuco (55,8%) e Paraíba (34,2%), mas houve respondentes de diversos outros estados brasileiros, tais como AL (3), AM (1), CE(1), DF(1), SP (3), PI (1), RN (1) e RR (1). A renda familiar dos participantes concentra-se em 3 a 10 salários mínimos (54,2%) e 10 a 20 salários mínimos (35,8%). O nível de instrução dos participantes está distribuído entre Graduados (25,8%), Pós-graduados (21,7%), Mestres (46,7%) e Doutores (5,8%), absolutamente todos em áreas relacionadas a ciência da computação ou correlatos.

Em relação a aspectos profissionais, mapeamos o tipo de empresa às quais os participantes estão vinculados, e obtivemos que 40,8% dos participantes são do setor público e 58,3% do setor privado, evidenciando que esta pesquisa não está enviesada de acordo com o setor de atuação dos participantes. Já em relação à experiência profissional, obtivemos que 30% tem entre 0 e 5 anos de experiência, 30,8% entre 6 e 10 anos, 28,3% entre 11 e 15 anos de experiência e 10,9% possui acima de 16 anos de experiência profissional, refletindo também um bom balanceamento de participantes com diferentes níveis de experiência profissional.

4.1. Distribuição de Estilos de Aprendizagem

A primeira análise realizada avaliou a existência de uma tendência ou padrão de concentração de engenheiros de software em algum conjunto específico de estilos de aprendizagem do VARK. A Tabela 2 mostra as médias e desvios padrão absolutos, referentes aos quatro estilos de aprendizagem. Embora todas as médias se aproximem da mediana, utilizando o teste de Kolmogorov-Smirnov [Massey, 1951] verificamos que nenhuma das quatro dimensões do VARK possui distribuição Normal entre os dados.

No entanto, a análise dos valores absolutos não produz informação útil, uma vez que o VARK *Questionnaire* sugere uma interpretação das preferências individuais de acordo com a ordenação da pontuação nos estilos de aprendizagem. Sendo assim, para cada indivíduo, verificamos a classificação individual (1º, 2º, 3º e 4º) de cada estilo de aprendizagem, e retratamos na Tabela 3 a concentração dos estilos em cada posto.

Tabela 2 - Descrição absoluta dos dados

Est.	N	Méd.	Desv. Pad.
(V)	119	6,29	3,13
(A)	119	7,73	2,62
(R)	119	5,40	2,89
(K)	119	8,11	2,68

Tabela 3 - Descrição relativa dos dados

Est./ Posto	1º	2º	3º	4º
(V)	25%	17%	27%	31%
(A)	38%	29%	28%	6%
(R)	9%	18%	29%	44%
(K)	49%	34%	11%	7%

Tabela 4 - Resultados da aplicação do teste de Wilcoxon

Est.	(V)	(A)	(R)
(A)	3.3**		
(R)	2.4*	6.1**	
(K)	5.2**	2.0*	6.5**

* $p < 0.05$
** $p < 0.01$

Por fim, utilizando o teste de Wilcoxon para avaliar a significância da diferença entre os postos de cada estilo de aprendizagem (resultados na Tabela 4), evidenciamos que há diferença significativa entre a concentração de todos os estilos em cada um dos seus postos. Sendo assim, de acordo com estes dados, é possível afirmar que há uma tendência entre os engenheiros de software estudados de seguirem o padrão 1º=K, 2º=A, 3º=V e 4º=R na sua ordem preferência de estilos de aprendizagem.

4.2. Fontes de conhecimento e sua relação com diferentes estilos de aprendizagem

Há diversos meios de se adquirir conhecimento, variando em disponibilidade, eficiência, dificuldade, dedicação e custos. Quando questionados sobre seus hábitos de estudo, quase todos os participantes da pesquisa destacaram o que geralmente usam para fins de aprendizado. Apenas 3 participantes não forneceram conteúdo suficiente para analisar a resposta, enquanto a média de quantidade de caracteres nas respostas foi 248. A Tabela 5 contém os resultados da análise temática das respostas.

A partir das estratégias de aquisição de conhecimento para cada estilo de aprendizagem apresentado anteriormente na Tabela 1, mapeamos as práticas informadas pelos engenheiros de software com o respectivo estilo de aprendizagem, descritas na Tabela 5.

Tabela 5 - Fontes de estudo de engenheiros de software por perfis de aprendizagem (n=119)

Indetermináveis	(V)	(A)	(R)	(K)
44% Youtube/Video 35% Curso online 22% Blog 13% Sites 2% Aula presencial	3% Diagramas 1% Figuras	7% Discussão em grupo 4% Podcast 1% Leitura em voz alta	43% Livro 29% Fóruns 21% E-book 13% Artigo 8% Escrita de resumo 7% Tutorial	17% Experimentação

Conforme indicado na primeira coluna, algumas práticas foram mapeadas como indetermináveis, uma vez que podem envolver mais de uma dimensão do VARK. Por exemplo, um vídeo pode oferecer aulas explicativas (V), áudios (A), slides com os principais tópicos sobre o tema (R), e exemplos práticos para subsidiar a realização de exercícios (K).

Observe ainda que a distribuição de frequências dos temas nesta tabela não reflete necessariamente uma escala de preferência. O objetivo desta análise foi tão somente revelar e documentar algumas das fontes de conhecimento buscadas pelos engenheiros de software. Hipotetizamos que os participantes com maiores índices de (R) poderiam ter a tendência de documentar respostas mais compridas, e consequentemente influenciar no resultado desta análise, no entanto esta correlação mostrou-se insignificante (p=.94). De todo modo, as sugestões de fontes indicadas na pergunta podem ter influenciado as respostas dos participantes.

4.3. Hábitos de estudo no contexto industrial

A literatura sugere que o indivíduo possua hábitos que tornem o esforço sustentável, proporcionando um senso de responsabilidade e valor pelo próprio aprendizado [Credé; Kuncel, 2008]. Hábitos de estudo denotam o grau com que o indivíduo se engaja em atos regulares de estudo caracterizado por rotinas apropriadas de estudo em um ambiente propício ao aprendizado.

Em uma escala de 1 (nunca) a 5 (sempre), os engenheiros de software foram questionados sobre os hábitos de estudo dentro e fora do horário de trabalho. Quanto à importância de o profissional aprender coisas novas independentemente das tecnologias que utiliza atualmente no trabalho, a grande maioria foi favorável (5 - 73,3%; 4 - 22,5%). No entanto, a maioria indicou que nem sempre tem a oportunidade de dedicar tempo para aprender coisas novas dentro do horário de trabalho (3 - 39,2%; 4 - 32,5%) e fora do ambiente de trabalho (4 - 33,3%; 3 - 26,7%). Além disso, as coisas que os engenheiros de software decidem aprender nem sempre têm relação direta com as tarefas que executam no horário de trabalho (3 - 32,5%; 4 - 28,3%).

Através da análise temática das perguntas realizadas ao final do questionário sobre os hábitos de estudo, foram identificados elementos relacionados ao **planejamento** e **concentração**. A necessidade de planejamento através do pré-estabelecimento de tempo de estudo, delimitando previamente o objetivo a ser alcançado em cada etapa do estudo, foi mencionado pelos engenheiros de software, conforme exemplificado abaixo:

> — "*Acho importante ter a disciplina diária de estudos. Planejar horários e tentar cumprir o planejado.*" (ES_071)

Quanto à necessidade de concentração, os engenheiros de software enfatizaram o uso de música para se manter o foco e técnicas de gerenciamento de tempo. A música de fundo tem sido utilizada em vários tipos de ambiente de trabalho. Huang and Shih [2011] concluíram que a atenção do indivíduo é influenciada pela percepção de música de fundo durante a realização de tarefas, principalmente quando há um sentimento do indivíduo em relação à música. De acordo com os autores, evitar tal tipo de música pode ajudar a desviar a atenção, aumentar a concentração e promover o bom desempenho no trabalho. O uso da música pelos engenheiros de software é exemplificado através da extração abaixo:

> — "*Ouvir música enquanto estudo me ajuda bastante a me concentrar e evitar distrações externas."* (ES_16)

> — "*Quando for sentar para estudar, manter o foco, escutar uma musica (de preferência uma que você não saiba cantar).*" (ES_10)

Além disso, foi mencionado também o uso de técnicas para gerenciamento de tempo, como a técnica pomodoro [Cirillo, 2013], a qual sugere o uso de um cronômetro para dividir o trabalho em períodos de 25 minutos chamados de "pomodoros".

> — "*Uma sugestão é o uso de técnicas como pomodoro que podem auxiliar no momento de manter a concentração por um dado período de tempo, alternando com momentos de distração.*" (ES_36)

Sendo assim, o **planejamento** prévio dos estudos e a **concentração** obtida através do uso de técnicas de gerenciamento de tempo e do uso de música para se manter o foco são hábitos em comum dos engenheiros de software. Enquanto o primeiro apresenta-se

como estratégia para diminuir as distrações, o segundo apresenta-se como um meio de se diminuir o ruído em ambientes barulhentos, comuns em empresas de desenvolvimento de software cujas estações de trabalho estão distribuídas em baias.

5. Considerações Finais

Estilos de aprendizagem têm sido comumente estudados na engenharia de software como forma de subsidiar o desenvolvimento de sistemas educacionais. No entanto, são raras as pesquisas que tratam dos estilos de aprendizagem dos próprios engenheiros de software. Neste artigo, buscamos levantar dados sobre os estilos de aprendizagem em engenheiros de software profissionais, e seu hábitos e práticas autônomo de estudo no contexto do trabalho.

Como resultados, mapeamos que os engenheiros de software têm uma tendência de preferir fontes de conhecimento seguindo os estilos K-A-V-R, respectivamente. Além disso, documentamos um conjunto hábitos de estudo do dia-a-dia dos engenheiros, sendo cursos online e vídeos as mais relatadas. Observe-se que o VARK é composto por questões que visam identificar preferências, enquanto nosso questionário buscou complementar essa informação com dados sobre de fontes de conhecimento utilizadas na prática pelos engenheiros.

Uma vez que fontes de diferentes naturezas (VARK) podem assumir estratégias completamente diferentes, aparentemente, pode-se considerar uma consistência com a maior representatividade dos estilos K e A. No entanto, nossos dados não subsidiam diretamente esta conclusão. Porém, esta informação pode ajudar a guiar a construção de materiais instrucionais de larga escala, dirigidos a engenheiros de software profissionais.

É importante notar também que esta ordem de preferência mapeada não reflete os resultados absolutos dos estilos de aprendizagem. Isto é, o fato do R ser o elemento menos preferido não significa que engenheiros de software tem dificuldade com aquela fonte. Os nossos resultados agregados também não substituem nem eliminam o valor do resultado individualizado.

Nossos dados refletem ainda uma dissonância entre as necessidades por aprendizado de engenheiros de software e a sua prática profissional. Trabalhos futuros devem buscar investigar as causas dessa dissonância, que não foram objetos desta pesquisa. Ao mesmo tempo, esses dados fornecem um importante subsídio às empresas de desenvolvimento de software para flexibilização do horário de trabalho bem como a elaboração de programas de forma a incentivar o aprendizado contínuo e útil, e consequente motivação dos engenheiros de software [FRANÇA, 2014].

Pelo fato do maior número dos resultados (90%) terem sido obtidos a partir de dois estados (Pernambuco e Paraíba), não é possível assumir que estes resultados sejam representativos da população de engenheiros de software brasileiros como um todo.

Embora existam pesquisas tais como a de Leite et. al [2010] que afirme que o nível de confiabilidade do modelo VARK seja de .85 para visual, .82 para auditivo, .84 para leitor/escritor e .77 para o cinestésico, este modelo carece de mais investigações que avaliem sua confiabilidade, em particular considerando a sua tradução para português brasileiro, que utilizamos nesta pesquisa, bem como a sua atualização para uma realidade mais contemporânea (uma vez que CDs e câmeras digitais podem não pertencer mais à realidade dos estudantes atuais). Para estudos futuros, sugerimos ainda considerar a

utilização de outros modelos de aprendizagem amplamente difundidos, tal como o de Feldman e Solomon (vide Coffield [2004]) para habilitar análises comparativas com os resultados da presente pesquisa.

Referências

BOYDE, M. et al. Learning style and learning needs of heart failure patients (The Need2Know-HF patient study). European Journal of Cardiovascular Nursing, v. 8, n. 5, p. 316–322, 2009.

BRAAKHUIS, A. et al. A Comparison between Learning Style Preferences, Gender, Sport and Achievement in Elite Team Sport Athletes. Sports, v. 3, n. 4, p. 325–334, 2015.

BROWN, T.; COSGRIFF, T.; FRENCH, G. Learning style preferences of occupational therapy, physiotherapy and speech pathology students: A comparative study. The Internet Journal of Allied Health Sciences and Practice, v. 6, n. 3, p. 1–12, 2008.

CAPRETZ, L. F. Clues on Software Engineers' Learning Styles. International Journal of Computing & Information Sciences. Vol. 4, No. 1, April 2006.

CIRILLO, F. The Pomodoro Technique: Do More and Have Fun with Time Management, Third Edition, FC Garage, 2013..

COFFIELD, F. et al. Learning styles and pedagogy in post-16 learning A systematic and critical review. Learning and Skills Research Centre, p. 84, 2004.

CREDÉ, M., and KUNCEL, N. R. Study Habits, Skills, and Attitudes: The Third Pillar Supporting Collegiate Academic Performance. Perspectives on Psychological Science, v. 3, n. 6, p. 425-453, 2008.

CRUZES, D. S.; DYBA, T. Recommended Steps for Thematic Synthesis in Software Engineering. International Symposium on Empirical Software Engineering and Measurement. 2011.

CURY, H. Estilos de Aprendizagem de Alunos de Engenharia. Anais do Congresso da Associação Brasileira de Educação de Engenharia. 2000.

DE BELLO, T. C. Comparison of Eleven Major Learning Styles Models: Variables, Appropriate Populations, Validity of Instrumentation, and the Research Behind Them. Journal of Reading, Writing, and Learning Disabilities International, v. 6, n. 3, p. 203–222, 1990.

DUNN, J. L. Using Learning Preferences to Improve Coaching and Athletic Performance. Journal of Physical Education, Recreation, & Dance, v. 80, n. 3, p. 1–60, 2009.

DUNN, R. et al. Learning Style Researchers Define Differences Differently. Educational Leadership, v. 38, n. 5, p. 372–376, 1981.

FELDER, R.; SILVERMAN, L. Learning and teaching styles in engineering education. Engineering education, v. 78, n. June, p. 674–681, 1988.

FELDER, R.; SILVERMAN, L.; SOLOMON, B. Index of Learning Styles (ILS). Skynet.Ie, p. 1–10, 2000.

FLEMING, N. D.; MILLS, C. Not another inventory, rather a catalyst for reflection. To improve the academy, v. 11, n. 1, p. 137, 1992.

FRANÇA, C. A Theory of Motivation and Satisfaction of Software Engineers. Tese de Doutorado apresentada ao Programa de Pós Graduação em Ciência da Computação do Centro de Informática, Universidade Federal de Pernambuco, 2014.

FRANZONI, A. L.; ASSAR, S. Student learning styles adaptation method based on teaching strategies and electronic media. Educational Technology & Society, v. 12, p. 15–29, 2009.

GUERRA, S. R. Esutod da Aplicação de um Experimento Remoto para Apoio ao Ensino da Lei de Hooke em Alunos do Ensino Técnico. Trabalho de Conclusão de Curso de Bacharel em Tecnologias da Informação e Comunicação, submetido à Universidade Federal de Santa Catarina. 2014.

GELLER, M. Educação a Distância e Estilos Cognitivos: Construindo um novo olhar sobre os ambientes virtuais. Tese de Doutorado apresentada ao Programa de Pós-Graduação em Informática na Educação da Universidade Federal do Rio Grande do Sul. 2004.

HUANG, R. H., and SHIH, Y. N. Effects of background music on concentration of workers. Work, v. 38, n. 4, p. 383-387, 2011.

ADÁN-COELHO, J. M. et al. Conflito Sócio-Cognitivo E Estilos De Aprendizagem Na Formação De Grupos Para O Aprendizado Colaborativo De Programação De Computadores. Rbie, v. 16, n. 3, p. 9–20, 2008.

KATSIOLOUDIS, P.; FANTZ, T. A Comparative Analysis of Preferred Learning and Teaching Styles for Engineering, Industrial, and Technology Education Students and Faculty. Journal of Technology Education, v. 23, n. 2, p. 61–69, 2012.

KOLB, D. A. Experiential Learning: Experience as The Source of Learning and Development. Prentice Hall, Inc., n. 1984, p. 20–38, 1984.

MASSEY JR., F. The Kolmogorov-Smirnov Test for Goodness of Fit. Journal of the American Statistical Association. Volume 46, Issue 253, 1951.

LEITE, W. L., SVINICKI, M. & SHI, Y. (2010). Attempted Validation of the Scores of the VARK: Learning Styles Inventory With Multitrait-Multimethod Confirmatory Factor Analysis Models. Educational and Psychological Measurement. 70, 323-339.

MARCONI, Marina de Andrade. LAKATOS, Eva Maria; Fundamentos de metodologia científica. 5. ed. São Paulo: Atlas, 2003.

MARCY, V. Adult Learning Styles: How the VARK © learning style inventory can be used to improve student learning. Journal of the Association of Physician Assistant Programs, v. 12, n. 2, p. 1–5, 2001.

MOAZENI, S.; POURMOHAMMADI, H. Smart teaching quantitative topics through the VARK learning styles model. ISEC 2013 - 3rd IEEE Integrated STEM Education Conference, 2013.

RICHARDS, G. P. et al. An Examination of Learning Styles and its Impact on Curriculum Development. American Society for Engineering Education, 2011.

RIDING, R.; CHEEMA, I. Educational Psychology : An International Journal of Experimental Cognitive Styles — an overview and integration. Educational

Psychology: An International Journal of Experimental Educational Psychology, v. 11, n. 3-4, p. 37–41, 1991.

ROBERTSON, L. et al. Learning styles and fieldwork education : Students ' perspectives. New Zealand Journal of Occupational Therapy, v. 58, n. 1, p. 36–40, 2011.

SAEED, N.; YANG, Y.; SINNAPPAN, S. Emerging web technologies in higher education: A case of incorporating blogs, podcasts and social bookmarks in a web programming course based on students' learning styles and technology preferences. Educational Technology and Society, v. 12, n. 4, p. 98–109, 2009.

THOMAS, L.; RATCLIFFE, M.; WOODBURY, J.; JARMAN, E. Learning styles and performance in the introductory programming sequence. Proceedings of the 33rd SIGCSE technical symposium on Computer science education. 2002.

VALENTE, J. A. Questão do Software: parâmetros para o desenvolvimento de software educativo. Núcleo de Informática Aplicada à Educação Universidade Estadual de Campinas (NIED) – Memo Nº 24. 1989.

WILCOXON, F. Individual comparisons by ranking methods. Biometrics bulletin, 1945.

WITKIN, H. A. et al. Field-dependent and field-independent cognitive styels and their educational implications. ETS Research Bulletin Series, v. 1975, n. 2, p. 1–64, 1975.

Apêndice A –Pesquisa sobre Aprendizagem de Engenheiros de Software

Apresentação

Olá, você foi convidado(a) para participar como voluntário em uma pesquisa sobre aprendizagem de engenheiros de software. O objetivo desta pesquisa é mapear um conjunto de características de engenheiros de software relacionadas ao seu perfil de aprendizagem. Esta pesquisa é de natureza acadêmica e está sendo conduzida por alunos do Mestrado Profissional em Engenharia de Software do CESAR, e com a cooperação de pesquisadores da UFRPE e UFPE. Para mais informações sobre o Mestrado acesse: http://www.cesar.edu.br. Os dados fornecidos serão publicados apenas de forma agregada e em eventos de natureza científica. Por razões de ética e privacidade, a identificação do respondente nesta pesquisa é opcional. No entanto, mesmo em optando por identificar-se, garantimos que qualquer informação ou elemento que possa de qualquer forma identificar o participante será mantida em sigilo, e tampouco os dados serão divulgados individualmente, a fim de preservar a identidade dos participantes. Ao participante também é reservado o direito de retirar o seu consentimento a qualquer momento, mesmo após o preenchimento dos dados, sem precisar justificar, e sem que isto leve a qualquer prejuízo em sua relação com os pesquisadores. Em caso de dúvida, entrar em contato com a equipe de pesquisadores pelos emails abaixo. Ao final desta pesquisa, a síntese dos resultados será enviada para todos os participantes que preencherem o campo de E-Mail.

Estimamos que o questionário levará entre 20 e 30 minutos para ser preenchido, e o questionário permanecerá aberto para respostas apenas até o dia 15/05/2016.

Agradecemos a sua contribuição!

Os Autores

Parte I – Identificação e Dados Demográficos
Nome: _____
E-Mail (opcional): _____
Qual o seu gênero? ( ) Masculino ( ) Feminino
Ano de nascimento?: _____
Cidade onde reside?: _____
Estado: _____
Renda mensal familiar?: _____
Qual seu nível de instrução?: _____
Qual sua área de formação?
( ) Ciência da Computação, Engenharia da Computação, Sistemas de Informação , ou similares
( ) Outros
Quantos anos você possui de experiência profissional na área?: _____
Qual o tipo de empresa em que você trabalha atualmente? ( ) Pública ( ) Privada ( ) Própria ( ) Nenhum

Parte II - Hábitos de Estudo
Você acha importante o profissional aprender coisas novas independentemente das tecnologias que utiliza atualmente no trabalho?
Nunca 1 2 3 4 5 Sempre

Dentro do seu horário de trabalho, você tem a oportunidade de dedicar tempo para aprender coisas novas? *
Nunca 1 2 3 4 5 Sempre

Você costuma a reservar - fora do horário de trabalho - algumas horas diariamente para aprender coisas novas? *
Nunca 1 2 3 4 5 Sempre

As coisas novas que você decide aprender têm relação direta com as tarefas que você executa atualmente no horário de trabalho? *
Nunca 1 2 3 4 5 Sempre

Conte-nos um pouco mais sobre os seus hábitos de estudo: _____

UMA PROPOSTA DE CURSO DE GRADUAÇÃO EM GESTÃO DE PROJETOS NO CONTEXTO DA EDUCAÇÃO SUPERIOR BRASILEIRA

Vitor Abílio Sobral Dias Afonso, Lilian Maria Gonçalves, Jefferson Ferreira Barbosa, Hermano Perrelli de Moura
Centro de Informática – Universidade Federal de Pernambuco (UFPE)
vitor.fip@gmail.com, lilian.goncalves@ifmt.edu.br, jfb2@cin.ufpe.br, hermano@cin.ufpe.br

Resumo. *A área de Gestão de Projetos (GP) cresce exponencialmente e, com isso, cria oportunidades de trabalho. Junto a essa expansão surge a necessidade de profissionais atualizados e conscientes da realidade, com habilidades e competências renovadas. Por outro lado, em um cenário globalizado e cada vez mais competitivo, as organizações precisam buscar profissionais qualificados e experientes para gerir seus projetos. Nessa direção, a ampliação do Ensino Superior no Brasil trouxe possibilidades de acesso à graduação e diversificou a oferta de novos cursos e modalidades. Assim, o presente artigo objetiva discutir a criação de um Curso Superior de Tecnologia (CST) em Gestão de Projetos através da construção coletiva de uma proposição curricular. Para tanto, este estudo foi embasado em pesquisa qualitativa, tendo como métodos a pesquisa bibliográfica e a pesquisa de campo.*

Abstract. *The area of Project Management (PM) grows exponentially thereby creating job opportunities. Along with this expansion comes the need for qualified professionals with a an awareness of their project environment, with skills and competencies renewed. Moreover, in a global scenario and increasingly competitive, companies need to seek qualified and experienced professionals to manage their projects. In this direction, the expansion of higher education in Brazil brought opportunities for access to undergraduate and diversified offering of new courses and modalities. Thus, this research aims discuss to create a Course of Technology (TC) in Project Management through the collective construction of a curriculum proposition. Therefore, this study was based on qualitative research, with the bibliographic research methods and field.*

1. Introdução

A Engenharia de Software é uma ciência que estuda processos de desenvolvimento de software contendo atividades, parcialmente ordenadas, com a finalidade de obter um produto de software de qualidade.

Educacionalmente, tais atividades vêm ganhando mais espaço, principalmente em ambiente acadêmico, no tocante à necessidade pontual de expansão de seus conhecimentos, através do aumento de sua carga horária e da criação e aplicação de novas disciplinas e cursos advindos desta área.

Dentro dessa perspectiva, a Gestão de Projetos (GP) é uma das atividades que mais cresce na atualidade e, saber da existência de cursos superiores específicos fora do país fez surgir sucessivas inquietações quanto à necessidade de contribuir para as áreas de gestão de projetos e da educação brasileira. Tais inquietações acometem vários profissionais da área com responsabilidade de capacitar, inovar e expandir mercado.

Para tanto, torna-se necessária a participação de profissionais da área: professores universitários, consultores e gerentes de projetos dispersos pelo Brasil, trazendo consigo seus regionalismos, conhecimentos especializados e formas de trabalho e visão distribuídas. Os mesmos foram interpelados por meio de questionário semiestruturado e posteriores análises que contribuíram para a elaboração da proposta. Outrossim, cursos em áreas afins, tais como: engenharia, tecnologia e gestão, bem como, diretrizes curriculares específicas para o Curso Superior de Gestão de Projetos existentes no exterior, também auxiliaram para o desenrolar deste trabalho.

Atualmente, o processo de administrar projetos tem sido tarefa árdua em diversas áreas do conhecimento, com ênfase no mercado de Tecnologia da Informação, pois as mudanças nas formas de gerir junto à necessidade de competir de forma global têm acontecido de forma bastante rápida. Assim, a presente pesquisa apresenta como motivação um ponto muito importante nesse âmbito: a Educação em Gestão de Projetos.

Dado o exposto, a área específica de Gestão de Projetos tem crescido exponencialmente, aumentando seu leque de atribuições e seu panorama de visibilidade. Dessa forma, o desenvolvimento do curso de graduação em Gestão de Projetos é ratificado pela necessidade de profissionais especializados por meio de bases mais sólidas e competentes e, assim, tornando-se um desafio para a área.

A esse respeito, Rigolon (2012) afirma que há um crescimento nos cursos de capacitação em GP, com enfoque maior na memorização dos termos e nomenclaturas de ferramentas e técnicas do que no uso e nas funções. Coloca ainda, que há uma grande quantidade de profissionais certificados, porém sem habilidade e competência necessária para liderar equipes e fazer gestão eficiente das atividades de um projeto.

Ainda segundo Rigolon (2012), sem habilidades apropriadas para os desafios de um projeto, esses profissionais, mesmo especializados, seguem despreparados para executar suas atividades nos ambientes corporativos. O aprofundamento e a maturidade unidos à experiência dos profissionais no conhecimento e utilização contribuem decisivamente para o sucesso da área. Nesse sentido, tem-se que ampliar e qualificar melhor os cursos, relacionando mais detalhadamente teoria e prática.

Deste modo, a problemática gira em torno da existência de vários cursos ineficazes e/ou incompletos na área de GP no Brasil e, consequentemente, de um grande número de profissionais despreparados para o mercado atual, em uma realidade que se fazem necessárias formalização, padronização, e amplitude da ciência, promovendo a geração de conhecimento e abrangência de diversas áreas e, sanando problemas de carências e lacunas descobertas, devido à minimalização do ensino em que se encontram os cursos na atualidade.

Entretanto, como se pode propor essa criação para o ensino superior brasileiro? Qual o quadro teórico relacionado à temática no tocante a currículo, educação superior e gestão de projetos? Como profissionais e docentes da área se manifestam sobre a necessidade e possibilidade de criação?

Assim, o presente artigo resulta de pesquisa que teve como objetivo geral discutir a proposição de um curso de graduação em GP no Brasil, por meio da análise do contexto mundial da educação em gestão de projetos, da realidade brasileira da educação superior, considerando a diversidade dos cursos existentes e suas aplicações e, da proposição de um currículo na área de gestão de projetos partindo do conhecimento da variedade dos projetos, tendências e contribuições de profissionais da área. Na Seção 2 é apresentado o referencial teórico, na Seção 3 é apresentada a metodologia. Nas seções seguintes são apresentados os resultados e as considerações finais do estudo.

2. Referencial Teórico

A presente pesquisa tem o intuito de buscar cursos de graduação em Gestão de Projetos disponibilizados pelo mundo e, com isso, observar o cenário internacional quanto à área em evidência. Desse modo, países como: Inglaterra, Alemanha, Singapura, Espanha e outros, apresentaram-se como detentores de cursos de GP, porém quando efetivamente pesquisados, todos apontaram para a pós-graduação e certificação em GP e, portanto, até o momento, não havendo cursos de graduação em GP.

A partir disso, foi mostrado como cenário de pesquisa os Estados Unidos da América (EUA). Este país mostrou-se mais avançado no tocante à Gestão de Projetos do que muitas partes do mundo, dentre os motivos o fato de os EUA serem um dos precursores da área (KERZNER, 2006, pp. 380-381).

Um ponto importante da pesquisa ressalta que em todos os cursos a ciência da Gestão de Projetos está relacionada a uma área afim, possibilitando e dando suporte à vertente de curso de bacharelado, com no mínimo quatro anos para conclusão.

Outro ponto significativo reside no modelo de ensino comum nos EUA, tendo em vista que se é dado todo o alicerce em uma área maior e, posteriormente (penúltimo e último ano ou somente o último), habilitar o graduando para a área de concentração.

Nesta direção, Melo e Luz (2005) comentam que os cursos de graduação preparam para carreiras acadêmicas ou profissionais estando vinculados a conselhos ou não. Os graus que são conferidos nos cursos são de Bacharel, Licenciado, Tecnólogo ou algo específico referente à profissão, conferindo uma ou mais habilitações.

Adicionando a isto, o Ministério da Educação e Cultura - MEC preconiza que o Curso de Bacharelado "confere ao diplomado competências em determinado campo do saber para o exercício de atividade acadêmica ou profissional". Cursos de Licenciatura habilitam o indivíduo para o exercício do ensino na educação básica, enfatizando os aspectos pedagógicos. Por fim, os Cursos de Tecnologia possuem tempo reduzido e assim focalizam na concentração e, de acordo com o MEC, igualmente concede diploma aos egressos.

Sob este ângulo de visão, os anseios da sociedade brasileira obtiveram respostas por parte das instituições de ensino superior referentes aos cursos de tecnologia, em virtude do fato de que as inovações tecnológicas vêm causando intensas alterações no modus operandi, na força e perfil do trabalhador contemporâneo. Numa época em que o mercado anseia por profissionais qualificados e, que não pode esperar por muito tempo, os cursos de tecnólogos foram criados de modo a corresponder a essa demanda por preparação Prado (*apud* GOMES e OLIVEIRA, 2006, pp. 3-4).

As práticas de GP são constituídas por uma série de processos, métodos e ferramentas para o alcance dos intentos do projeto. Elas devem abarcar um roteiro para o manuseio laboral do projeto, sendo necessária a interligação de comunicação e de ideologias entre os que atuarão no referido gerenciamento (KERZNER, 2001).

Cada empresa ou projeto requer um modelo de gerenciamento a ser empregado, de acordo com as minudências que permeiam os anseios da mesma. Nessa visão, faz-se necessário ratificar o que já fora exposto através das palavras de Charvat (2003) sobre o que se trata a gestão de projetos onde é colocado que metodologia é um conjunto de orientações e princípios que podem ser adaptados e aplicados em uma situação específica, que em ambientes, é conhecido como uma lista de coisas a fazer. Esta metodologia pode ser uma abordagem específica, modelos, formulários, *checklists*, sendo usado durante todo o ciclo de vida do projeto.

Para se definir o tipo de gestão a ser adotada por uma empresa, vários critérios são levados em consideração, levando-se em conta as peculiaridades do caso concreto. Além disso, todos os recursos disponíveis pela empresa definirão a viabilidade de se

atingir os objetivos da empresa ou não, bem como, o norteamento administrativo para alcança-lo ou, simplesmente, redefinir os rumos a serem tomados.

As constantes mudanças sociais, tecnológicas e econômicas que as empresas vêm passando estão corroborando para que se adequem as novas demandas mercadológicas, visando se manter e se destacar na nova conjuntura que amolda a atualidade. As inovações dos produtos, serviços e processos de gestão são conformatados de modo a promover uma viabilidade competitiva que garanta o crescimento e sobrevivência das entidades organizacionais (MARTIN, 1996).

Entende-se que os projetos estejam absortos em uma imensa complexidade técnica e almejam uma relevante variedade de manobras, além de um espaço cada vez mais restrito e severo em termos de recursos. Assim sendo, novas formas de gestão de projetos se desenvolveram de modo a se adequarem às circunstâncias concernentes as peculiaridades das demandas, bem como, aos novos desafios que se impõem constantemente nos caminhos das gerências organizacionais.

3. Metodologia

A pesquisa bibliográfica é um caminho utilizado para conhecer e analisar o que já foi escrito sobre o assunto em questão, a fim de expor por meios mais confiáveis os dados que proporcionam a imprescindível credibilidade a este trabalho. A avaliação de um conteúdo bibliográfico pode vir a constituir um excelente método de abordagem de informações qualitativas (LUDCKE e ANDRÉ, 1986).

A pesquisa de campo deste trabalho demandou o levantamento de informações através de questionário virtual enviado via Googledocs[1] a profissionais e docentes dispersos pelo Brasil, que aplicam as práticas de Gestão de Projetos em suas diretrizes laborais, a saber: RO, AM, MT, DF, TO, SP, RJ, PR, PE, PB, CE e RN e, portanto, contemplando todas as regiões do país.

Para tanto, como instrumento da pesquisa foi elaborado um questionário semiestruturado contendo catorze perguntas ao todo, dessas: quatro objetivas, seis mistas e quatro subjetivas. Do total, sete pertencem ao núcleo de identificação, seis ao núcleo básico e uma ao complementar.

Uma vez concebida a primeira versão do instrumento da pesquisa, a mesma foi encaminhada para três especialistas a fim de validar o questionário, assim respectivamente: um docente e doutorando na área de GP, um gerente de projetos e mestrando na área de GP e uma docente e mestra em Educação. Seus contributos culminaram na ferramenta que orientou o desenvolvimento desta proposição.

Devido ao caráter qualitativo da pesquisa, não se definiu um número mínimo de respondentes, nem consequentemente, amostra. No entanto, o questionário foi disponibilizado entre os meses de agosto e dezembro de 2014, posteriormente começou o processo de análise dos resultados, cinquenta e um respondentes, desses 30% são docentes, 52% são profissionais da área e 18% são docentes e profissionais de GP.

Para a correta identificação das unidades dos conteúdos componentes nos vários materiais alcançados na coleta de informações, foi feito um confronto com a metodologia de abordagem sobre o assunto contida no referencial teórico. Por último, chegou-se a fase de apuração e interpretação de conteúdo, cujo resultado final se encontra na exposição das referidas informações.

Para finalizar, é válido ressaltar que a discussão da proposição curricular para um curso superior em GP é fruto direto da colaboração dos inquiridos, por meio de seus conhecimentos, opiniões, culturas e pesquisas de âmbito acadêmico como em Afonso

[1] http://bit.ly/2aQuYoF

(2013). Outra parceria significativa reside na existência de cursos de graduação em cenários internacionais, que muito cooperaram no sentido de firmar e estimular o presente e a área de educação em Gestão de Projetos.

4. Resultados

O questionário enquanto instrumento de coleta de dados foi dividido em três núcleos importantes: Identificação, Básico e Complementar. O primeiro aborda o perfil dos respondentes e seus respectivos conhecimentos na área de Gestão de Projetos, o segundo questiona a necessidade da criação do curso superior e o cenário atual e, o terceiro auxilia por meio de sugestões para a construção do referido curso.

A distribuição dos questionários definitivos ocorreu via Internet em redes sociais, instituições e indivíduos da área. Como resultado, 51 instrumentos foram preenchidos. Então, analisou-se interpretativamente, a fim de compreender, averiguar e explicar os resultados, observando as respostas claras e diretas e, o subentendido.

A fim de apurar os dados coletados para análise desta pesquisa utilizou-se um software provido de recursos estatísticos e construção de gráficos (Microsoft Excel). A revisão de literatura e a análise dos questionários aplicados a profissionais da área, constituíram elementos essenciais para a discussão sobre o CST em GP.

De acordo com Sotille (2013) a área de GP é uma área que cresce e está cheia de oportunidades, havendo milhões de vagas para gerentes de projetos ao redor do mundo.

Nesta, destaca-se a heterogeneidade dos respondentes, quanto ao cargo exercido pelos participantes da pesquisa, uma vez que 13 ou 18% são gerentes de projetos, 22 ou 29% docentes, 8 ou 11% consultores, 13 ou 17% pesquisadores e os 19 restantes ou 25% responderam "Outros", obtendo decrescentemente no geral as seguintes explicações: "Analista de TI", "Estudante" e "Supervisor", conforme a Figura 1.

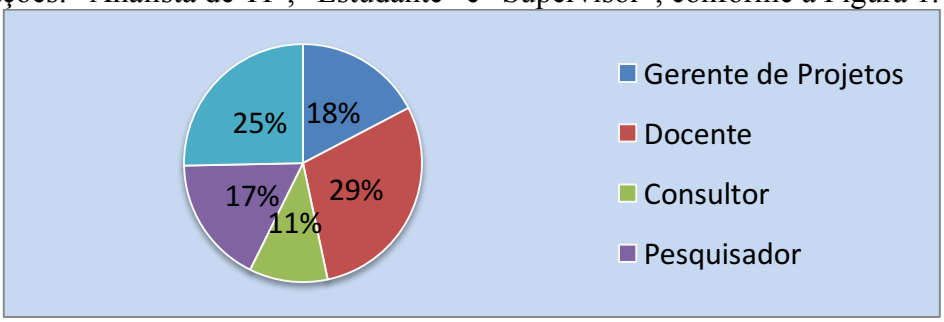


Figura 1. Cargo exercido em GP. Fonte: Pesquisador do estudo, 2015.

Sob esta ótica Nijhuis (2012) diz que os cursos de Engenharia e Construção, além de Tecnologia da Informação e Comunicação (TIC), ensinam o gerenciamento de projetos aos seus alunos, inclusive considera como um componente importante em uma variedade de qualificações acadêmicas de graduação.

Ainda assim, apresentou-se "apenas" e aproximadamente a metade como parcela de indagados que passou por uma disciplina relacionada em seus cursos de graduação.

Devido aos critérios de participação da pesquisa girarem em torno de conhecimento e experiência na área específica, partiu-se da premissa que todos os inquiridos deveriam possuir curso superior e que esta graduação contemplasse cursos nas áreas de Computação, Administração, Engenharias e Psicologia. Quando foi perguntado o ano de conclusão do curso, obtiveram-se obviamente diversas respostas, tendo um intervalo com limitadores entre "1992" (mais antigo) e "2012" (mais recente) e, mediana em "2007".

Ao serem perguntados se realizaram curso de pós-graduação na área, 32, ou 63%, responderam "Sim" e 19, ou 37%, "Não". Dos que realizaram o curso 20, ou 62%, concluíram e 12, ou 32%, assinalaram "Incompleto".

Sabe-se que experiência profissional é o tempo em que um indivíduo executa e/ou aperfeiçoa-se na sua área de atuação.

Complementando, Bondía (2002) diz que a experiência é tudo o que profissional vivencia em seu trabalho, considerando o que ele passa, o que acontece a ele.

A experiência está interligada com o que acontece ao profissional no tempo de serviço e assim, quando questionados sobre o tempo de experiência em GP 18 pessoas ou 35% assinalaram "até 1 ano", 15 ou 29% assinalaram "entre 1 e 3 anos", 7 ou 14% assinalaram "entre 3 e 5 anos", 9 ou 18% assinalaram "entre 5 e 10 anos" e 2 ou 4% assinalaram "acima de 10 anos", como se pode conferir na Figura 2.

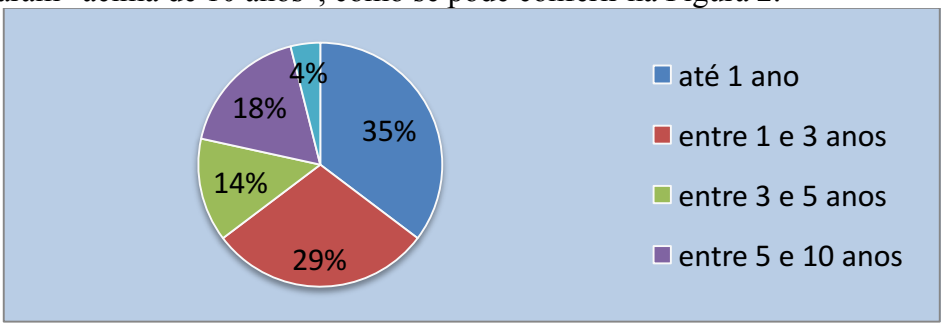


Figura 2. Tempo de experiência em GP. Fonte: Pesquisador do estudo, 2015.

Vale ressaltar, que para efeitos de análise, foram utilizadas as respostas mais incidentes e realizadas aglutinações em respostas similares.

Ao perguntar se os conhecimentos adquiridos atingem as necessidades da área 32 pessoas, ou 63%, responderam "Sim, parcialmente", 9, ou 18%, responderam "Sim, totalmente" e 10, ou 19%, responderam "Não".

Dentre as justificativas apresentadas, destacam-se três correntes: quanto à carência de prática, quanto à superficialidade da disciplina e quanto ao alinhamento de habilidades e técnicas, de acordo com essas manifestações:

"Os cursos geralmente são muito teóricos e não dão muita importância a exemplos práticos e à partilha de experiências."
"Teoria bem diferente da prática".
"Parcialmente, pois as aulas foram em sua maioria teóricas, sem aplicação dos conhecimentos adquiridos na disciplina".
"(...) fiz cursos de extensão e esses cursos não de forma clara e precisa todas as competências necessárias para atuar nessa área".
"Há necessidade de aperfeiçoamento em conhecimento para a gestão de projetos".
"(...) quanto à formação os conhecimentos foram mínimos, um pouco na engenharia de software, não houve aprofundamento ou abordagem específica".

Com um cenário mercadológico que inflama crescentemente a concorrência, conduz o indivíduo à necessidade de adaptação a ele, culminando em mais empregabilidade. A este respeito, Almeida (2006) discorre que as oportunidades antes eram basicamente na indústria. Conforme o desenvolvimento do mercado, as pessoas começaram a buscar seu constante desenvolvimento de habilidades e competências, conhecida como empregabilidade, agregadas por conhecimentos específicos e pela multifuncionalidade.

Ainda conforme demonstrado por Almeida (2006) as oportunidades saíram apenas da indústria e se estenderam até o setor de serviços, exigindo assim a mudança de perfil do trabalhador.

Isso se deve ao fato da elevada precisão de modernização e ciência da realidade. Diante disso, ao serem indagados sobre os conhecimentos adquiridos em função das

exigências do mercado de projetos 30, ou 59%, responderam “Sim, parcialmente”, 5, ou 10%, responderam “Sim, totalmente” e 16, ou 31%, responderam “Não”.

Dentre as justificativas apresentadas, destacam-se duas correntes: quanto ao fator tempo que impossibilita formação direcionada e quanto à necessidade de atualização contínua, mudanças constantes e geração de situações novas, de acordo com essas manifestações:

“Necessito de tempo maior para me aprofundar mais com definições, guias e ferramentas”.
"Acredito que sem conhecimentos especializados, adquiridos com maior tempo, ainda possuo muitas deficiências em relação a profissão”.
“Acho que precisaria de uma formação mais direcionada. Pelo menos uma disciplina na graduação ou uma especialização, se fosse o caso”.
“O mercado muda constantemente e as regiões também são diferentes (...)”.

É evidente que o crescimento do Ensino Superior no Brasil, possibilitou a inserção de qualquer pessoa, indistintamente, dando oportunidades de acesso a cursos de graduação e diversificando a oferta de novos cursos e modalidades. Ainda nessa linha Stallivieri (2007) afirma que através da oferta de cursos superiores, tanto de cursos profissionais, como de bacharelados e licenciatura, há o desenvolvimento de habilidades para o exercício da profissão escolhida, ingressando assim no mercado de trabalho.

Nessa perspectiva, foi questionado aos participantes sobre o principal argumento de defesa para a oferta do referido curso 17 respondentes ou 33% assinalaram “Exploração do mercado como alternativa, em função da área”, 7 ou 14% assinalaram “Necessidade do curso em razão de sua viabilidade”, 6 ou 12% assinalaram “Quantidade de pessoas interessadas no setor”, igualmente 6 ou 12% assinalaram ”Benefícios socioeconômicos do curso”, 2 ou 4% assinalaram “Outro” complementando com “Multidisciplinaridade” e 13 ou 25% assinalaram “Não defendo a oferta do curso”, conforme consta na Figura 3.

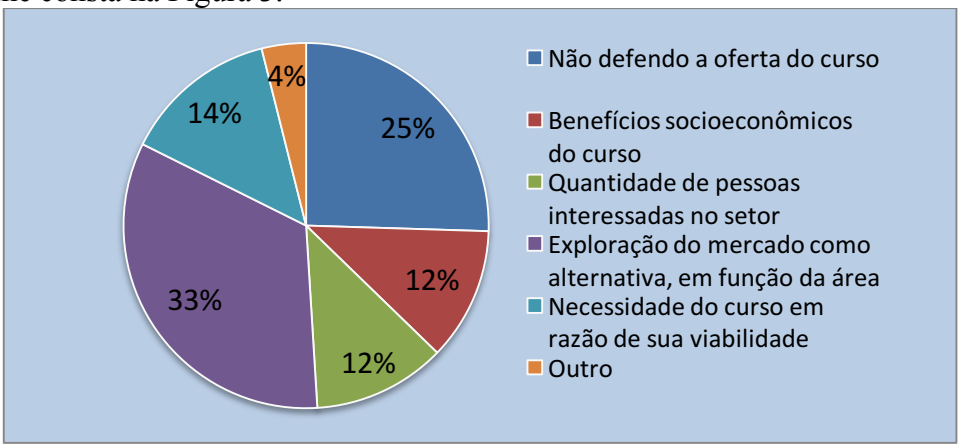


Figura 3. Argumento que defende a oferta do curso. Fonte: Pesquisador do estudo, 2015.

Ao todo 75% dos inquiridos mostraram-se defensores da criação e abertura do curso, conforme as explanações a seguir:

“O mercado necessita, uma vez que a área cresce”.
“Gestão de Projetos se pensarmos bem não se limita, relaciona-se ao todo de uma empresa. Abrange a totalidade não só uma única área”.
“Acho que é uma necessidade do mercado e precisamos preparar a mão de obra para isso”.
“Acredito que o conhecimento em Gestão de Projetos seja complementar a várias outras áreas”.
“Defendo a criação do curso, pois trata-se de uma área muito ampla e

multidisciplinar".
"A possibilidade de popularizar o assunto de forma que as atividades possam ser mais planejadas, gerenciadas e medidas. O que atualmente observa-se que há poucos profissionais que trabalham orientados a projetos e processos".
"Acredito que sempre a razão de oferta de um curso deve vir da expansão do conhecimento e sua aplicação e dos benefícios gerados por isso. Profissionais mais competentes traz benefício para toda a sociedade".
"Acredito que o mercado já está precisando de pessoas especialistas na área, e os cursos de pós-graduação não estão suprindo a necessidade".

A minoria composta por 25% dos restantes não defende a oferta do curso em questão, conforme as explanações a seguir:

"A área, no meu entender, não justifica um curso de graduação específico. Creio, no entanto, que a área de GP demanda uma maior atenção e maior espaço dentro dos cursos de graduação existentes".
"Não vejo necessidade de um curso de graduação completo sobre Gestão de Projetos. Especialização seria mais adequado".
"Creio que seja interessante uma base em uma determinada área que essa possa ser utilizada como área de atuação do gerente de projetos".

Concordando e discordando respectivamente Paes, Araújo e Kamimura (2012) comentam que houve um grande aumento de demanda do mercado, com volumes em franca ascensão e com esse aumento, aumentou também a exigência de negócio com margens e resultados financeiros. Com isso há também o aumento de metodologias, cada qual com suas vantagens e desvantagens e a partir disso, aumentando também a exigência na educação desses profissionais que estão sendo inseridos no mercado pelas instituições de ensino.

Outra razão para esse pensamento está no sentido de não enxergar na área de gestão de projetos a vertente multiprofissional e com isso, extensa e variada. Em conformidade, Michalick (2013) refuta o entendimento de que somente administradores podem atuar na área. E continua apresentando que a área de Tecnologia da Informação (TI) provém mais profissionais para a área de Gerenciamento de Projetos e a partir disso, outras áreas como a área de engenharia e construção, se interessou a se inserir neste mercado, tendo assim diversidade na procura e na formação desses profissionais.

Para finalizar esta discussão, todo e qualquer projeto depende efetivamente dos conhecimentos técnicos, das competências de gestão e dos conhecimentos do negócio por parte do gerente de projetos.

Ao questionar se o respondente considera necessária a oferta do curso 21 ou 41% assinalam "Sim", 21 ou 41% assinalaram "Talvez" e 9 ou 18% assinalaram "Não", de acordo com a Figura 4.

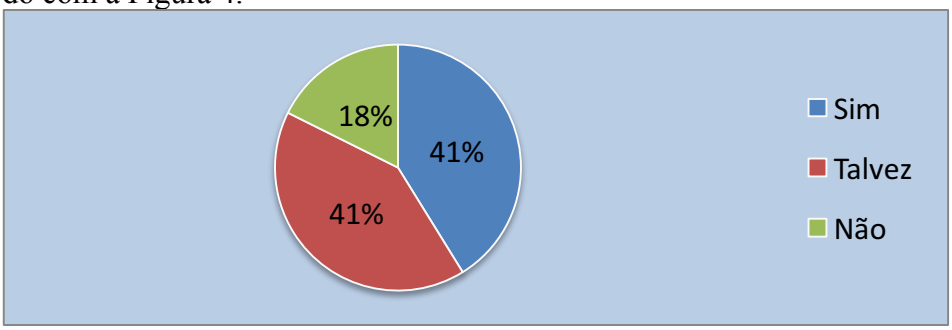


Figura 4. Necessidade de oferta do curso superior. Fonte: Pesquisador do estudo, 2015.

Ao todo 82% dos inquiridos mostraram parcial ou total certeza da necessidade de oferta do curso, conforme as explanações a seguir:

"É necessário devido ao núcleo de conhecimentos e a ausência em complementações".
"(...) cada vez mais o mercado necessita de profissionais preparados para a Gestão de Projetos, nesse sentido um curso superior na área só iria ajudar a melhorar a qualidade dos profissionais disponíveis ao mercado de trabalho".
"É uma área promissora devido ao crescimento do mercado, das tecnologias, concorrência acirrada, etc".
"Defendo a criação do curso, pois se trata de uma área muito ampla e multidisciplinar".

A parcela restante (18%) não julga necessária a oferta curso em questão, conforma as explanações a seguir:

"Por ser uma subárea de administração e ser uma possibilidade de atuação do administrador".
"Não vejo necessidade de um curso de graduação completo sobre Gestão de Projetos. Especialização seria mais adequado".
"Acho que diversos cursos trabalham esta temática e acho que não é um tópico que justifique uma graduação".

Em relação a suas opiniões quanto ao impacto de não se ter esse curso no Brasil, os respondentes comentaram:

"Continuar na mesma e não efetivar/implantar de fato planejamento e gestão".
"O impacto maior em minha opinião está na qualidade dos profissionais que estão saindo dos cursos de Pós-graduação em Gestão de Projetos. Um curso superior na área melhoraria a qualidade desse profissional".
"Capacitação específica, alunos tem que buscar outras áreas e se aperfeiçoar sozinho".
"Projetos que não funcionam como deveriam, projetos inacabados, projetos que estouram orçamentos, entre outros".

Quando inquiridos sobre o porquê, em suas opiniões, do Brasil ainda não ter oferecido tal curso, os respondentes comentaram:

"Porque o Brasil é muito tradicionalista e tem medo de arriscar coisas novas para melhoria".
"Por falta de uma perspectiva futura de crescimento da GP no Brasil (...)".
"Por causa da burocracia dos órgãos competentes para a criação de novos cursos".
"Talvez por causa da falta de iniciativa do mercado ou do colegiado das universidades que podem decidir essas coisas".
"Mercado de trabalho limitado e a oferta de cursos de aperfeiçoamento e especializações".
"Pensamento de que esta necessidade está sendo suprida em outros cursos não tão específicos".

A grande maioria dos respondentes (44 ou 86,2%) foi taxativa em assinalar muitos ou todos os campos dispostos para receberem maior carga horária, com ênfase em Escopo, Tempo, Custos, Riscos, Qualidade e Comunicação. O restante, representando 7 ou 13,8%, assinalaram apenas um ou dois campos, enfocando Escopo, Qualidade, RH e Comunicação.

Vale ressaltar, que esta questão de múltipla escolha possibilitou a marcação de mais de uma Área do Conhecimento, incluindo todas. Isso fica evidente ao confrontar a quantidade de respondentes da pesquisa (51) com o número de participantes por área escolhida.

Verifica-se que "Tempo" é a área de conhecimento mais focada e "Aquisições" a menos focada. Mesmo assim, percebe-se que todas as áreas apontadas na questão, segundo os respondentes, são essenciais, não só porque a diferença numérica entre as

escolhas foi mínima, como também o equilíbrio dos participantes em assinalar várias opções, estabelecendo a ideia de interdependência devido à mesclagem das áreas.

Esta pergunta também solicitou justificativa referente às escolhas, surgindo assim as seguintes indagações:

"Todas igualmente importantes e interdependentes".
"Acredito ser os pilares para se ter uma boa gestão".
"São, provavelmente, as áreas mais exigidas pelo mercado".
"Acredito que essas áreas deverão ser mais evidenciadas, pois implica em resultado mais eficaz".

O PMI (2013) explica que são componentes essenciais e aceitos da área de Gestão de Projetos, apontando o conhecimento praticado à maioria dos projetos e havendo consenso quanto ao seu valor. Explica ainda que possui inovações para todas as áreas de conhecimento que envolve projetos: tempo, escopo, custos, comunicação, RH qualidade, riscos, aquisições, integração e partes interessadas.

Ao questionar com relação a quais disciplinas seriam necessárias no currículo do curso uma parcela relevante de inquiridos (42 ou 82,3%) foi incisiva em apontar muitas ou todas as disciplinas elencadas para inserção na proposta curricular, com ênfase em Gestão Estratégica, Negociação e Gestão de Conflitos, Escritório de Projetos e Processos, Gestão Empreendedora e Gestão de Operações. Os demais, representando 9 ou 17,7%, apontaram apenas um ou dois campos, enfocando Gestão Estratégica, Negociação e Gestão de Conflitos, Escritório de Projetos e Processos.

Um total de 7 ou 13,7% dos respondentes assinalaram "Outros", obtendo decrescentemente no geral as seguintes explicações: "Psicologia", "Gestão de Programas e Portfólios", "Análise de Sistemas", "Modelos de Melhoria de Processos", "Teoria Geral da Administração" e "Sociologia".

Lembrando que esta pergunta de múltipla escolha possibilitou a marcação de mais de um campo de conhecimento correlato, incluindo todos. Isso fica evidente ao confrontar a quantidade de respondentes da pesquisa (51) com o número de participantes por disciplina escolhida.

Verifica-se que "Gestão Estratégica" é a disciplina mais focada e "Gestão de Marketing" a menos focada. Entretanto, percebe-se que todas as disciplinas apresentadas, segundo os respondentes, são significantes e direcionadas á área de Gestão de Projetos, tendo em vista que a diferença numérica entre as escolhas foi mínima e por considerarem que as mesmas fornecem fundamentação à profissão.

Esta questão também solicitou justificativa referente às escolhas, trazendo os seguintes argumentos:

"Todas participam dos núcleos de cursos de gestão. A disciplina embasará teoricamente".
"Abranger a base de uma empresa e seus projetos".
"(...) em se tratando de curso superior possam agregar valor a formação do Gerente de Projetos".
"Estão mais proximamente relacionadas com a área de GP".

Por fim, a última pergunta do questionário aplicado a profissionais da área solicitou destes sugestões e complementações no tocante à construção de um percurso curricular para o curso de Graduação em Gestão de Projetos. Dessa forma, surgiram as seguintes contribuições:

"Acredito que o conteúdo cabe em um curso tecnológico (2,5 ou 3 anos) mas é pequeno para uma graduação (bacharelado) de 4 ou 5 anos. Agora que tem espaço para os cursos tecnológicos e sequenciais, acredito que seja o momento oportuno para esse

curso".
"Abranger sua totalidade que o profissional consiga ser multidisciplinar".
"O curso superior em Gestão de Projetos deve ter também disciplinas voltadas a Gestão de Projetos em instituições públicas. Essas instituições tem uma visão de projetos diferente da visão de projetos para instituições privadas. Deve-se atentar para esse aspecto uma vez que o Brasil tem tradição em "estourar" os prazos e orçamentos dos seus projetos em instituições públicas".
"Teoria da Ação Comunicativa em Gestão de Projetos com foco nos aspectos humanos".
"Oratória e habilidades interpessoais, fortalecendo ainda mais a questão de liderança".
"O curso deve buscar estar sempre em contato com o mercado de trabalho para atender suas necessidades".

Nos dias de hoje, os projetos curriculares universitários devem ser orientados a mudanças, capacitando o homem para confrontar o mercado, estar preparado para ele e propor desafios. A necessidade aponta para profissionais com visão crítica, aptos a pesquisas e dispostos à Educação Continuada, sempre com vistas ao melhoramento e a constante atualização.

Portanto, a criação do Curso Superior de Tecnologia em Gestão de Projetos visa formar / capacitar efetivamente indivíduos com toda a amplitude e detalhamento que a área exige, caminhando em sincronia com o trinômio: Ciência – Tecnologia – Inovação.

5. Considerações Finais

Propor um currículo inovador para cursos de graduação em qualquer área é algo que implica pesquisas diversas no tocante a várias dimensões: viabilidade e demanda na sociedade, processos de ensino e aprendizagem, metodologias, recursos e abordagens de ensino e insumos para a estruturação do curso em si. E quando a oferta é inédita quanto ao objeto e cenário, como é o caso de um CST em GP no Brasil, essa tarefa torna-se mais complexa, por ser terra desconhecida e, portanto, mais árdua e distante.

Quanto à colaboração dos respondentes fica a constatação da grande maioria que declara ser positiva a implantação do CST em GP dado a sua viabilidade, ao crescente número de interessados e à demanda nacional. O fato de haver lacunas abertas no mercado brasileiro referentes à quantidade e qualificação de profissionais, mesmo com a existência de cursos de pós-graduação e certificações (considerados insuficientes), é um forte indicativo de que não só esta proposta é válida, como já deveria ter sido efetivada há mais tempo, tendo em vista que em outras partes do mundo já é uma promissora realidade.

Contudo, as principais contribuições deste trabalho estão relacionadas à área de Gestão de Projetos no que concerne à formalização e padronização de suas práticas enquanto Curso de Graduação no Brasil, trazendo consigo a possibilidade de mais olhares crentes e adeptos e, uma política governamental direcionada para o setor. Outro contributo reside no encorajamento de cursos em outras áreas com situações similares.

Para concluir, faz-se mister constatar que esta proposição não representa uma versão final preparada para a execução, mas o início de reflexões, discussões e melhoramentos na área a que se destina.

Referências

Afonso, V. A. S. D. (2013) "Educação em Gestão de Projetos: uma proposta para o ensino superior brasileiro". Dissertação de Mestrado. Universidade Federal de Pernambuco – CIn. Recife, Pernambuco, Brasil.

Almeida, M. G. de. (2006) **"**Pedagogia empresarial: Saberes, Práticas e Referências". Rio de Janeiro: Brasport.

Bondía, J. L. (2002) "Notas sobre a experiência e o saber de experiência". Revista Brasileira de Educação, 19, Universidade Estadual de Campinas, Campinas, p. 21.

Charvat, J. (2003) "Project Management Methodologies". John Wiley & Sons, NJ, p. 52.

Kerzner, H. (2006) "Gestão de Projetos: As melhores práticas". 2ª ed. Porto Alegre. Bookman.

Kerzner, H. (2001) "Project Management: A system approach to planning scheduling and controlling". John Wiley & Sons, 7ª ed.

Ludke, M.; André, M. E. D. A. (1986) "Pesquisa em educação: abordagens qualitativas". São Paulo: EPU.

Martin, J. (1996) "A Grande Transição". São Paulo: Futura.

Melo, P. A.; Luz, R. J. P. da. (2005) "A Formação Docente no Brasil". Instituto de Pesquisas e Estudos em Administração Universitária (INPEAU/UFSC). Florianópolis, p. 40.

Michalick, I. (2015) "Para transformar projetos em resultados são necessárias ferramentas de gestão adequadas"**.** Disponível em: <http://bit.ly/2aG2Ir6>. Acesso em: 30 abr. 2015.

Nijhuis, S**. (**2015**)** "Learning For Project Management in a Higher Education Curriculum"**.** Disponível em: <http://www.pmi.org>. Acesso em: 15 jan. 2015.

Paes, E. S.; Araújo, E. A. S.; Kamimura, Q. P. (2012) "Pós-graduação ou certificação em projetos, vantagens e desvantagens"**.** In: IV Congresso Internacional de Cooperação Universidade-Indústria, UNINDU, Taubaté, p. 10.

Project Management Institute. (2013) Um Guia do Conjunto de Conhecimentos em Gerenciamento de Projetos. 5ª Edição. Project Management Institute, Inc. Newtown Square, Pensilvânia, EUA.

Prado apud Gomes, C. G.; Oliveira, E. L. de. (2006) "Curso Superior como instrumento de inserção no mercado de trabalho regional: O caso do Norte Fluminense"**.** In: XV Encontro Nacional de Estudos Populacionais, ABEP, Caxambu – MG, pp. 3-4.

Rigolon, G. (2014) "Somente a certificação PMP garante a qualidade do Gerente de Projetos?". Disponível em: <http://bit.ly/2b7bGNr>. Acesso em: 05 out. 2014.

Sotille, M. PM Tech – Capacitação em Projetos. (2013) "Como iniciar uma carreira em Gerenciamento de Projetos". Disponível em: <http://bit.ly/2aFMbq1>. Acesso em: 20 ago. 2014.

Stallivieri, L. (2007) "O Sistema de Ensino Superior do Brasil: características, tendências e perspectivas"**.** Universidade de Caxias do Sul, Caxias do Sul.

Refinamento de Competências do Egresso do Curso de Engenharia de Software

Daltro J. Nunes[1], Marcelo H. Yamaguti[2], Ingrid Nunes[1]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Porto Alegre – RS – Brasil

[2]Faculdade de Informática – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brasil

{daltro,ingridnunes}@inf.ufrgs.br, yamaguti@pucrs.br

Resumo. *O número de cursos de graduação de Engenharia de Software cresce a cada ano no Brasil. A fim de dar orientação a estes novos cursos, ou mesmo homogeneizar os existentes, a comunidade de Engenharia de Software do Brasil trabalhou ao longo de quatro anos para consolidar um Refinamento de Competências do Egresso do Curso de Engenharia de Software. Diversas universidades participaram do trabalho, e participantes do Fórum de Educação em Engenharia de Software (FEES) contribuíram nos anos de 2013 a 2015. Neste artigo, apresentamos um relatório deste trabalho e o seu resultado.*

1. Introdução

O número de cursos de graduação de Engenharia de Software cresce a cada ano no Brasil. A fim de dar orientação a estes novos cursos, ou mesmo homogeneizar os existentes, foi iniciado um trabalho para identificar conteúdos que devem fazer parte do currículo do curso. Este trabalho consiste da dedução de conteúdos necessários para que egressos do Curso de Engenharia de Software tenham determinadas competências. As competências foram obtidas das Diretrizes Curriculares Nacionais (DCNs) para os cursos de graduação em Computação (MEC, 2012).

O Conselho Nacional de Educação (CNE) estabeleceu as bases para construção de Diretrizes Curriculares (PARECER N.º: CNE/CES 67/2003), abandonando a abordagem conteudista (adotada nos currículos mínimos) e adotando a abordagem baseada em competências (adotada nas Diretrizes Curriculares Nacionais).

A metodologia, desenvolvida pelo Prof. Daltro Nunes, foi inspirada em uma matriz, utilizada pelo INEP no projeto da prova do ENADE. Ele foi convidado pela Universidade de Goiás para discutir o Projeto Pedagógico do Curso (PPC) de Engenharia de Software na qual foi discutida a adaptação da matriz do ENADE no desenvolvimento de currículos. A metodologia para produzir conteúdos com base em refinamentos de habilidades e competências foi, entretanto, estabelecida mais tarde. Convidado para avaliar o currículo do curso de Engenharia de Software da Universidade de Brasília, Campus Gama, a metodologia foi discutida e aprimorada chegando-se a forma atual. A metodologia foi amplamente discutida e aplicada pela comunidade de Engenharia de Software do Brasil – no contexto do Fórum de Educação em Engenharia de Software (FEES), parte do Simpósio Brasileiro de Engenharia de Software (SBES), nos anos de

2013 a 2015. Alguns seminários foram realizados na PUCRS com a participação de diversos professores de várias universidades gaúchas.

Neste documento, são descritos detalhes relativos ao trabalho realizado, bem como o resultado obtido.

2. Metodologia

O trabalho realizado baseia-se na educação por competências. Em vez de se fazer uma listagem *ad hoc* de conteúdos a serem ministrados, há um questionamento inicial a respeito de quais competências o egresso do curso deve possuir, conforme determinação do CNE. Competência é a "capacidade de articular e mobilizar *conhecimentos*, *habilidades* e *atitudes*, colocando-os em ação para resolver problemas e enfrentar situações de imprevisibilidade em uma dada situação concreta de trabalho e em um determinado contexto cultural". A partir destas competências, que são refinadas em níveis de granularidade cada vez mais baixos, listam-se, ao final, conteúdos necessários para adquiri-las. O resultado é uma relação de ordem (grafo dirigido) onde os nodos terminais são conteúdos.

O ponto de partida do trabalho foram as Diretrizes Curriculares Nacionais (DCNs) para os cursos de graduação em Computação (MEC, 2012) – a adequação das diretrizes não foi questionada neste trabalho. Das DCNs, foram extraídas informações relativas ao curso de Bacharelado em Engenharia de Software. As habilidades e competências específicas do egresso de Engenharia de Software, descritas no artigo 5º, parágrafo 3º das DCNs foram consideradas como as competências do nível inicial.

Uma competência de nível N pode, no nível N+1, (i) ser refinada em duas ou mais competências de granularidade mais fina, ou (ii) ter conteúdos apropriados listados para que o estudante adquira aquela competência. Como resultado, as folhas (último nível) do refinamento de cada uma das competências oriundas das DCNs consistem de conteúdos.

Para o detalhamento dos refinamentos de competências, foi utilizada uma terminologia para evitar ambiguidade dos termos. Essa terminologia tem origem na taxonomia utilizada no *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering* da ACM/IEEE (ACM/IEEE, 2004), que consiste em conhecimento, entendimento e aplicação.

A seguir, cada um dos termos utilizados são especificados.

- **Conhecer**: lembrar do material previamente ensinado. Testa a observação e recuperação da informação, isto é, "trazer à mente a informação apropriada".
- **Compreender**: entender a informação e o significado do material apresentado. Por exemplo, ser capaz de traduzir o conhecimento a um novo contexto, interpretar fatos, comparar, contrastar, ordenar, agrupar, inferir causas, predizer consequências, etc.
- **Aplicar**: usar o material aprendido em situações novas e concretas. Por exemplo, usando informação métodos, conceitos, teorias para resolver problemas que requerem as habilidades e conhecimento apresentados.

No trabalho realizado, “Aplicar” engloba “Compreender” que por sua vez engloba “Conhecer”.

3. Ciclos de discussão

O trabalho foi realizado em vários ciclos de reuniões de grupos locais, grupos de trabalho no FEES e discussões remotas em grupos de área. Precisamente, após a reunião na UnB, que resultou em uma primeira versão dos refinamentos das competências, chegando-se a conteúdos, foram realizadas rodadas de trabalho, as quais são detalhadas a seguir.

1. Em setembro de 2013 houve uma reunião, coordenada pelo Prof. Daltro Nunes, com representantes da UFRGS, PUCRS, UNIPAMPA, UNILASALLE, na PUCRS em Porto Alegre, que refinou algumas competências com base no trabalho inicial feito.
2. Em outubro de 2013 ocorreu, no FEES 2013 em Brasília, uma sessão de refinamento com os participantes do evento, coordenada pelo Prof. Daltro Nunes e Prof. Hilmer Neri.
3. Em dezembro de 2013, houve nova reunião, coordenada pelo Prof. Daltro Nunes, com representantes da UFRGS, PUCRS, UNIPAMPA, UNILASALLE, na PUCRS em Porto Alegre, para nova sessão de refinamentos e ajustes.
4. Em janeiro de 2014 foi criado um fórum Web (https://groups.google.com/forum/?hl=pt-BR#!forum/engswpropostacurriculo) para discussão com interessados no trabalho em desenvolvimento, gerenciado pelo Prof. Marcelo Yamaguti.
5. Em maio de 2014, a PUC Minas, em Belo Horizonte, colaborou com o refinamento de competências comuns e conteúdos, relatado pela Profa. Maria Augusta Nelson.
6. Em outubro de 2014, no FEES 2014, em Maceió, houve nova sessão de refinamento com os presentes, coordenado pelo Prof. Daltro Nunes e Prof. Marcelo Yamaguti. O intuito original era de se finalizar o refinamento e consolidação de todas as competências. Entretanto, após várias discussões, conseguiu-se efetivamente consolidar apenas 2 de 14 competências. Assim, foram criados grupos de trabalhos para refinamento de cada competência, por sugestão do Prof. José Carlos Maldonado.
7. Em novembro de 2014, cada grupo relatou os seus resultados no fórum Web ou por e-mails.
8. Em abril de 2015, houve uma reunião, coordenada pelo Prof. Daltro Nunes, com representantes da UFRGS, PUCRS, UNIPAMPA, na PUCRS em Porto Alegre, para ajustes de consistência dos refinamentos gerados pelos diversos grupos.
9. Em outubro de 2015, no FEES 2015 em Belo Horizonte, houve a consolidação das competências com a participação dos presentes ao evento, com coordenação do Prof. Marcelo Yamaguti e Profa. Ingrid Nunes. Ainda

assim, houve uma das competências que não foi consolidada no evento e solicitou-se ao grupo de trabalho um parecer.

10. Em janeiro de 2016, houve a consolidação da última competência, realizada por meio de comunicações por e-mail.
11. Em abril de 2016, foi publicado no fórum Web o resultado dos refinamentos de competências do egresso de Engenharia de Software.
12. Finalmente em junho de 2016, foi encaminhado para a Comissão de Educação da SBC o refinamento de competências do egresso de Engenharia de Software.

4. Considerações finais

A partir da metodologia de trabalho descrita, obteve-se como resultado o modelo que consta no Apêndice 1. No Apêndice 2, listam-se os envolvidos no trabalho realizado durante os ciclos de discussão, inclusive com indicações dos coordenadores e participantes dos grupos de trabalho responsáveis por cada uma das competências não consolidadas no FEES 2014.

Dessa forma, este grande esforço da comunidade da Engenharia de Software do Brasil é documentado e divulgado, a fim de que sirva para ações de melhoria na educação de cursos de bacharelado em Engenharia de Software do Brasil.

A metodologia tem, entretanto, limitações. Considerando que as competências técnicas são profissionais, a aplicação da metodologia gera conteúdos que levam a estas competências. Entretanto, não gera conteúdos básicos (teoria da computação, matemática discreta, lógica, complexidade, etc.) que suportam os conteúdos profissionais.

Referências

ACM/IEEE. **Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering**. Technical Report. ACM, New York - NY, USA. 2004.

MEC. **Diretrizes Curriculares Nacionais para os cursos de graduação em Computação**. 2012. Disponível em: http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=11205-pces136-11-pdf&category_slug=julho-2012-pdf&Itemid=30192. Último acesso em: 8/8/2016.

Apêndice 1: Refinamento de Competências

Competências	Refinamento/Conteúdo	Refinamento/Conteúdo	Refinamento/Conteúdo	Refinamento/Conteúdo
1. Investigar, compreender e estruturar as características de domínios de aplicação em diversos contextos	Conhecer e analisar as características de domínios de aplicação em diversos contextos	Conteúdo: técnicas de elicitação de requisitos		
	Aplicar técnicas de estruturação das características de domínios de aplicação em diversos contextos	Conteúdos: técnicas de especificação, modelagem, verificação, validação e gerência de requisitos		
2. Compreender e aplicar processos, técnicas e procedimentos de construção de software	Conhecer os fundamentos da teoria de processos	Conteúdos: Teoria Geral de Processos (modelagem, especificação, análise e controle, adaptação)		
	Aplicar processos, técnicas e procedimentos de construção de software	Aplicar processos de construção de software	Conteúdos: Conceito de processo de software; Modelos de processo de software; representação de processo de software	
		Aplicar técnicas e procedimentos de especificação	Conteúdo: técnicas de elicitação de requisitos, técnicas de especificação, modelagem, verificação, validação e gerência de requisitos,	
		Aplicar técnicas e procedimentos de desenvolvimento	Conteúdos: princípios de projeto (*design*), projeto (*design*) de arquitetura de software, padrões, reutilização de software, projeto (*design*) detalhado, projeto (*design*) de dados, projeto (*design*) de interface com usuários e projeto (*design*) de interface com outros sistemas	
		Aplicar técnicas e procedimentos de validação, verificação e teste	Aplicar técnicas e procedimentos de validação e verificação estáticos	Conteúdos: técnicas de revisão e análise estática de artefatos de software
			Aplicar técnicas e procedimentos de validação e verificação dinâmicos	Conteúdos: técnicas de análise dinâmica de artefatos de software
		Aplicar técnicas e procedimentos de evolução	Conteúdos: refatoração, engenharia reversa, reengenharia, análise de impacto, manutenção, depuração	
3. Analisar e selecionar tecnologias adequadas para a construção de software	Aplicar tecnologias para a definição do ambiente de construção de software	Conteúdos: ferramentas e *frameworks* de desenvolvimento e de gerenciamento de configuração de software		

	Aplicar tecnologias a serem utilizadas no produto de software	Conteúdos: técnicas de programação; paradigmas de programação.	
4. Conhecer os direitos e propriedades intelectuais inerentes à produção e utilização de software	Conhecer os direitos, deveres e propriedades intelectuais inerentes à produção de software	Conteúdos: noções básicas de Direito, direito autoral, registro de software, propriedade intelectual, leis, acórdãos e instruções normativas sobre Engenharia de Software	
	Conhecer os direitos, deveres e propriedades intelectuais inerentes à utilização de software	Conteúdos: noções básicas de Direito, direito autoral, registro de software, propriedade intelectual, leis, acórdãos e instruções normativas sobre Engenharia de Software	
5. Avaliar a qualidade de sistemas de software	Entender quais são os atributos de qualidade do produto de software e sua utilidade	Conteúdo: atributos de qualidade de produto de software	
	Aplicar mecanismos de medição da qualidade do produto de software	Conteúdos: métricas de produto de software, técnicas de avaliação de produto	
	Aplicar técnicas e procedimentos de validação, verificação e teste	Aplicar técnicas e procedimentos de validação e verificação estáticos	Conteúdos: técnicas de revisão e análise estática de artefatos de software
		Aplicar técnicas e procedimentos de validação e verificação dinâmicos	Conteúdos: técnicas de análise dinâmica de artefatos de software
6. Integrar sistemas de software	Aplicar técnicas de integração de partes de um sistema	Conteúdos: ambientes de integração, ferramentas de *build*	
	Aplicar técnicas de integração de sistemas heterogêneos	Conteúdos: interoperabilidade de sistemas, *wrappers*, software como serviço, sistemas de sistemas, ecossistemas/plataformas (APIs)	
7. Gerenciar projetos de software e processos de desenvolvimento de software	Aplicar técnicas, ferramentas e práticas de gerenciamento considerando as dimensões de gestão de projetos de software	Conteúdo: conceitos básicos de gestão de projetos. Alinhamento da TI com o negócio, formas de gestão, gerenciamento de escopo, tempo, custo, qualidade, comunicação, riscos, pessoas, aquisição, integração, partes interessadas e valor de negócio, métricas de produto e de projeto.	

	Aplicar técnicas, ferramentas e práticas para gerenciamento do processo da produção, aquisição e evolução de um software	Conteúdo: gerenciamento do ciclo de vida de produção; gerenciamento do fluxo de trabalho; engenharia de produto, modelos de ciclo de vida: história e perspectivas, artefatos de software, papéis, métricas de processo de software	
	Conhecer Teorias aplicadas na Gestão da Produção de Software	Conteúdo: Teorias - da Administração Científica, das Restrições, da Complexidade, dos Jogos, do Caos, da Dinâmica de Sistemas, Geral dos Sistemas, Pensamento Sistêmico	
	Entender os elementos que compõe um sistema de produção	Entender as estratégias de Operações	Conteúdo: Cadeia de Valor, Tomada de Decisão, Alinhamento entre a estratégia de TI e estratégia de negócios
		Entender a estrutura dos processos de produção	Conteúdo: competências competitivas, estrutura do processo de bens (manufatura) e serviços (produtos de software)
8. Aplicar adequadamente normas técnicas	Entender as normas de qualidade de produto de software	Conteúdo: modelos e normas de qualidade de produto (nacionais e internacionais)	
	Entender as normas de qualidade do processo de desenvolvimento de software	Conteúdo: modelos e normas de qualidade de processo (nacionais e internacionais)	
	Aplicar conceitos de qualidade de processo para a definição de um processo de software	Conteúdo: modelos e normas de qualidade de processo (nacionais e internacionais), métricas de processo	
9. Qualificar e quantificar seu trabalho baseado em experiências e estudos experimentais	Aplicar métodos de pesquisa experimental	Conteúdo: conhecimento científico; método científico e experimental; métodos quantitativos, qualitativos e mistos de pesquisa; métodos de pesquisa e experimentação em Engenharia de Software; estudos primários e secundários; protocolos de pesquisa	
	Entender procedimentos de análise, interpretação e apresentação de resultados de estudos experimentais em ES	Conteúdo: estatísticas descritivas, teste de hipóteses, análise qualitativa, relato de estudos experimentais de Engenharia de Software	

10. Exercer múltiplas atividades relacionadas a software como: desenvolvimento, evolução, consultoria, negociação, ensino e pesquisa	Aplicar os conhecimentos adquiridos para o desenvolvimento e evolução de software	Conteúdos: práticas de laboratório no desenvolvimento e evolução de software
	Aplicar técnicas de comunicação para apresentar conhecimentos adquiridos	Conteúdos: técnicas de comunicação
	Aplicar métodos de ensino e pesquisa em ES	Conteúdos: metodologias de ensino e pesquisa
	Conhecer metodologias do trabalho de consultoria.	Conteúdos: técnicas de consultorias
	Conhecer os principais modelos e etapas do processo de negociação.	Conteúdos: técnicas de negociação
11. Conceber, aplicar e validar princípios, padrões e boas práticas no desenvolvimento de software	Aplicar os princípios, padrões e boas práticas de desenvolvimento de software	Conteúdos: princípios de ES, aplicação de padrões em ES, melhoria contínua, aplicação de gestão de conhecimento
	Conceber e validar os princípios, padrões e boas práticas de desenvolvimento de software	Conteúdo: conhecimento científico; método científico e experimental; métodos quantitativos, qualitativos e mistos de pesquisa; métodos de pesquisa e experimentação em ES; estudos primários e secundários; protocolos de pesquisa, estatísticas descritivas, teste de hipóteses, análise qualitativa, relato de estudos experimentais de ES, princípios de ES, aplicação de padrões em ES, melhoria contínua, aplicação de gestão de conhecimento.
12. Analisar e criar modelos relacionados ao desenvolvimento de software	Aplicar e selecionar técnicas de modelagem de software	Conteúdos: modelos estáticos, modelos funcionais, modelos dinâmicos, modelos formais.
	Aplicar técnicas de análise de modelos de software	Conteúdos: técnicas de análise de correção, de completitude, de consistência interna e entre modelos, de rastreabilidade entre modelos, de redundância, de ambiguidade.

13. Identificar novas oportunidades de negócios e desenvolver soluções inovadoras.	Entender conceito de empreendedorismo como processo de criar algo novo e com valor de negócio.	Conteúdos: empreendedorismo, análise e modelos de negócio; *frameworks* para construção de modelos de negócio.
	Entender o conceito de inovação como processo que transforma uma ideia em produto ou serviço.	Conteúdos: modelos de análise do processo de inovação, estratégias de inovação, planejamento e implementação de inovações, indicadores de inovação, políticas públicas e marco regulatório da inovação, inovação e desenvolvimento sustentável. Diferenciação entre inovação e pesquisa. Ecossistemas de inovação.
14. Identificar e analisar problemas avaliando as necessidades dos clientes, especificar os requisitos de software, projetar, desenvolver, implementar, verificar e documentar soluções de software baseadas no conhecimento apropriado de teorias, modelos e técnicas	Aplicar técnicas para identificar e analisar problemas avaliando as necessidades dos clientes	Conteúdo: técnicas de elicitação de requisitos
	Aplicar técnicas de especificação de requisitos de software	Conteúdos: técnicas de especificação, modelagem, verificação, validação e gerência de requisitos

	Aplicar teorias, modelos e técnicas para projetar, desenvolver, implementar e documentar soluções de software	Conteúdos: técnicas de revisão e análise estática de artefatos de software, técnicas de análise dinâmica de artefatos de software, refatoração, engenharia reversa, reengenharia, análise de impacto, manutenção, depuração, conceito de processo de software; modelos de processo de software; representação de processo de software, princípios de projeto (*design*), projeto (*design*) de arquitetura de software, padrões, reutilização de software, projeto (*design*) detalhado, projeto (*design*) de dados, projeto (*design*) de interface com usuários e projeto (*design*) de interface com outros sistemas, princípios de ES, aplicação de padrões em ES, melhoria contínua, aplicação de gestão de conhecimento
	Aplicar teorias, modelos e técnicas para verificar soluções de software	Conteúdos: técnicas de revisão e análise estática de artefatos de software, técnicas de análise dinâmica de artefatos de software

Apêndice 2: Colaboradores (em ordem alfabética)

- Abraham Lincoln Rabelo de Sousa (UNILASALLE)
- Adriana Pereira de Medeiros (UFF)
- Alessandra C. Smolenaars Dutra (PUCRS) *,**
- Alessandro Garcia (PUCRio) *
- Alexandre Cidral (UNIVILLE)
- Alfredo Goldman (IME) *
- Altigran da Silva (UFAM) *
- Amon José Aidukaitis (PETROBRÁS)
- Ana Paula Chaves Steinmacher (UTFPR) **
- Ana Paula Terra Bacelo (PUCRS) *
- Ana Regina Rocha (UFRJ) *
- André Villasboas (CPQD) *
- Arilo Dias Neto (UFAM) *
- Augusto Sampaio (UFPE) *
- Auri Vicenzi (UFG) *
- Avelino Francisco Zorzo (PUCRS) *
- Christina Chaves (UFBA) *
- Cláudia Werner (UFRJ) *
- Daltro José Nunes (UFRGS) *
- Edmundo Sérgio Spoto (UFG) *
- Elaine Venson (UNB)
- Ellen Francine (USP – São Carlos) *
- Evandro Franzen (UNIVATES)
- Fabio Gomes Rocha (UNIT) **
- Fábio Kon (USP-IME) *
- Fábio Lucena (UFG) *
- Fabricio Souza Pinto (UESB)
- Flávio Wagner (UFRGS) *
- Francisco José Mônaco (ICMC-USP) *
- Gleison dos Santos Souza (UNIRIO) *
- Guilherme Travassos (UFRJ) *

- Heitor Costa (UFLA) *
- Hilmer Rodrigues Neri (UNB) **
- Igor Fabio Steinmacher (UTFPR) *
- Igor Wiese (UTFPR) *
- Ingrid Oliveira de Nunes (UFRGS) *,**
- Itana Gimenes (UEM) *
- Jair Leite (UFRN)
- Jean Felipe Cheiran (UNIPAMPA)
- João Pablo Silva da Silva (UNIPAMPA)
- José Carlos Maldonado (USP-ICMC) *
- José Reginaldo Carvalho (UFAM) *
- Julio Cesar Sampaio do Prado Leite (PUC-Rio) *
- Leila Ribeiro (UFRGS)
- Manoel Mendonça (UFBA) *
- Marcelo Hideki Yamaguti (PUCRS) *,**
- Marcelo Quinta (UFG) *
- Marcelo Werneck (PUC Minas) *
- Marcia Lucena (UFRN)
- Márcio Delamaro (ICMC-USP) *
- Marcos Kalinowski (UFF) *
- Maria Augusta Vieira Nelson (PUC Minas) *
- Maria Claudia Figueiredo Pereira Emer (UTFPR) *
- Maurício Aniche (USP) *
- Michael Móra (PUCRS) *
- Newton Braga Rosa (UFRGS) *
- Paulo Borba (UFPE) *
- Paulo Cesar Masiero (USP – ICMC) *
- Paulo Meirelles (UNB) **
- Rafael Prikladnicki (PUCRS) *
- Raul Wazlawick (UFSC) *
- Rejane Figueiredo (UnB)
- Ricardo A. C. de Souza (UFRPE)

- Ricardo Anido (UNICAMP) *
- Rosana Vacari (ICMC-USP) *
- Sabrina Marczak (PUCRS) *
- Sandra Fabbri (UFSCAR) *
- Seiji Isotani (USP) *
- Sílvio Meira (UFPE) *
- Tayana Conte (UFAM) *,**
- Thais Batista (UFRN) *
- Thelma Elita Colanzi (UEM) **
- Uirá Kulesza (UFRN) *
- Vander Alves (UnB) *

Obs.:

* Membro de grupo de trabalho (convidado a partir de indicação no FEES 2014, ou por indicação de membros iniciais do grupo)

** Coordenador de grupo de trabalho (definido no FEES 2014)