Panorama do Ensino de Engenharia de Software em Cursos de Graduação Focado em Teste de Software: Uma Proposta de Aprendizagem Baseada em Jogos

Tarcila Gesteira da Silva – Universidade Federal de Santa Maria – tarcila@inf.ufsm.br
Felipe Martins Müller – Universidade Federal de Santa Maria – felipe@inf.ufsm.br
Giliane Bernardi – Universidade Federal de Santa Maria – giliane@inf.ufsm.br

Resumo. Este trabalho apresenta um panorama do ensino de engenharia de *software* em cursos de graduação na área de computação com base nos currículos de referência da Sociedade Brasileira de Computação (SBC) e da *Association for Computing Machinery* (ACM), focando no conteúdo referente ao tópico de teste de *software*. Com base neste panorama é apresentada uma proposta de aprendizagem baseada em jogos para apoio ao ensino-aprendizagem de estratégias de teste de *software*.
Palavras chaves: aprendizagem baseada em jogos, jogos sérios, ensino de engenharia de software, teste de software.

Overview of Software Engineering Education at Undergraduate Courses Focused on Software Testing: A Proposal of Games-Based Learning

Abstract. This paper presents an overview of software engineering education based in SBC and ACM reference curricula for undergraduate degree programs in computing, focusing in software testing content. Based on this outlook, we propose a game based learning approach to support teaching and learning of software testing strategies.
Keywords: game based learning, serious games, software engineering education, software testing.

1. Introdução

A engenharia de *software*, de modo geral, fornece toda a estrutura para o desenvolvimento e manutenção do *software*, sendo o teste de *software* uma das áreas de conhecimento da engenharia de *software* (IEEE, 2004). A área de teste de *software* destaca-se pela sua importância no desenvolvimento de sistemas que atendem a requisitos de qualidade. A área de engenharia de *software* é ampla e é um assunto recorrente nos currículos de graduação em computação. As Diretrizes Curriculares de Cursos da Área de Computação e Informática do MEC (Ministério de Educação) (CEEInf, 1999) reforçam a importância da assimilação dos conceitos e teorias relacionadas à engenharia de *software*, mas também recomendam que estes conhecimentos sejam aplicados através da prática em laboratórios e estágios. No entanto, a pesquisa de Wangenheim e Silva (2009), cujo público alvo abrange bacharéis da área de Ciência da Computação (CC) que trabalham como profissionais da área de *software*, confirma que os profissionais aprendem mais sobre tópicos de engenharia de *software* depois da graduação, ou seja, as competências de engenharia de *software* não estão sendo adequadamente abordadas nos cursos de CC. Stroustrup (2010) também discute as contradições dos currículos do curso de CC em relação às necessidades da indústria. Uma das causas dessa deficiência pode estar relacionada à carência de práticas educacionais alternativas à tradicional, que entende-se neste trabalho como sendo uma prática educacional unicamente expositiva. Uma aula expositiva acaba sendo pouco eficiente, já que ativa apenas o sentido da audição, enquanto que eventos simulados e atividades vivenciais permitem assimilar situações diferentes na prática (Prikladnicki *et al.* 2009). Entre as estratégias educacionais alternativas destacadas por

Prikladnicki *et al.* (2009) para o ensino-aprendizagem de Engenharia de *Software*, está o uso de jogos. Vários autores, entre eles, Mattar (2010), Wangenheim e Shull (2009) e Garris, Ahlers e Driskel (2002), destacam que o uso de jogos na educação pode ser um fator motivacional no processo de ensino-aprendizagem. Logo, levando em consideração a aprendizagem baseada em jogos (Garris, Ahlers e Driskel, 2002), os jogos sérios demonstram um grande potencial na educação.

Este trabalho tem o objetivo de apresentar uma abordagem para apoio ao processo de ensino-aprendizagem de teste de *software* baseada em conceitos de aprendizagem baseada em jogos. Esta proposta é embasada em uma visão geral do ensino de engenharia de *software* em curso de graduação com base em currículos de referência da área de computação. O trabalho está organizado da seguinte maneira: na seção 2 é apresentada uma analise de currículos de referência nacionais e internacionais para cursos de graduação na área de computação; na seção 3, aprendizagem baseada em jogos e jogos sérios são brevemente abordados e conceituados, sendo também apresentados trabalhos correlatos; na seção 4 é apresentada a proposta para apoio ao processo de ensino-aprendizagem de teste de *software*; e na seção 5 são realizadas considerações finais.

2. Analise de Currículos de Referência com foco em Teste de Software

Esta seção tem o objetivo de apresentar um levantamento inicial de como o tema teste de *software* é trabalhado em cursos de graduação em computação levando em consideração os currículos de referência existentes para a área. A tarefa de analisar os currículos dos cursos de graduação na área de computação não é simples, tendo em vista que, apenas no Brasil, existem 1790 cursos (INEP, 2008). Por esta razão, foi feita a analise de currículos de referência, que têm como objetivo servir como base para criação desses cursos.

A Sociedade Brasileira de Computação (SBC) propõe dois currículos de referência, sendo um para os cursos de Bacharelado em Ciência da Computação (CC) e Engenharia de Computação (EC) (SBC, 2005), e o outro para cursos de Bacharelado em Sistemas de Informação (SI) (SBC, 2003). No currículo proposto pela SBC para o curso de CC e EC as disciplinas da área de computação estão organizadas em dois núcleos: *Fundamentos da Computação* e *Tecnologia da Informação*. O núcleo de *Tecnologia da Computação* compreende, entre outras, a disciplina *Engenharia de Software*, sendo que o assunto teste de *software* é abordado em apenas um dos tópicos desta disciplina, que é *Verificação, Validação e Teste*. No currículo do curso de SI, as disciplinas de computação são organizadas em três núcleos: *Fundamentos da Computação* e *Sistemas de Informação* e *Formação Tecnológica*. No núcleo de *Formação Tecnológica*, assim como no currículo de CC e EC, é definida a disciplina *Engenharia de Software*, no entanto, a SBC (2003) recomenda que essa disciplina seja abordada com profundidade e deve cobrir, entre outros, o tema de teste de *software*.

A *Association for Computing Machinery* (ACM), em conjunto com outras associações, também propõe currículos de referência para cursos de graduação na área de computação, sendo que esses cursos são os seguintes: Ciência da Computação (ACM e IEEE, 2008), Engenharia da Computação (IEEE e ACM, 2004a), Sistemas de Informação (ACM e AIS, 2010), Tecnologia de Informação (IEEE e ACM, 2008) e Engenharia de *Software* (IEEE e ACM, 2004b). Nestes currículos de referência, com exceção do currículo de SI, são definidos números mínimos de horas necessárias para cobrir determinado assunto. A Tabela 1 apresenta, em detalhes, como o conteúdo referente a teste de *software* é distribuído nos currículos da ACM.

Tabela 1 – Mapeamento do conteúdo Teste de *Software* nos currículos ACM

Curso	**Disciplina**	**Tópico Relacionado a Teste**	**Conteúdo de Teste Coberto**
Ciência da Computação	Engenharia de *Software* - 31h	Verificação e validação de *software* - 3h	Fundamentos de teste, tipos de teste, técnicas de teste e estratégias de teste
		Ferramentas e ambientes - 3h	Ferramentas de teste de *software*
		Evolução de *software* - eletiva	Teste de regressão
Engenharia da Computação	Engenharia de *Software* - 13h	Teste e validação de *software* - 2h	Fundamentos e técnicas de teste, testes de unidade, integração, validação e sistema e teste orientado a objeto
		Ferramentas e Ambientes de *Software* - 2h	Teste de *software*
	Engenharia de Sistemas Computacionais - 18h	Teste - 2h	Plano de teste, tipos, níveis e ferramentas de teste e teste de placa de circuito impresso
	Interação Humano-Computador - 8h	Fundamentos da interação humano-computador - 2h	Introdução ao teste de usabilidade
		Avaliação de *software* centrada no humano - eletiva	Teste de usabilidade
Engenharia de *Software*	Garantia de Qualidade e Teste de *Software* - 37h	Teste - 14h	Técnicas de teste
	Teste de *Software* - 23h	Teste - 16h	Aborda em profundidade todos os aspectos do teste
	Fundamentos de Programação - 39h	Teste - 1h	Casos de teste, estratégias e ferramentas de teste e debuggig e teste de unidade
	Algoritmos e Estrutura de Dados - 31h	Teste - 1h	Fundamentos de teste
	Introdução à Engenharia de *Software* e Computação - 34h	Teste - 2h	Princípios básicos de teste
	Engenharia de *Software* e Computação II - 36h	Teste - 2h	Conceitos de teste são visto de forma prática
	Abordagem de Engenharia de *Software* para Interação Humano-Computador - 24h	Teste e avaliação de interface de usuário humano-computador - 6h	Teste de usabilidade
Sistemas de Informação	Analise e Projeto de Sistemas	Verificação e validação de sistemas	Fundamentos de teste
	Introdução à Interação Humano-Computador	Métodos de avaliação	Teste de usabilidade
	Desenvolvimento de Aplicativo	Teste de unidade	Teste de unidade
Tecnologia da Informação	Arquitetura e Integração de Sistemas - 21h	Teste e garantia de qualidade - 3h	Padrões e técnicas de teste, sendo aprofundados testes de usabilidade, aceitação, estresse e performance

		Requisitos - 6h	Teste dos requisitos
		Aquisição e fornecimento - 4h	Aborda a importância do teste na tomada de decisão de aquisição ou fornecimento
		Integração e implantação - 3h	Aborda o impacto do teste na integração e testes de aceitação
	Interação Humano-Computador - 20h	Avaliação Centrada no Humano - 3h	Teste de usabilidade
	Sistemas e Tecnologias Web - 22h	Desenvolvimento web - 3h	Teste de sistemas baseados na web

Com base nesses currículos de referência, observa-se que todos os cursos abordam assuntos relacionados a teste de *software*, sendo que o curso que menos aborda o tema é o curso de Tecnologia da Informação, seguido de Ciência da Computação e Engenharia da Computação. O curso de Engenharia de *Software* é o que aborda de forma mais completa o tema, estando presente em várias disciplinas do curso.

3. Aprendizagem baseada em jogos

A aprendizagem baseada em jogos é um tipo de aprendizagem que utiliza o jogo como uma ferramenta para que os estudantes se engajem ao aprendizado enquanto jogam (Sidhu, 2010). O modelo de aprendizagem proposto por Garris, Ahlers e Driskel (2002), apresentado na Figura 2, ilustra como ocorre o processo de aprendizagem baseada em jogos. A entrada deste processo combina, em um sistema, o conteúdo instrucional com as características de jogo. O ciclo do jogo consiste na interação do estudante com o sistema, que inclui as decisões ou reações do mesmo (como diversão ou interesse), seu comportamento (como maior persistência ou tempo na tarefa) perante essas decisões tomadas e o *feedback* do sistema que, conseqüentemente, desencadeia novamente o ciclo. Com a interação e o engajamento do estudante no jogo, ele alcança os objetivos educacionais e o ciclo termina com resultados de aprendizagem.

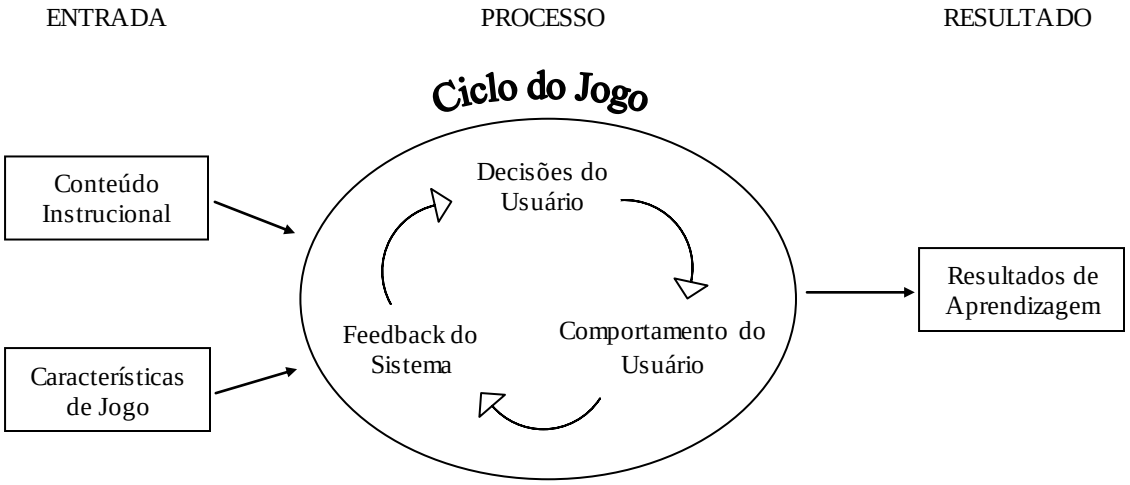

Figura 1 - Modelo de aprendizagem baseada em jogos (Fonte: Garris, Ahlers e Driskel, 2002)

Na literatura, os jogos usados no contexto educacional são denominados de várias formas, entre elas, jogos educacionais, jogos educativos, jogos instrucionais e jogos sérios, sendo que esta última é a nomenclatura adotada neste trabalho e será abordada na próxima seção.

3.1. Jogos sérios

Jogos podem ser jogados de forma séria ou casual. A expressão "jogos sérios" une a seriedade do pensamento e da resolução de problemas com o experimental e a liberdade emocional do jogo, sendo assim, os jogos sérios combinam a concentração analítica e questionadora do ponto de vista científico com a liberdade intuitiva e recompensa construtiva dos atos artísticos (Abt, 1987). A principal característica do jogo sério é sua finalidade educacional explícita e cuidadosamente pensada, não tendo como intenção principal ser utilizado por diversão, o que não significa que o jogo sério não possa ser divertido (Michael e Chen, 2006). Jogos sérios oferecem então um novo mecanismo para ensino e treinamento, combinando jogos digitais com educação. Logo, jogos sérios podem ir além das vídeo-aulas e livros, permitindo aos jogadores não apenas o aprendizado, mas também a demonstração e aplicação dos conhecimentos aprendidos (Michael e Chen, 2006). No entanto, a utilização de jogos sérios requer planejamento, pois, nenhum jogo sério pode ser bem sucedido se os jogadores não entenderem as regras, seus objetivos no jogo, as conseqüências de suas ações e as razões para suas conseqüências (Abt, 1987).

3.2. Jogos para engenharia

Como já discutido na introdução, o uso de jogos pode ser agregado à estratégia de ensino-aprendizagem de engenharia de *software*. Existem vários trabalhos na literatura sobre jogos sérios para engenharia de *software* como, por exemplo:

- Elicit@ção (Vargas et. al, 2010): em que o estudante assume o papel de analista e simula a elicitação de requisitos de *software*;
- XMED v1.0 (Wangenheim et al., 2009): aborda o assunto de medição de *software*;
- SE•RPG (Benitti e Mollíre, 2008): é um jogo de RPG que proporciona a experiência de gerenciamento e desenvolvimento de um projeto de *software*;
- SimSE (Navarro, 2009): simula processos de engenharia de *software* em diferentes metodologias de desenvolvimento; e
- U-TEST (Silva, 2010): um jogo para o ensino de teste de unidade com a aplicação de técnicas e práticas relacionadas à seleção de dados de entrada.

Todos esses jogos são monousuário e possuem gráficos 2D (duas dimensões), sendo o XMED v1.0 e o SimSE desenvolvidos com a linguagem Java e os demais utilizando a ferramenta Adobe Flash. O U-TEST foi o único jogo encontrado na literatura voltado especificamente para o ensino de teste de *software*, no entanto, o mesmo aborda apenas uma pequena parte relacionada à área de teste de *software*, o teste de unidade. Considerando este cenário, na próxima seção será apresentada a proposta de um jogo seguindo a abordagem de aprendizagem baseada em jogos para teste de *software*.

4. Proposta de Jogo para Teste de Software

Como destacado na seção 3 e ilustrado na Figura 1, a aprendizagem baseada em jogos une o conteúdo instrucional com características de jogo em uma aplicação e a partir da interação do estudante têm-se os resultados de aprendizagem. Logo, esta seção está divida em três partes, sendo que a primeira descreve as características do jogo, a

segunda apresenta o *design* instrucional e a última relata como será realizada a avaliação dos resultados da aprendizagem.

4.1. Características do Jogo

O jogo sério proposto, denominado Jogo da Equipe de Teste de *Software* (JETS), visa apoiar a relação de ensino-aprendizagem de teste de *software* em cursos de graduação na área de computação. O diferencial do JETS, perante aos trabalhos correlatos, está no gráfico 3D, no fato de ser multiusuário e também na possibilidade de customização por parte do professor (característica encontrada somente no SimSE), tendo em vista a adaptação aos diferentes currículos e planos de aula. O JETS é classificado quanto ao gênero como simulador e está sendo desenvolvido no mundo virtual *Open Simulator*[1]. O mundo virtual será integrado ao ambiente virtual de ensino aprendizagem Moodle[2], de modo que os resultados do jogo possam ser enviados ao ambiente na forma de tarefa.

Este jogo visa simular as interações de uma equipe de teste de uma empresa de desenvolvimento de *software*. O jogo é constituído de quatro fases, sendo que em cada fase o estudante assumirá um cargo diferente dentro de uma equipe de testadores. Na primeira fase, o estudante ingressa no jogo com o cargo de *Testador de Software* e deve executar casos de teste prontos. Na segunda fase, o estudante assume o papel de *Analista de Teste* e deve elaborar os casos de teste a partir de interfaces e métodos pré-determinados. Os casos de teste criados pelo estudante poderão ser exportados para a ferramenta *TestLink*[3], de modo a permitir o contato e experimentação de uma ferramenta específica da área de teste, ultrapassando assim os limites da simulação do jogo para o mundo real. Na terceira fase, o estudante assume o cargo de *Arquiteto de Teste* e deve realizar atividades relacionadas com a infra-estrutura do teste, isto é, ele deve definir o ambiente de teste e escolher as ferramentas de teste apropriadas. Na quarta e última fase do jogo, o estudante assume o cargo de *Líder (gerente) da Equipe de Teste*, onde será responsável pela alocação de recursos, acompanhamento das atividades de teste, verificando prazos e custos do processo de teste de *software* de um projeto. Os desafios de cada fase, bem como as respectivas pontuações, poderão ser editados pelo professor.

4.2. Design Instrucional

O JETS tem como objetivo aprimorar o conhecimento de estratégia de teste de *software* do estudante e fornecer uma visão geral de como o teste de *software* pode ser implementado em uma empresa. Cabe destacar que a proposta é de um jogo de apoio ao ensino-aprendizagem, pois como destaca (CEEInf, 1999) a disciplina de engenharia de *software* deve fornecer uma base teórica consistente, logo, o jogo não deve excluir a necessidade de um professor. Neste jogo será abordado o tópico estratégia de teste de *software*. O *design* instrucional segue o modelo proposto por Kochanski (2009) e está representado na Tabela 2.

Tabela 2 – Design instrucional da abordagem de aprendizagem baseada em jogos

Contexto

[1] http://www.opensimulator.org/
[2] http://www.moodle.org/
[3] http://www.teamst.org/

<table>
<tr><td colspan="3">O público alvo são estudantes de cursos de graduação na área de computação. Esta abordagem visa proporcionar aos estudantes embasamento teórico e prático sobre os conceitos básicos de teste de software.</td></tr>
<tr><td colspan="3">Pré-Condições</td></tr>
<tr><td colspan="3">O estudante deve estar cursando ou ter cursado alguma disciplina que aborde o tema teste de software.</td></tr>
<tr><td colspan="3">Ementa</td></tr>
<tr><td colspan="3">Fundamentos de teste de software, casos de teste e estratégia de teste de software.</td></tr>
<tr><td colspan="3">Objetivos Gerais e de Aprendizagem</td></tr>
<tr><td colspan="3">Identificar os principais conceitos de teste de software. Reconhecer e entender as técnicas de teste de software. Entender e aplicar as técnicas relacionadas à estratégia de teste de software.</td></tr>
<tr><td>Conteúdos</td><td>Objetivos de Aprendizagem</td><td>Estratégias de Ensino</td></tr>
<tr><td>Unidade1 – Fundamentos de teste de software
Referências: [1] e [2]</td><td>Reconhecer os principais conceitos relacionados a teste de software</td><td>Aula expositiva.</td></tr>
<tr><td>Unidade 2 – Documentação de teste de software e desenvolvimento de casos de teste
Referências: [3] e [4]</td><td>Identificar e compreender a documentação do processo de teste de software.</td><td>Aula expositiva e exercícios de fixação.</td></tr>
<tr><td>Unidade 3 – Aplicação de conceitos relacionados à estratégia de teste de software
Referências: [1], [2], [3] e [4]</td><td>Aplicar conceitos execução e desenvolvimento de casos de teste, definição de ambientes e ferramentas de teste, estratégia de teste.</td><td>Exercício prático por meio do JETS (Jogo da Equipe de Teste de Software).</td></tr>
<tr><td colspan="3">Avaliação</td></tr>
<tr><td colspan="3">A avaliação ocorrerá por meio de dois testes de 10 questões, sendo divididas em três segmentos:
Reconhecimento: 3 questões de múltipla escolha, em que o objetivo é verificar se o aluno é capaz de reconhecer os conceitos básicos de teste de software.
Entendimento: 3 questões de múltipla escolha, em que o objetivo é verificar se o aluno compreende os principais conceitos relacionados à estratégia de teste de software.
Aplicação: 4 questões, sendo 3 de múltipla escolha e 1 discursiva, em que é verificada a capacidade do aluno de aplicação dos conceitos de estratégia de teste de software.</td></tr>
<tr><td colspan="3">Referências</td></tr>
<tr><td colspan="3">[1] PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. Porto Alegre: AMGH, 7.ed. 2011.
[2] SOMMERVILLE, I. Engenharia de software. São Paulo: Pearson Addison Wesley, 8.ed. 2007.
[3] INSTITUTE OF ELECTRICAL AND ELECTRONIC ENGINEERS COMPUTER SOCIETY. SWEBOK: Guide to the Software Engineering Body of Knowledge, 2004.
[4] MYERS, G. J. et al. The art of software testing. New Jersey: John Wiley & Sons, 2.ed. 2004.</td></tr>
</table>

4.3. Avaliação

A avaliação seguirá o *framework* proposto por Kochanski (2009), para apoiar a construção de experimentos na avaliação empírica de jogos educacionais na área de engenharia de *software*. Este *framework* é dividido em cinco partes: definição do experimento, planejamento do experimento, operação do experimento, analise e interpretação dos dados e apresentação dos resultados. A avaliação do JETS deve ser realizada em cursos de graduação na área de computação, em disciplinas que possuam o tópico de teste de *software*. Esta avaliação tem como objetivo confirmar ou não a hipótese de que um jogo sério para apoio ao ensino e aprendizagem de estratégias de teste de *software* é uma estratégia educacional alternativa válida, podendo realmente auxiliar no processo de ensino-aprendizagem de teste de *software*.

Por se tratar de uma ferramenta de apoio ao professor, antes da aplicação do JETS será ministrada uma aula teórica expositiva, onde serão abordados conceitos relacionados a estratégias de teste de *software*. Após a aula, será realizado um teste para avaliar o conhecimento adquirido pelos estudantes sobre o tema abordado. A turma será dividida em dois grupos (A e B), sendo que a distribuição ocorrerá de forma aleatória. O grupo A utilizará o JETS e o grupo B será o grupo de controle do experimento. Após a aplicação do jogo será realizado um novo teste que avaliará o conhecimento adquirido após a utilização do jogo. O grupo A também responderá um questionário com perguntas referentes à usabilidade, jogabilidade e percepções do jogo.

Os resultados serão analisados de forma quantitativa, medindo a aprendizagem por meio da comparação dos testes realizados antes e depois da aplicação do jogo, e de forma qualitativa, analisando as respostas do questionário referente ao jogo.

5. Considerações Finais

É fato que os currículos dos cursos de graduação na área de computação variam dependendo da instituição. No entanto, a engenharia de *software* é um tema de extrema importância para área e dificilmente não é abordada nesses currículos. O teste de *software* tem um papel importante no desenvolvimento de sistemas, pois a partir dos testes podem ser identificadas possíveis mudanças no projeto e também é um importante quesito da garantia da qualidade. O teste de software é abordado em todos os currículos de referência propostos tanto pela SBC, quanto pela ACM, portanto, é um tema de relevância para a área. Diferentes estratégias de ensino-aprendizagem podem ser adotadas na abordagem do tema, entre elas está o uso de jogos.

A aprendizagem baseada em jogos leva em consideração que os estudantes mudaram em diferentes pontos essenciais, pois é uma geração que cresceu jogando *videogames* e utilizando o computador, e por esse motivo, mudou a forma de pensar, aprender e processar informações (Prensky, 2001). Logo, hoje o aprendizado necessita de motivação para um envolvimento intenso, o que é atingido pelos jogos digitais (Mattar, 2010). A aprendizagem baseada em jogos digitais é uma abordagem inovadora no domínio das universidades, sendo que os jogos têm sido explorados como uma nova forma de conteúdo interativo (Pivec, 2007). Vale destacar que experiências contemporâneas com jogos na educação já estão deixando de lado a mera analise e interpretação para utilizar a própria criação e *design* de jogos como objetivos educacionais (Mattar, 2010).

O JETS está em fase de desenvolvimento. Para agilizar a construção dos cenários do jogo, estão sendo utilizados objetos prontos, desenvolvidos pela comunidade do OpenSim. Para o armazenamento das informações do jogo em um banco de dados, devido à arquitetura de comunicação de dados deste mundo virtual, é necessário uma aplicação *web* externa, sendo que a mesma está em fase de finalização. No momento, estão sendo desenvolvidos os *scripts* do jogo. Finalizado o desenvolvimento, esta abordagem será aplicada em cursos de graduação na área de computação. Ainda, a colaboração deste trabalho não está somente na área de teste de *software*, mas também visa servir como alicerce para o desenvolvimento de outras aplicações de aprendizagem baseada em jogos para apoio ao ensino-aprendizagem de outros tópicos de engenharia de *software* ao até mesmo de outras disciplinas dos currículos de computação.

6. Referências

ABT, C. C. **Serious games**. Boston: University Press of America, 1987.

ACM; AIS. **Model curriculum and guidelines for undergraduate degree programs in information systems**, 2010.

ACM; IEEE COMPUTER SOCIETY. **Computer science curriculum 2008**: An Interim Revision of CS 2001, 2008.

BENITTI, F. B. V.; MOLLÉRI, J. S. Utilização de um RPG no ensino de gerenciamento e processo de desenvolvimento de software. In: **WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO**, 16., 2008, Belém do Pará. Anais. Belém do Pará: SBC, 2008.

CEEInf. **Diretrizes Curriculares de Cursos da Área de Computação e Informática**. 1999. Disponível em: <http://www.inf.ufrgs.br/ecp/docs/diretriz.pdf>. Acesso em: 26 mai. 2011.

GARRIS, R.; AHLERS, R., DRISKELL, J. E. Games, motivation, and learning: a research and practice model. **Simulation & Gaming**, v. 33, n. 4, p. 441-467, dez. 2002. Disponível em: <http://sag.sagepub.com/content/33/4/441>. Acesso em: 20 mar. 2011.

IEEE COMPUTER SOCIETY. **SWEBOK**: guide to the software engineering body of knowledge, 2004.

IEEE COMPUTER SOCIETY; ACM. **Curriculum guidelines for undergraduate degree programs in computer engineering**, 2004a.

_____. **Curriculum guidelines for undergraduate degree programs in software engineering**, 2004b.

_____. **Curriculum guidelines for undergraduate degree programs in information technology**, 2008.

INEP (2007) **Censo da educação superior 2008**. Disponível em: <http://www.ufrgs.br/sai/dadosresultados/dadoseresultados.html>. Acesso em: 3 ago. 2011.

KOCHANSKI, D. **Um framework para apoiar a construção de experimentos na avaliação empírica de jogos educacionais**. 2009. 224f. Dissertação (Mestrado em Computação Aplicada) – Universidade do Vale do Itajaí, São José, 2009.

MATTAR, J. **Games em educação**: como nativos digitais aprendem. São Paulo: Person, 2010.

MICHAEL D.; CHEN, S. **Serious games**: games that educate train and inform. Boston: Thomson Course Technology, 2006.

NAVARRO, E. e HOEK, A. Multi-site evaluation of SimSE. In: **The 40th ACM Technical Symposium on Computer Science Education**. New York: ACM, 2009.

PIVEC, M. Play and learn: potentials of game-based learning. **British Journal of Educational Technology** v. 38, n. 3, p. 387-393, 2007.

PRENSKY, M. **Digital game-based learning**. New York: McGraw-Hill, 2001.

PRIKLADNICKI, R. et al. Ensino de engenharia de software: desafios, estratégias de ensino e lições aprendidas. In: **FÓRUM DE EDUCAÇÃO EM ENGENHARIA DE SOFTWARE**, 2.,2009, Fortaleza. Anais. Fortaleza: UFC, 2009.

SIDHU, M. S. **Technology-assisted problem solving for engineering education: interactive multimedia applications**. Hershey: Engineering Science Reference, 2010.

SILVA, A. C. **Jogo educacional para apoiar o ensino de técnicas para elaboração de testes de unidade**. 2010. 179f. Dissertação (Mestrado em Computação Aplicada) – Universidade do Vale do Itajaí, São José, 2010.

SBC. **Currículo de referência da SBC para cursos de graduação em bacharelado em ciência da computação e engenharia de computação**, 2005.

_____. **Currículo de referência para cursos de bacharelado em sistemas de informação**, 2003.

STROUSTRUP, B. What should we teach new software developers? Why? **Communications of the ACM**, v.53, n.1, jan. 2010.

VARGAS et. al. Desenvolvimento de um jogo de empresa baseado em agentes de software e instituições eletrônicas para simulação de elicitação de requisitos de software. In: **WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO**, 18., 2010, Minas Gerais. Anais. Minas Gerais: PUC, 2010.

WANGENHEIM, C. G. et al. Desenvolvimento de um jogo para ensino de medição de software. VIII **SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE**, 8., 2009, Minas Gerais. Anais. Minas Gerais: PUC, 2009.

WANGENHEIM, C. G.; SHULL, F. To game or not to game? **IEEE Software**, v.26, n.2, mar./abr., p. 92-94, 2009.

WANGENHEIM, C. G.; SILVA, D. A. Qual conhecimento de engenharia de software é importante para um profissional de software? In: **FÓRUM DE EDUCAÇÃO EM ENGENHARIA DE SOFTWARE**, 2.,2009, Fortaleza. Anais. Fortaleza: UFC.