# SPIAL: Uma Ferramenta de Apoio ao Aprendizado de Melhoria de Processos de Software

**Daniela C. C. Kupsch (autora), Rodolfo S. F. Resende (orientador)**

Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte – MG – Brasil

{cascini,rodolfo}@dcc.ufmg.br

***Resumo.*** *Os requisitos para desenvolvimento de software estão mais complexos e as soluções devem ser obtidas de forma rápida e barata. As disciplinas de Engenharia de Software que tratam desse conhecimento proveem uma visão introdutória da área e não abordam suficientemente as práticas das organizações desenvolvedoras de software. Com o objetivo de prover uma experiência mais próxima à da indústria, nós criamos SPIAL, um jogo de simulação que aborda as melhores práticas de Engenharia de Software. SPIAL apresenta a metáfora de uma organização desenvolvedora de software que está executando uma iniciativa de melhoria de processos.*

***Abstract.*** *The complexity of software development is increasing and at the same time, the markets require reduced costs and short time production. The Software Engineering disciplines provide an introductory overview of the area and do not sufficiently address the practical issues of software development organizations. With the aim of providing to students experiences that resemble more closely those in industry, we created SPIAL, a simulation game that addresses the Software Engineering best practices. SPIAL presents a metaphor of a software development organization that is running a process improvement initiative.*

## 1. Descrição do Problema e Motivação

Um dos grandes desafios enfrentados pela sociedade e em particular pelas universidades é a necessidade de tornar o processo de formação e treinamento das pessoas mais efetivo e eficaz. A expectativa de uma maior qualidade na formação e treinamento não tem sido satisfeita de uma maneira geral e existe um aumento da insatisfação da indústria em relação à preparação dos estudantes de vários setores. Soma-se a isso a dificuldade de aplicação prática dos conceitos apresentados em sala de aula no contexto de ensino centrado no professor [Chen e outros 2008]. Uma forma de melhorar este cenário consiste na utilização de tecnologias e métodos de apoio ao ensino como, por exemplo, desenvolvimento de atividades em projetos da indústria, realização de programas de residência, jogos e simuladores.

Mudanças profundas no processo de educação e nas instituições de ensino são esperadas com a utilização de novas tecnologias educacionais e pedagógicas [Hiltz e Turoff 2005]. Atualmente, é possível notar adaptações da metodologia tradicional. Cursos face-a-face, centrados no professor, que são comumente oferecidos por universidades do mundo inteiro, estão sendo substituídos por cursos mais dinâmicos, híbridos com a utilização de tecnologias inovadoras. Exemplos importantes incluem os

DOI: 10.5753/CBIE.WCBIE.2013.50

cursos on-line e gratuitos (MOOC – *Massive Open Online Course*) oferecidos por universidades renomadas que operam numa escala global como Stanford, com a tecnologia do Coursera[1], e MIT, Berkeley e Harvard, com edX[2]. Outros exemplos incluem a utilização de jogos de simulação como método complementar ao ensino de determinados assuntos abordados em disciplinas de graduação, como por exemplo, SimSE [Navarro 2006] e SESAM [Drappa e Ludewig 2000].

A simulação é uma ferramenta educacional poderosa que é comumente utilizada no ensino de processos e técnicas que são de execução impossível ou de alto custo para a indústria. O seu uso surgiu a partir da avaliação do potencial das novas tecnologias aplicadas ao ensino. Com a simulação é possível despertar o interesse dos estudantes, e os resultados são frequentemente surpreendedores [Ricci e outros 1996]. Apresentar questões relacionadas a grandes projetos em um simulador permite a aplicação dos conceitos ensinados em sala de aula dentro de um contexto mais próximo ao da indústria. Alguns trabalhos mostram que os simuladores baseados em jogos podem ser utilizados como uma ferramenta de apoio ao ensino [Navarro 2006, Drappa e Ludewig 2000, Boehm e Huang 2003, Pfahl e outros 2001]. Essas ferramentas tratam das atividades executadas e de seus responsáveis em projetos de grande porte, de forma que a universidade, com os seus projetos de pequeno porte, não conseguiriam tratar.

A utilização de jogos de simulação consiste de uma mudança de paradigma de ensino: do aprendizado ao se escutar (*learning by listening)* para o aprendizado ao se fazer (*learning by doing*) [Garris e outros 2002]. Uma razão para a sua adoção é que existem evidências empíricas que evidenciam que os jogos de simulação são ferramentas efetivas para a melhoria do aprendizado e compreensão de assuntos complexos [Ricci e outros 1996, Cordova e Lepper 1996]. Os jogos de simulação permitem aumentar o interesse dos estudantes nos assuntos abordados em sala de aula. Além disso, possibilitam que o professor aborde certos aspectos que não são satisfatoriamente cobertos nas atividades práticas da disciplina.

No contexto da Engenharia de Software observa-se que é muito difícil treinar os estudantes em situações próximas das que ocorrem nas organizações desenvolvedoras de software. Isso se deve, principalmente, a natureza das aplicações de software e a diversidade das culturas organizacionais. Uma série de jogos de simulação foi desenvolvida com o objetivo de melhorar o aprendizado e a compreensão de Engenharia de Software [Peixoto e outros 2011b]. Esses jogos de simulação abordam diferentes aspectos da Engenharia de Software o que sugere que este é um campo de pesquisa atraente, com grandes possibilidades de expansão.

Com o objetivo de ter um entendimento melhor das dificuldades dos alunos de um curso introdutório de Engenharia de Software, nós analisados os defeitos produzidos pelos alunos do curso de Engenharia de Software do Departamento de Ciência da Computação da UFMG [Peixoto e outros 2010a]. Neste curso de graduação, os estudantes desenvolvem o trabalho prático em grupos de 4 a 6 alunos. O objetivo do trabalho é proporcionar aos alunos uma experiência prática dos conceitos apresentados em sala de aula. Cada grupo é responsável pela especificação, desenho e implementação de um pequeno software. Todos os artefatos produzidos durante o desenvolvimento são

[1] https://www.coursera.org/

[2] https://www.edx.org/

inspecionados pelo monitor da disciplina. Os defeitos identificados são relatados e documentados. Após analisarmos a distribuição dos defeitos durante dois anos do curso, observamos que existe uma lacuna entre o que é ensinado em sala de aula e o que os alunos deveriam fazer no trabalho em grupo. Além disso, identificamos uma grande dificuldade de entendimento e aplicação do processo de desenvolvimento de software, principalmente relacionado à aplicação dos processos gerenciais. Considerando esses aspectos e a necessidade de trazer questões práticas para a sala de aula, nós desenvolvemos SPIAL (*Software Process Improvement Animated Learning Environment*) [Peixoto 2012c]. SPIAL é um jogo de simulação gráfico, interativo e personalizável. SPIAL permite que os alunos pratiquem algumas habilidades de Engenharia de Software em um ambiente que se assemelha ao da indústria. Especificamente, com este jogo os alunos serão capazes de:

- Praticar conceitos relacionados aos processos técnicos e gerenciais do desenvolvimento de software.
- Reforçar os conceitos apresentados em sala de aula, principalmente, as boas práticas de Engenharia de Software, executando uma iniciativa de Melhoria de Processos de Software (MPS). Por exemplo, o jogo permite que os alunos identifiquem os efeitos da regra: "deficiências nos requisitos são a principal fonte de falhas de um projeto" [Glass 1998]. O jogo premia as decisões corretas (práticas apropriadas de Engenharia de Software) e penaliza as que não são justificáveis.
- Seguir as etapas de um processo de desenvolvimento de software similares a de uma organização.
- Aprender alguns eventos que podem ocorrer durante um projeto de MPS como, por exemplo, resistência à mudança, e comprometimento da alta gerência.
- Observar os possíveis resultados de ações de melhoria realizadas durante o desenvolvimento (por exemplo, redução de defeitos após a execução de atividades de Engenharia de Requisitos).
- Analisar os efeitos de um processo de desenvolvimento de software imaturo (por exemplo, visibilidade restrita das medidas do processo, e maior número de defeitos). Portanto, reforçar as vantagens de se ter um processo definido e documentado.

As seções seguintes estão estruturadas da seguinte forma. A Seção 2 apresenta uma descrição resumida do SPIAL. A Seção 3 descreve os trabalhos relacionados. A Seção 4 detalha as contribuições desta tese. A Seção 5 apresenta os principais resultados alcançados. Finalmente, a Seção 6 conclui este documento.

## 2. SPIAL – Visão Geral

SPIAL é um jogo monousuário no qual o jogador assume o papel de um gerente de um grupo de MPS em uma organização desenvolvedora de software. O escopo do SPIAL envolve dois projetos: (i) um projeto de desenvolvimento de software, e (ii) um projeto de melhoria de processos de software. No início do jogo, o jogador recebe uma tarefa de melhoria. Durante o jogo, o jogador pode interagir com outras pessoas que são representados pelos papéis das partes interessadas, como a alta gerência, gerente de projeto, membro da equipe, consultor e cliente. Esses papéis são controlados pelo computador e não pelo jogador (veja Figura 1a). Com o objetivo de completar uma tarefa, o jogador pode fazer investimentos para melhorar áreas de processo que

impactam o projeto correspondente de desenvolvimento de software. Uma boa estratégia de investimento irá resultar em melhorias das áreas de processo, um orçamento maior para investir e um impacto positivo nas medidas do projeto de desenvolvimento. O jogador pode visualizar as estimativas de projeto, indicações do nível de capacidade das áreas de processo e decidir em qual área de processo investir. Durante o desenvolvimento do software, o jogador pode visualizar os efeitos das suas decisões nas saídas que são representadas por medidas de produtividade, custo, defeito, e tempo de entrega. Após esta análise, o jogador decide se será necessário realizar mudanças na sua estratégia de investimento (veja Figura 1b). O resultado final do jogo é uma pontuação que representa o quão perto os resultados estão da meta inicial (tarefa de melhoria). Durante o jogo, as partes interessadas comunicam os efeitos das ações do jogador através de mensagens que aparecem sobre a imagem das suas cabeças.

O objetivo do jogo é melhorar o aprendizado de Engenharia de Software, utilizando simulação. Vários temas da Engenharia de Software são explorados no contexto da chamada Melhoria de Processo de Software que é simulada no SPIAL com base no CMMI [CMMI 2010], um modelo bastante difundido internacionalmente. O SPIAL permite replicar a realidade de uma organização desenvolvedora de software, o que, na maioria das vezes, não é possível apresentar para os estudantes durante um projeto de uma disciplina.

A avaliação do SPIAL foi realizada através de um experimento piloto e de uma inspeção utilizando o Método de Inspeção Semiótica (MIS) [Peixoto e outros 2010d]. O MIS aplica um método interpretativo e qualitativo do domínio da Engenharia Semiótica [de Souza 2005]. O foco está na análise de como o projetista do software se comunica com os usuários através da interface do software. Os aspectos educacionais foram avaliados em um experimento piloto. Neste experimento foi abordada a capacidade dos alunos em compreender, lembrar e aplicar conceitos de Engenharia de Software no contexto de uma iniciativa de MPS baseada no CMMI. Além das impressões dos estudantes, nossa avaliação obteve evidências de que o SPIAL é uma metodologia complementar e útil para o ensino de MPS e de conceitos de Engenharia de Software.

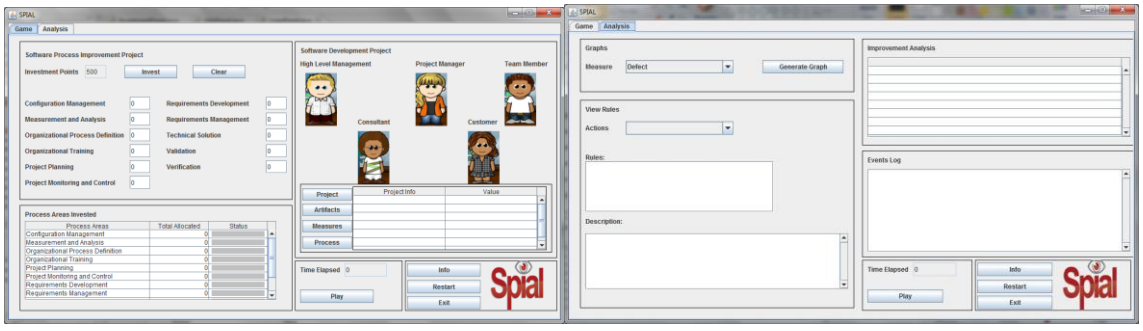


**Figura 1. Interface Gráfica do SPIAL: (a) Aba principal; (b) Aba de análise.**

## 3. Trabalhos Relacionados

Existem vários jogos educacionais com características e enfoques variados. Em um contexto próximo ao deste trabalho, destacam-se jogos cujo objetivo é o ensino de processos de desenvolvimento de software, em que o jogador assume o papel de gerente de projetos (exemplos incluem SESAM [Drappa e Ludewig 2000], SimSE [Navarro

2006], SimJavaSP [Shaw e Dermoudy 2005] e MO-SEProcess [Ye e outros 2007]); e jogos que abordam uma disciplina específica de Engenharia de Software (exemplos incluem qGame [Knauss e outros 2008], The Incredible Manager [Barros e outros 2002], SimVBSE [Jain e Boehm 2006] e TREG [Vega e outros 2009]).

A partir dos relatos encontrados na literatura [Ricci e outros 1996, Boehm e Huang 2003] é possível observar que grande parte dos jogos foca nos mesmos aspectos, o gerenciamento de um projeto de desenvolvimento dentro do orçamento, prazo e escopo previamente estabelecidos. Torna-se evidente que o escopo da área de desenvolvimento de software ainda não foi amplamente contemplado.

Com o propósito de tornar a experiência mais educacionalmente efetiva e abrangente, foi criado o SPIAL, um jogo que aborda práticas de Engenharia de Software e Melhoria de Processos de Software. Além disso, SPIAL incorpora questões que ainda não foram tratadas de forma adequada neste contexto de pesquisa, como adaptabilidade do modelo de simulação, avaliação da interação e verificação dos resultados utilizando experimentos. Assim, de forma geral, esta tese avança o conhecimento acerca do desenvolvimento e aplicação de jogos educacionais no contexto de Engenharia de Software.

## 4. Contribuições

Duas contribuições destacam-se no escopo desta pesquisa. A primeira consiste na identificação de um processo de desenvolvimento e validação que deu suporte a criação do SPIAL. A segunda consiste na concepção e aplicação de um jogo de simulação no ensino de Engenharia de Software. Os resultados deste trabalho possuem grande enfoque em aspectos de interação.

As principais contribuições obtidas com o desenvolvimento deste trabalho podem ser sumarizadas como abaixo:

**a. Aspectos teóricos:**

- Descrição das principais características que tornam a adoção de jogos de simulação um sucesso. Estas informações foram consolidadas em Peixoto e outros [2012a].
- Especificação de um jogo de simulação de MPS com: (i) definição de papéis, atividades e resultados; (ii) definição de um modelo de simulação; e (iii) definição de um *framework* reutilizável;
- Identificação e caracterização dos principais resultados das iniciativas de MPS coletados da literatura.
- Identificação dos problemas e desafios durante o desenvolvimento do SPIAL [Peixoto e outros 2012b].

**b. Aspectos práticos:**

- Desenvolvimento de um conjunto de componentes que podem ser reutilizados no desenvolvimento de novos jogos de simulação de Engenharia de Software [Peixoto e outros 2012a].
- Integração de um modelo de simulação específico de MPS com os componentes reutilizáveis [Peixoto e outros 2013].
- Desenvolvimento do jogo de simulação juntamente com o seu modelo de simulação de MPS.

**c. Aspectos empíricos:**

- Avaliação de comunicabilidade do SPIAL conduzida por um especialista. Pela primeira vez, o método de inspeção semiótica foi aplicado neste contexto de jogos educacionais.
- Avaliação prática do SPIAL realizada pelos alunos através de um experimento piloto.

## 5. Resultados

O principal resultado desta tese consiste na concepção, desenho, aplicação e validação do SPIAL, um jogo de simulação de MPS. A inovação deste trabalho pode ser caracterizada em dois aspectos. Primeiramente, este trabalho apresenta um simulador de MPS, com seu modelo de simulação e componentes reutilizáveis. Em segundo, ele apresenta os passos necessários para a criação de um jogo de simulação, que inclui a análise de questões de Interação Humano Computador e da literatura de MPS.

Esta pesquisa gerou resultados nos níveis teórico, prático e empírico, como sumarizados na seção anterior. Os principais resultados podem ser caracterizados como abaixo:

**a) Identificação de erros dos estudantes**: Nós investigamos os principais problemas incorridos por estudantes de um curso de Engenharia de Software [Peixoto e outros 2010a]. Os aspectos mais marcantes consistem (i) na dificuldade dos alunos em preencher a lacuna entre as aulas teóricas e trabalho prático; (ii) conjunto limitado de habilidades que os estudantes aplicam durante o desenvolvimento do projeto e, (iii) geralmente, eles não seguem o processo de desenvolvimento de software durante o desenvolvimento do projeto. Eles primeiro elaboram os artefatos técnicos e então complementam a linha de base com os artefatos gerenciais. (iv) Mesmo quando explicitamente solicitados, os alunos não inspecionam seus artefatos. Infelizmente, em algumas ocasiões, percebemos que os relatórios de defeitos eram preenchidos com dados irreais. Isto foi observado por causa de inconsistências entre artefatos.

**b) Análise de outros jogos de simulação:** Nós identificamos e comparamos oito jogos de simulação de Engenharia de Software (Seção 2) [Peixoto e outros 2011b]. Baseada nesta análise, nós definimos os aspectos essenciais de desenho para o desenvolvimento do nosso jogo. A avaliação desses jogos de simulação revelou que, embora eles tenham sido desenvolvidos com objetivos diferentes, eles possuem um número considerável de aspectos em comum, como por exemplo, estratégia de "tentativa e erro", regras de Engenharia de Software e algumas características de jogos. O que os diferenciam está relacionado com o momento que a realimentação (*feedback)* está disponível para o jogador, o tipo de realimentação, e características de adaptabilidade.

**c) Investigação dos domínios de MPS e Engenharia de Software**: Nós investigamos o domínio do nosso jogo de simulação. Como um jogo de simulação deve refletir o que acontece nas organizações desenvolvedoras de software, nós analisamos os principais resultados relatados por essas organizações em publicações científicas relacionados com iniciativas de MPS. Realizando uma revisão sistemática da literatura, nós conseguimos uma visão atualizada da área, permitindo a identificação e caracterização dos principais resultados de iniciativas de MPS. No contexto do SPIAL, os resultados deste estudo proveram uma base para a definição do modelo de simulação e dos requisitos. Nós

analisamos 91 estudos relacionados com os resultados dos esforços de MPS [Peixoto 2012c]. A maioria dos dados é de grandes empresas que focam principalmente em melhorias de seus processos de gestão de projetos, de engenharia de requisitos, de revisão de software e no estabelecimento de um processo no nível organizacional. Isto envolve principalmente as áreas de processo dos níveis de maturidade 2 e 3 do CMMI. Considerando todos os tipos de organizações, CMM/CMMI são os modelos de melhoria mais utilizados. Nós observamos que as iniciativas de MPS não trazem mudanças "dramáticas" (os trabalho relatam melhorias entre 5 e 35%), e as principais razões para iniciar um esforço de MPS consistem (i) na redução da taxa de defeitos detectados durante o desenvolvimento e após a entrega do produto, (ii) na melhoria da produtividade, (iii) na redução do tempo de entrega e (iv) na redução dos custos. Um percentual considerável de dados originou-se de entrevistas, observações e questionários, revelando que a maioria das organizações possuem dificuldades em medir e analisar a melhoria quantitativamente. Adicionalmente, os estudos apresentaram limitações em termos de rigor, credibilidade e validade em seus achados. Eles não fornecem informação suficiente sobre o contexto da organização de desenvolvimento de software e a correlação entre a melhoria do processo e o efeito nos processos da organização, o que nos impossibilitou de produzir uma melhor avaliação.

Além desta análise, nós coletamos 123 regras de Engenharia de Software de livros textos e de outros jogos de simulação. Este conjunto inclui regras que são dependentes de valores que variam de acordo com a aplicação e regras que estão além do escopo de Engenharia de Software, incluindo um escopo maior de processos de negócio. Deste conjunto, nós selecionamos as regras relacionadas às áreas de processo do SPIAL, resultando em 57 regras de Engenharia de Software. Essas regras foram utilizadas para ensinar as melhores práticas de Engenharia de Software para os estudantes, premiando ou penalizando suas ações.

**d) Desenvolvimento do SPIAL**: Baseado nas etapas conduzidas anteriormente, nós identificamos requisitos essenciais para o nosso jogo de simulação, os quais foram utilizados na definição do comportamento do SPIAL. Além disso, nós também avaliamos componentes para o desenvolvimento do SPIAL. Como nós não encontramos componentes que eram genéricos o suficiente para serem reutilizados, nós decidimos desenvolver o nosso próprio *framework*, denominado FASENG (*Framework for Software Engineering Simulation Games*) [Peixoto e outros 2012a]. FASENG é um *framework* para o desenvolvimento de jogos de simulação de Engenharia de Software. Ele foi desenvolvido baseado em um conjunto de requisitos relacionados com a aplicação das teorias de aprendizado. FASENG é composto de três componentes que podem ser reutilizados na criação de novos jogos de simulação - modelo simulação, simulador e máquina de simulação. O **modelo de simulação** representa aspectos das organizações desenvolvedoras de software e os elementos que serão simulados e apresentados ao jogador. O modelo de simulação é implementado em um arquivo XML e possui um conjunto de construções que são definidas para cada modelo a ser simulado como, por exemplo, tipos de objeto, ações e regras. O comportamento do modelo de simulação é representado por um modelo matemático que pode ser alterado de acordo com a aplicação do jogo [Peixoto e outros 2013]. O **simulador** é o elemento interativo responsável por controlar os passos da simulação do modelo, percorrendo iterativamente suas equações para calcular o comportamento de cada elemento do sistema. A **máquina**

**de jogo** é o elemento com que o jogador interage e recebe realimentação visual dos resultados da simulação.

**e) Avaliação do SPIAL:** Finalmente, nós avaliamos o SPIAL de dois pontos de vista: especialista e jogador. Uma inspeção utilizando o MIS foi conduzida por um especialista e rupturas de comunicação foram identificadas e corrigidas. Além disso, nós avaliamos o jogo realizando um experimento piloto com 11 estudantes de graduação de um curso de Engenharia de Software. Na média, os estudantes acharam o jogo bastante agradável e eles se divertiram ao jogar. Eles concordam em adotar este jogo em um curso introdutório de Engenharia de Software como uma abordagem complementar. Na perspectiva dos alunos, eles aprenderam e conseguiram aplicar conceitos básicos com o jogo. Além disso, reforçaram vários conceitos apresentados em sala de aula.

## 6. Conclusão

Este artigo resumiu o trabalho de tese que consistiu na criação de um jogo de simulação para dar suporte ao ensino de Engenharia de Software e Melhoria de Processos de Software. Este jogo encontra-se disponível no site da autora[3]. A qualidade do trabalho é atestada pelas publicações resultantes da tese. Além disso, este trabalho ficou entre as seis melhores teses do CTD-CSBC 2013[4] dentre 34 teses inscritas. Além de ter tido um desempenho acadêmico excelente durante o seu doutorado, a autora ainda teve a oportunidade de fazer o estágio sanduíche de 11 meses na *Vrije University*, Holanda. Dois motivos garantem destaque a este trabalho de tese: inovação e qualidade. A inovação ao propor um jogo sobre melhoria de processo de software e identificar questões de melhoria de software na literatura. A qualidade evidenciada pelos trabalhos publicados em veículos nacionais e internacionais.

## Publicações Resultantes da Tese

Peixoto, Daniela C. C, Resende, Rodolfo S., and Pádua, Clarindo Isaías P. S. (2013) An Educational Simulation Model Derived from Academic and Industrial Experiences. In: 43rd Frontiers in Education Conference (FIE'13), Oklahoma City, Oklahoma, USA, October 23-26, To appear.

Peixoto, Daniela C. C. Possa, Rodrigo M., Resende, Rodolfo S., and Pádua, Clarindo Isaías P. S. (2012a) FASENG: A Framework for Development of Software Engineering Simulation Games. In: 42nd Frontiers in Education Conference (FIE'12), Seattle, Washington, USA, October 3-6.

Peixoto, Daniela C. C., Possa, Rodrigo M., Resende, Rodolfo S., and Pádua, Clarindo Isaías P. S. (2012b) Challenges and Issues in the Development of a Software Engineering Simulation Game. In: 42nd Frontiers in Education Conference (FIE'12), Seattle, Washington, USA, October 3-6.

Peixoto, Daniela C. C. (2012c) SPIAL: A Tool for Software Process Improvement Training. PhD Thesis, Universidade Federal de Minas Gerais, Brasil, setembro.

Meirelles, Lucas; Peixoto, Daniela; Monsalve, Elizabeth; Figueiredo, Eduardo; Werneck, Vera; Leite, Julio; Resende, Rodolfo; e Pádua, Clarindo (2011a) Uso de Jogos para o Ensino de Engenharia de Software. In: IV Fórum de Educação em Engenharia de

[3] http://www.dcc.ufmg.br/~cascini

[4] http://www.ic.ufal.br/csbc2013/noticias/ctd

Software, XXV Simpósio Brasileiro de Engenharia de Software, São Paulo-Brasil, 26 a 30 setembro.

Peixoto, Daniela C. C., Possa, Rodrigo M., Resende, Rodolfo S., and Pádua, Clarindo Isaías P. S. (2011b) An Overview of the Main Design Characteristics of Simulation Games in Software Engineering Education. In: 24th IEEE-CS Conference on Software Engineering Education and Training, Waikiki, Honolulu, Hawaii, May 22-24, pp. 101-110.

Peixoto, Daniela C. C., Batista, Vitor A., Resende, Rodolfo S., and Pádua, Clarindo Isaías P. S. (2010a) Learning from Students' Mistakes in Software Engineering Courses. In: Frontiers in Education (FIE'10) October 27-30, Virginia, USA, IEEE.

Peixoto, Daniela C. C., Batista, Vitor A., Resende, Rodolfo S., and Pádua, Clarindo Isaías P. S. (2010b) How to Welcome Software Process Improvement and avoid Resistance to Change. In: International Conference on Software Process (ICSP'10) July 8-9, Paderborn, Germany, pp. 138-149.

Peixoto, Daniela C. C., Batista, Vitor A., Resende, Rodolfo S., and Pádua, Clarindo Isaías P. S. (2010c) A Case Study of Software Process Improvement Implementation. In: 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE'10) July 1-3, Redwood City, San Francisco Bay, USA, pp. 716-721.

Peixoto, Daniela C. C., Prates, Raquel O., and Resende, Rodolfo S. F. (2010d) Semiotic Inspection Method in the Context of Educational Simulation Games. In: Proceeding of the 25th Annual ACM Symposium on Applied Computing (SAC'10) Sierre, Switzerland, pp. 1207-1212.

Peixoto, Daniela C. C., Batista, V. A., Rocha, G. M., Pádua, C. I. P. S., e Resende, R. S. F. (2008) Uma Experiência de Melhoria de Processo utilizando a Análise Causal de Defeitos. In: VII Simpósio Brasileiro de Qualidade de Software (SBQS'08) Florianópolis.

Peixoto, Daniela C. C., Pádua, C. I. P. S., Veloso, E. A., e Resende, R. S. F. (2007) Prevenção de defeitos em Requisitos de Software: Uma caracterização do processo de melhoria. In: VIII Simpósio Internacional de Melhoria de Processo de Software (SIMPROS'07) São Paulo.

## Referências

Barros, M. D. O., Werner, C. M. L., and Travassos, G. H. (2002) A system dynamics metamodel for software process modeling. Software Process: Improvement and Practice, 7(3-4):161–172.

Boehm, B., and Huang, L.G. (2003) Value-Based Software Engineering: A Case Study. IEEE Computer, Vol. 36, pp. 33-41

Chen, W., Wu, W., Wang, T. and Su, C. (2008) A Game-based Learning System for Software Engineering Education. In Proceedings of the 38th ASEE/IEEE Frontiers in Education Conference, pp. T2A-12-T2A-13, Saratoga Springs, New York, USA.

CMMI. (2010) CMMI for Development, Version 1.3. Technical Report CMU/SEI-2010-TR-033, Software Engineering Institute.

Cordova, D. I. and Lepper, M. R. (1996) Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice. Journal of Educational Psychology, Vol. 88, pp. 715-730

de Souza, C. S. (2005). The Semiotic Engineering of Human-Computer Interaction. The MIT Press.

Drappa, A. and Ludewig, J. (2000) Simulation in Software Engineering Training. In Proceedings of the 22nd International Conference on Software Engineering, pp. 199–208.

Garris, R., Ahlers, R. and Driskel J. E. (2002) Games, Motivation, and Learning: A Research and Practice Model. Simulation & Game, Vol. 33 (4), pp. 441-467, December.

Glass, R. L. (1998) Software Runaways: Lessons Learned from Massive Software Project Failures. NL: Prentice Hall.

Hiltz, S. R. and Turoff, M. (2005) Education goes digital: The evolution of online learning and the revolution in higher education. Communications of the ACM, Vol. 48(10) October, pp. 59-64.

Jain, A. and Boehm, B. (2006) SimVBSE: Developing a Game for Value-Based Software Engineering. In Proceedings of the 19th Conference on Software Engineering Education and Training, pp. 103 –114, Oahu, Hawaii.

Knauss, E., Schneider, K., and Stapel, K. (2008) A Game for Taking Requirements Engineering More Seriously. In First International Workshop on Multimedia Requirements Engineering, pp. 22–26, Barcelona, Catalunya, Spain, IEEE Computer Society.

Navarro, E. O. (2006) SimSE: A Software Engineering Simulation Environment for Software Process Education. PhD thesis, Donald Bren School of Information and Computer Sciences, University of California, Irvine.

Pfahl, D., Klemm, M., and Ruhe, G. (2001) A CBT module with integrated simulation component for software project management education and training. Journal of Systems and Software, pp. 283-298.

Ricci, K. Salas, E. and Cannon-Bowers, J. A. (1996) Do computer-based games facilitate knowledge acquisition and retention? Military Psychology, Vol. 8 (4), pp. 295-307.

Shaw, K. and Dermoudy, J. (2005) Engendering an Empathy for Software Engineering. In Proceedings of the 7th Australasian conference on Computing education - Volume 42, ACE '05, pp. 135–144, Darlinghurst, Australia, Australian Computer Society, Inc.

Vega, K. C., Fuks, H., and de Carvalho, G. R. (2009) Training in Requirements by Collaboration: Branching Stories in Second Life. In Simpósio Brasilerio de Sistemas Colaborativos, pp.116–122, Fortaleza, Ceará, Brazil. IEEE Computer Society.

Ye, E., Liu, C., and Polack-Wahl, J. (2007) Enhancing software engineering education using teaching aids in 3-D online virtual worlds. In Proceedings of the 37th ASEE/IEEE Frontiers in Education Conference, FIE '07, pp. T1E–8–T1E–13, Milwaukee, Wisconsin, USA.