

SimMS - Um Jogo Educacional de apoio ao Ensino de Manutenção de Software

Diógenes Dias Simão, Dyeimys Franco Correa, Paulo Afonso Parreira Júnior

Universidade Federal de Goiás (UFG) – Regional Jataí – Jataí/GO – Brasil

dio.ds@outlook.com, dyeimys@hotmail.com, paulojunior@jatai.ufg.br

Abstract. This paper presents a DEG, SimMS, for teaching Software Engineering, with emphasis on the IEEE 1219 standard for Software Maintenance. Its aim is to provide a different approach for teaching this subject in order to attract the attention of the students to the concepts of this area. In an evaluation conducted with twelve students of Computer Science, it was possible to notice that SimMS has obtained good marks regarding to its usefulness and easiness of use, respectively.

1. Introdução

Engenharia de Software (ES) é uma área que se preocupa com os aspectos da produção de um software, desde o levantamento de seus requisitos junto aos *stakeholders* até a fase de manutenção deste software [Sommerville 2011]. No contexto acadêmico, ES é uma disciplina que contempla grande variedade de termos e conceitos e, por isso, segundo Von Vangenheim e Von Vangenheim (2012), a metodologia mais utilizada para ensino desta disciplina é a ministração de aulas expositivas. Entretanto, pesquisas recentes têm apontado que aulas expositivas raramente satisfazem os desafios relacionados à aprendizagem [Biggs e Tang 2011]. Segundo os autores, quando presentes em aulas apenas expositivas, geralmente, os alunos perdem a concentração em cerca de 15 minutos e a absorção do conteúdo reduz drasticamente.

Neste sentido, alguns pesquisadores têm proposto a utilização de Jogos Educacionais Digitais (JEDs), como material de apoio ao processo de ensino-aprendizado [Fernandes e Werner 2009; Savi e Ulbricht 2009; Thiry *et al.* 2010; Navarro e Hoek 2009; Prikladnicki *et al.* 2007; Benitti e Molléri 2008]. JEDs podem ser vistos como ferramentas computacionais utilizadas como material didático complementar no aprendizado de alguma área ou de um conjunto de habilidades específicas [Annetta 2010]. A utilização de JEDs tem se mostrado eficiente, pois permite a simulação de ambientes que expõem os alunos a cenários reais, que são difíceis de se reproduzir em sala de aula [Lima *et al.* 2011].

Na área de ES, os JEDs possuem enfoque em simulações, explorando abstrações, trabalho cooperativo, tomadas de decisão, entre outros [Fernandes e Werner 2009]. Na literatura, há propostas de JEDs para diferentes áreas da ES, tais como Engenharia de Requisitos [Thiry *et al.* 2010], Medição de Software [Von Wangenheim *et al.* 2008] e Gerência de Projeto [Navarro e Hoek 2009; Prikladnicki *et al.* 2007; Benitti e Molléri 2008]. Entretanto, há escassez de trabalhos com enfoque em Manutenção de Software (MS). MS é uma das principais subáreas da ES e tem sido considerada, muitas vezes, como um gargalo no processo de desenvolvimento de um software; segundo Pressman (2011), dados da indústria indicam que entre 50% e 70% de todo esforço gasto em um software são despendidos após ele ter sido entregue ao cliente, ou seja, durante a fase de manutenção.

Considerando a escassez de trabalhos relacionados ao ensino de MS e a importância dessa fase para o ciclo de vida de um software, este trabalho tem como objetivo apresentar um JED de apoio ao ensino de MS, denominado *SimMS* (*Simulador de Manutenção de Software*). Mais especificamente, o jogo proposto visa a auxiliar alunos quanto ao entendimento dos conceitos relacionados ao processo de Manutenção de Software proposto pela norma IEEE 1219 [IEEE 1998].

Este trabalho está organizado da seguinte forma: na Seção 2 é apresentada uma breve contextualização sobre o processo de MS descrito na norma da IEEE 1219, bem como sobre JEDs. Na Seção 3 são apresentados alguns trabalhos relacionados, juntamente com uma análise comparativa dos mesmos. Nas Seções 4 e 5 são apresentados, respectivamente, o JED *SimMS* e a avaliação realizada sobre o mesmo. Por fim, na Seção 6 estão as considerações finais e as propostas de trabalhos futuros.

2. Manutenção de Software e Jogos Educacionais Digitais (JEDs)

Na Figura 1 são apresentadas as descrições das sete fases do processo para Manutenção de Software descrito na norma IEEE 1219 [IEEE 1998].

1 identificação da modificação solicitada	2 análise de viabilidade	3 projeto	4 implementação	5 teste de regressão	6 teste de aceitação	7 entrega
As modificações são classificadas, atribuindo-se um rank de prioridade a elas. Além disso, elas devem ser classificadas como manutenção corretiva, adaptativa, etc.	Utiliza-se as informações retiradas do pedido da fase anterior, juntamente com a documentação do projeto e então é estudada a viabilidade da realização da mudança solicitada	Os resultados das fases anteriores são utilizados para a atualização da documentação de projeto do software.	O novo projeto do software, o código fonte atual e a nova documentação são utilizados para conduzir a implementação da mudança solicitada.	Testes são realizados a fim de se verificar se o novo código provocou efeitos indesejados nos demais módulos do software.	Teste feito com o software totalmente integrado. Este teste deve ser realizado pelos usuários do software em geral ou apenas pelos usuários que trabalham com a parte modificada do software.	São descritas as instruções de funcionamento do software modificado; caso necessário é oferecido um treinamento aos usuários interessados.

Figura 1. Processo para Manutenção de Software segundo a norma IEEE 1219.

De acordo com esta norma, observa-se que várias fases devem ser seguidas e documentadas durante o processo de MS. Segundo Paduelli (2007), uma proposta interessante para que conteúdos como este sejam melhor explorados em sala de aula é fundamentá-los por meio da simulação de situações do mundo real. Uma abordagem que tem sido bastante utilizada para se atingir esse objetivo é a utilização de Jogos Educacionais Digitais (JEDs). A norma IEEE 1219 foi escolhida para ser contemplada neste trabalho, pois se trata de um padrão internacional, definido por uma instituição amplamente reconhecida e que tem sido abordada em alguns livros-textos recentes para a disciplina Engenharia de Software [Wazlawick 2013].

Os Jogos Digitais fazem parte de um dos ramos de entretenimento que mais cresce na indústria de software. Com o intuito de atrair a atenção e o comprometimento dedicados aos jogos por seus usuários para as salas de aulas, o número de pesquisas sobre a junção entre entretenimento e ensino com jogos educacionais tem crescido bastante [Savi e Ulbricht 2009; Fernandes e Werner 2009; Mitchell e Savill-Smith 2004]. Porém, realizar esta junção de forma adequada não é uma tarefa trivial. Para que os jogos possam atingir seus objetivos pedagógicos, alguns elementos devem estar presentes nos mesmos, tais como [Annetta 2010]: i) **identidade**: é importante que o jogador tenha uma identidade dentro do jogo, sendo representado por um personagem (*avatar*); ii) **interatividade**: é o elemento que permite a interação do jogador com outros personagens em ambientes de múltiplos jogadores ou com personagens *non-player* (*NPC – Non-Player Character*); iii) **complexidade crescente**: o jogo deve apresentar diferentes níveis de dificuldades, que aumentam conforme o jogador vai desempenhando corretamente o seu papel dentro do jogo; e iv) **análise de atuação**: é

necessário que o jogador receba um *feedback* sobre o seu desempenho durante e após a finalização do jogo. Na Seção 3 são apresentados sucintamente alguns JEDs de apoio ao ensino de ES existentes na literatura e, ao final, uma análise comparativa destes jogos é apresentada, com base nos elementos descritos anteriormente.

3. Trabalhos Relacionados

O primeiro trabalho relacionado a ser destacado consiste no JED “Ilha dos Requisitos” [Gros 2003], cuja história se baseia em um personagem, denominado “Jack Reqs”, que sofre um acidente de avião e cai em uma ilha isolada. Na ilha, há uma tribo de canibais e “Jack Reqs” precisa concluir uma série de tarefas relacionadas à área de Engenharia de Requisitos (ER) para que não seja devorado por estes canibais. Um exemplo de tarefa que deve ser realizada no jogo é colocar na ordem correta as principais atividades de um processo de ER.

“Planager” [Prikladnicki *et al.* 2007] é um jogo que apoia o ensino dos conceitos de gerência de projeto com enfoque no PMBOK (*Project Management Body of Knowledge*). Durante o jogo, o aluno pode interagir com diversos cenários de Projetos de Software pré-cadastrados. Em cada cenário, existe uma descrição de um projeto de software e o jogador deve passar pelos cinco processos de planejamento do PMBOK (definição do escopo, criação da estrutura analítica do projeto, definição das atividades, sequenciamento de atividades e desenvolvimento do cronograma). O jogo oferece dois módulos, um voltado para o aluno e outro para o professor. No módulo do professor, é possível cadastrar diferentes tipos de cenários de gerenciamento de projetos.

“SE-RPG” [Benitti e Molléri 2008] é um jogo desenvolvido no formato RPG (*Role-Playing Game*) que simula um ambiente de desenvolvimento de software. No “SE-RPG”, é possível que o jogador escolha um *avatar*, o projeto a ser desenvolvido, o modelo de processo de desenvolvimento (cascata, iterativo ou prototipação), a linguagem de programação a ser utilizada e a equipe de desenvolvimento. Após isso, o jogador deverá atribuir tarefas aos membros da sua equipe, de acordo com o modelo de processo escolhido, com o intuito de finalizar o projeto no menor tempo e com o menor custo possível. Nesta mesma linha, “SimSE” [Navarro e Hoek 2009] é um jogo no qual o jogador assume o papel de um gerente de projeto. No “SimSE”, é permitido ao jogador demitir ou admitir novos empregados, bem como atribuir diferentes tarefas a eles. O jogo permite ainda a comunicação dos membros da equipe para com o jogador, uma vez que eles podem demonstrar sua satisfação ou insatisfação, informar que estão doentes ou cansados; exigindo assim, que jogador tome decisões para manter o gerenciamento do projeto em ordem.

Por fim, “X-Med” [Von Wangenheim *et al.* 2008] é um jogo no qual o jogador assume o papel de um analista de medição, sendo responsável por realizar a simulação de um programa de medição de software, com base na abordagem GQM (*Goal-Question-Metric*). O objetivo do jogo é apoiar o ensino da competência de aplicação do conhecimento de medição de software.

No Quadro 1 apresenta-se uma análise comparativa dos JEDs descritos anteriormente, com base nos elementos que devem estar presentes em um jogo educacional [Annetta 2010] (Seção 2), bem como nos seguintes critérios adicionais: i) *Tópico de Ensino*: quais são os assuntos da ES tratados pelo JED?; ii) *Plataforma de Execução*: em quais tipos de ambientes o JED proposto pode ser executado (por exemplo, *web*, *desktop* ou *mobile*)?; e iii) *Tecnologia de Desenvolvimento*: em qual plataforma de desenvolvimento o JED foi construído?

Quadro 1. Análise comparativa dos trabalhos relacionados.

Critérios/JEDs	Ilha dos Requisitos	Planager	SE-RPG	X-Med	SimSE
<i>Identidade</i>	-	X	+	X	X
<i>Interatividade</i>	X	X	+	X	+
<i>Complexidade Crescente</i>	X	+	+	X	+
<i>Análise de Atuação</i>	X	+	X	+	X
<i>Tópico de Ensino</i>	Engenharia Requisitos	Gerência de projeto	Gerência de projeto	Medição de Software	Gerência de projeto
<i>Plataforma do JED</i>	Web	Desktop	Web	Desktop	Desktop
<i>Tecnologia de Desenvolvimento</i>	Flash	Java	Flash	Java	Java

Legenda: Elemento Identificado (+), Elemento parcialmente identificado (-), Elemento não identificado (X).

Como pode ser observado no Quadro 1, nenhum dos jogos analisados contempla o ensino de Manutenção de Software, nem contempla, por completo, os critérios apresentados por Annetta (2010) – Seção 2. Outro ponto a ser destacado é que apenas dois jogos foram desenvolvidos para serem executados via web (“Ilha dos Requisitos” e “SE-RPG”) e que a tecnologia utilizada para desenvolvê-los é proprietária (*Macromedia Flash*). O JED que contemplou mais elementos educacionais foi o “SE-RPG” e o que contemplou menos elementos, foi o JED “Ilha dos Requisitos”. No caso do jogo “Ilha dos Requisitos”, o elemento “Identidade” foi dado como parcialmente contemplado, pois apesar de o jogo fornecer um personagem que identifica o jogador (*Jack Reqs*), não há representação de identidade para usuários do gênero feminino. Os elementos menos contemplados pelos JEDs analisados foram “Identidade”, “Interatividade” e “Análise de Atuação”, elementos estes fundamentais para que o jogo atinja seus objetivos educacionais.

4. *SimMS* – Simulador de Manutenção de Software

SimMS é um JED *single-player* cujo objetivo é apoiar o ensino de Manutenção de Software, com ênfase no processo descrito pela norma IEEE 1219 (IEEE, 1998). O jogo simula o ambiente de uma empresa de software, que implementa requisições de manutenção solicitadas por um cliente em um software específico. A meta do jogo é atender às requisições de manutenção previamente cadastradas, de acordo com a norma. *SimMS* foi desenvolvido sobre a plataforma Java, que foi escolhida por: i) apresentar alta portabilidade; ii) permitir que produtos desenvolvidos nesta linguagem possam ser disponibilizados para serem executados via web (por meio da tecnologia de *Applets*); e iii) ser constantemente atualizada. Cabe ressaltar ainda que *SimMS* é software livre e encontra-se disponível para *download* via web¹.

4.1. Apresentação do Jogo

Antes de começar a jogar, o jogador tem a opção de abrir um tutorial, no qual são apresentadas as regras do jogo, bem como os principais conceitos sobre MS e sobre a norma IEEE 1219. Após iniciar o jogo, como primeiro passo, o jogador deve informar seu nome e escolher um *avatar* (do gênero masculino ou feminino) para representá-lo. O(A) jogador(a) assume então o papel de um(a) gerente de equipe de manutenção, responsável por selecionar as tarefas a serem realizadas, a ordem em que elas devem ocorrer e os funcionários que as cumprirão.

Após se identificar, o jogador deverá escolher os membros da sua equipe de manutenção, que deve ser composta por quatro funcionários (Figura 2). Cada

¹ <http://paulojunior.jatai.ufg.br/pages/49922-trabalho-de-conclusao-de-curso>

funcionário possui experiências (em anos) distintas para os diferentes tipos de atividades relacionadas ao desenvolvimento de software, tais como análise, projeto, implementação e teste. Feita a escolha da equipe, o jogo entra em sua tela principal, apresentada na Figura 3. De tempos em tempos, requisições de manutenção irão chegar e o jogador deverá tratá-las corretamente, conforme explicado mais a frente nesta seção. Há um conjunto de requisições pré-cadastradas no jogo e a cada partida, no máximo cinco requisições deverão ser tratadas pelo jogador. Essa decisão foi tomada para que o jogo não se prolongasse muito e para que o jogador pudesse entender com mais clareza os resultados de suas decisões sobre cada requisição de manutenção recebida.



Figura 2. Escolha da equipe de manutenção.



Figura 3. Escolha do avatar.

Como pode ser observado na Figura 3 - A, a primeira requisição de manutenção do cliente foi: “Olá. Preciso de uma tela para cadastrar os veículos dos médicos que podem usar o estacionamento. Obrigado!”. A descrição da requisição do cliente é bastante limitada, porém seu objetivo é apenas permitir que o jogador reconheça o tipo de manutenção (perfectiva, corretiva ou adaptativa) com o qual se refere esta requisição. O software que está sob manutenção é o “MedPro”, um sistema de informação hipotético relacionado à área da saúde. O jogador pode ter acesso a uma descrição mais detalhada deste sistema por meio do ícone “MedPro” (Figura 3 – B). Além desta descrição, o jogo apresenta também uma representação fictícia da qualidade da documentação e da taxa de erros do software em manutenção, que podem variar de 0 à 100% (Figura 3 - C). Essas informações são importantes, pois podem direcionar a tomada de decisão do jogador ao longo da partida. Por exemplo, se a legibilidade do código fonte do software é baixa, o jogador pode decidir por realizar uma manutenção preventiva antes de atender a qualquer tipo de requisição do cliente.

Uma vez que o jogador leu a requisição do cliente, ele deverá selecionar a tarefa a ser executada e um funcionário para realizá-la. Cada atividade do processo da norma IEEE 1219 possui uma tarefa relacionada no sistema. Além disso, há uma atividade extra, denominada “manutenção preventiva”, que permite aprimorar a qualidade da documentação do software. De acordo com a norma IEEE 1219 [IEEE 1998], o primeiro passo do processo de MS é a “identificação da modificação solicitada”, que consiste em saber se a requisição é do tipo corretiva, adaptativa ou perfectiva. O exemplo da Figura 3 (A) trata de uma requisição do tipo “perfectiva”, uma vez que ela se refere à inclusão de uma nova função no software. A cada atividade realizada pelo jogador, como “ranqueamento”, “análise da viabilidade”, entre outras, dois eventos irão ocorrer: i) atualização da qualidade da documentação e da taxa de erros do software em

manutenção; e ii) atualização da quantidade de dias em que o software está sob manutenção (Figura 3 – D).

A atualização a ser realizada na qualidade da documentação do projeto, na sua taxa de erros e o tempo para execução de uma atividade dependem da experiência do funcionário que está realizando a tarefa e da qualidade atual da documentação do software legado. Por exemplo, se o usuário selecionar um funcionário com pouca experiência em testes para executar a atividade de “testes de aceitação”, a taxa de erros do software irá aumentar e o tempo para execução desta tarefa será maior do que se um funcionário mais experiente tivesse sido escolhido. Outro aspecto importante a ser observado pelo jogador é que a ordem em que as atividades são realizadas deve estar em conformidade com o que é especificado na norma IEEE 1219. Um exemplo de como a não realização de uma atividade, ou a realização de uma atividade na ordem incorreta, pode prejudicar o desempenho do jogador é quando o mesmo entrega o software sem ter realizado a atividade de “teste de aceitação”. Caso isso ocorra, a taxa de erros do software poderá aumentar após a entrega do software ao cliente. Devido à limitação de espaço, outras características do jogo não puderam ser apresentadas. Mais detalhes sobre o *SimMS* podem ser encontrados em [Simão 2013].

4.2 Análise Comparativa do SimMS

Com relação aos elementos educacionais da Seção 2, tem-se que: quanto ao elemento “identidade”, no *SimMS*, o(a) jogador(a) é representado(a) por um(a) gerente de uma equipe de MS, cujo gênero é escolhido pelo próprio usuário. Apesar de o *SimMS* não fornecer interatividade com outros jogadores, o elemento “interatividade” é contemplado por permitir que o jogador interaja com os personagens *non-players*, representados pelos membros da equipe de manutenção. Quanto ao elemento “complexidade crescente”, o grau de dificuldade do jogo varia de acordo com os níveis de qualidade da documentação e da taxa de erros do software; quanto pior a qualidade do software, mais tarefas o jogador terá que realizar para completar as requisições com êxito. Quanto ao elemento “análise de atuação”, ao final de cada requisição entregue ao cliente, é fornecido um *feedback* informando as atividades do processo que foram executadas corretamente ou não pelo jogador. Além disso, ao longo do jogo, o usuário pode observar diretamente o reflexo de suas ações sobre o tempo gasto para realização de cada tarefa e sobre a qualidade da documentação e taxa de erros do software.

5. Avaliação do SimMS

Para avaliação do *SimMS*, o modelo de aceitação de tecnologia *TAM* (*Technology Acceptance Model*) [Davis *et al.* 1989] foi utilizado. Esse modelo possui como objetivo explicar o comportamento das pessoas no que diz respeito à aceitação de uma tecnologia. O modelo *TAM* define dois constructos básicos: i) *utilidade percebida*, que mede o quanto uma pessoa acredita que utilizar determinada tecnologia aumenta seu desempenho no trabalho ou estudo; e ii) *facilidade de uso percebida*, que mede o quanto uma pessoa acredita que o uso de determinada tecnologia é simples. Além disso, o modelo *TAM* sugere a criação de questionários, para os quais são atribuídas afirmações relacionadas à facilidade de uso e utilidade percebidas da tecnologia em análise. Para cada afirmação, o respondente poderá escolher uma opção na seguinte escala do tipo *Likert*: “DT - Discordo Totalmente”, “DT - Discordo Parcialmente”, “N - Neutro”, “CP - Concordo Parcialmente” e “CT - Concorde Totalmente”.

O questionário desenvolvido para avaliação do *SimMS* foi respondido por doze alunos de graduação em Ciência da Computação da Universidade Federal de Goiás (Regional Jataí), que passaram pela disciplina “Engenharia de Software”. Os resultados obtidos quanto à facilidade de uso e utilidade percebidas são sumarizados nos gráficos das Figuras 4 e 5, respectivamente. Com relação à facilidade uso percebida (Figura 4), pode-se destacar que 100% dos participantes da avaliação concordaram totalmente que o acesso ao jogo *SimMS* é fácil. Acredita-se que isso seja devido à escolha da plataforma de desenvolvimento (Java), por meio da qual, pode-se desenvolver um jogo que não requer instalação na máquina do usuário, facilitando assim, o acesso ao mesmo. Além disso, a grande maioria (92%) concordou que utilizar o *SimMS* é uma boa ideia. Outro ponto importante a ser destacado é que 41% dos avaliadores escolheram a opção “neutro” ou “discordo parcialmente”, quando questionados se mesmo antes de clicarem em um botão, eles sabiam a ação que iria ocorrer. Isso pode indicar que é preciso melhorar a interface do jogo, para que o jogador fique mais seguro de suas ações.

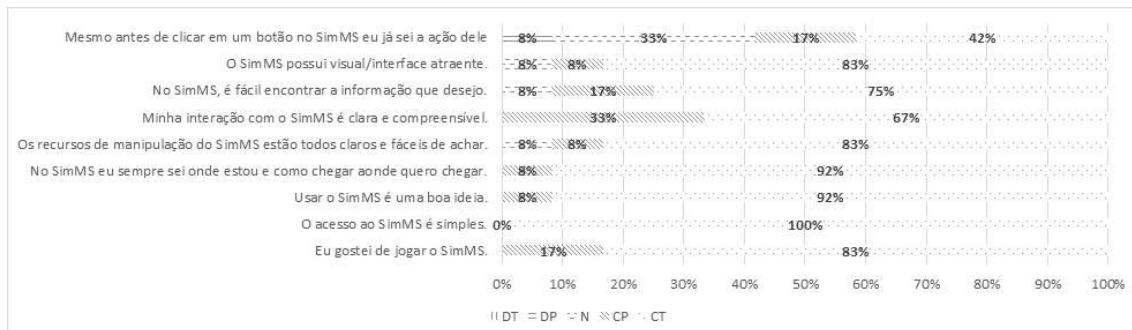


Figura 4. Resultados obtidos para o constructo “Facilidade de uso percebida”.

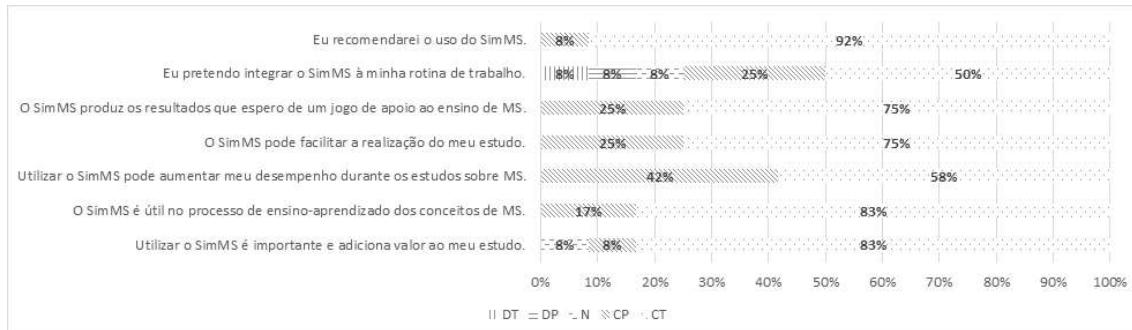


Figura 5. Resultados obtidos para o constructo “Utilidade percebida”.

Quanto à utilidade percebida, a maioria dos avaliadores (83%) concordaram totalmente que o *SimMS* pode ser útil no processo de ensino-aprendizagem dos conceitos de MS e pode adicionar valor ao seu estudo. Porém, 16% dos avaliadores discordaram total ou parcialmente, quando lhes foi questionado sobre a utilização do *SimMS* em sua rotina de trabalho. Isso é compreensível, pois o *SimMS* foi desenvolvido com a finalidade de fornecer apoio aos alunos das disciplinas envolvidas com MS. Por fim, quando perguntado se os avaliadores recomendariam o *SimMS* a outras pessoas, 92% responderam positivamente (concordo totalmente).

6. Considerações Finais e Trabalhos Futuros

Jogos Educacionais Digitais (JEDs) podem ser uma alternativa promissora como ferramenta de apoio ao processo de ensino-aprendizagem de várias disciplinas, dentre elas, a Engenharia de Software. O JED apresentado neste trabalho, *SimMS*, pode ser

utilizado como ferramenta de apoio ao ensino de Engenharia de Software, bem como outras disciplinas que abordem o tema “Manutenção de Software”.

Como propostas de trabalhos futuros, tem-se: i) adaptação do *SimMS* para torná-lo flexível a ponto de permitir que professores possam adicionar novas requisições e novas regras ao jogo; ii) implementação de novos tipos de estratégias de manutenção, como por exemplo, as utilizadas pelos métodos ágeis, como *Scrum*, *XP*, entre outros; e iii) implementação de técnicas de Inteligência Artificial para realização dos cálculos de duração dos eventos e pontuação dos usuários. Acredita-se que o jogo pode tornar-se mais realístico com essa melhoria, além disso, espera-se um aumento na complexidade do jogo com a inclusão de novas requisições e novas estratégias de execução da MS.

Referências

- ANNETTA, L. “The “I’s” Have It: A Framework For Serious Educational Game Design”. *Review Of General Psychology*. 2010.
- BENITTI, F. B. V.; MOLLÉRI, J. S. “Utilização de um RPG no Ensino de Gerenciamento e Processo de Desenvolvimento de Software”. XXVIII Congresso da SBC. 2008.
- BIGGS, J.; TANG, C. “Teaching For Quality Learning At University”. 4^a Edição. 480p. McGraw-Hill. 2011.
- DAVIS, F.D.; Bagozzi, R. P.; Warshaw P.R., “User Acceptance of Computer Technology: A Comparison of two Theoretical Models”. In: *Manag. Science*. v. 35, n. 8, p. 982-1003, 1989.
- FERNANDES, L.; WERNER, C. “Sobre o uso de Jogos Digitais para o Ensino de Engenharia de Software”. II Fórum de Educação em Engenharia de Software. 2009.
- GROS, B. “The Impact of Digital Games in Education”. *First Monday: Peer-reviewed Journal on the Internet*. 2003.
- IEEE Std. 1219-1998. “IEEE Standard for Software Maintenance”. Junho/1998.
- LIMA, T.; CAMPOS, B.; SANTOS, R.; WERNER, C.; Limoeiro, F. “Desenvolvimento de Jogos Educacionais para o Ensino de Engenharia de Software”. X SBGames, 2011.
- MITCHELL, A.; SAVILL-SMITH, C. “The Use of Computer and Video Games For Learning: A review of the literature”. 84p. Learning and Skills Development Agency. 2004.
- NAVARRO, E.; HOEK, A. “Multi-Site Evaluation of SIMSE”. 40th ACM Technical Symposium On Computer Science Education. 2009.
- PADUELLI, M. M. “Manutenção de Software: problemas típicos e diretrizes para uma disciplina específica”. Dissertação de Mestado. Universidade de São Paulo. 2007.
- PRIKLADNICKI, R.; ROSA, R.; KIELING, E. “Ensino de Gerência de Projetos de Software com o Planager”. XVIII Simpósio Brasileiro de Informática na Educação. 2007.
- SAVI, R.; ULRICH, V. R. “Jogos Educacionais: Benefícios e Desafios”. RENOTE - Revista Novas Tecnologias na Educação. 2009.
- SIMÃO, D. D. “SimMS – Um Jogo Educacional Digital de Apoio ao Ensino de Manutenção de Software”. Monografia de Graduação. UFG/Regional Jataí. Jataí/GO. 2013.
- SOMMERVILLE, I. “Engenharia De Software”. 9^a Edição. 568p. Pearson Education. 2011.
- THIRY, M.; ZOUCAS, A.; GONÇALVES, R. “Promovendo a Aprendizagem de Engenharia de Requisitos de Software através de um Jogo Educativo”. XXI Simpósio Brasileiro de Informática na Educação. 2010.
- VON WANGENHEIM, C. G.; VON WANGENHEIM, A. “Ensinando Computação com Jogos”. Bookess. 2012.
- VON WANGENHEIM, C. G.; THIRY, M.; KOCHANSKI, D. “Empirical Evaluation of an Educational Game on Software Measurement”. *Empirical Softw. Eng.* 14, 4, 418-452. 2008.
- WAZLAWICK, R. S. “Engenharia de Software: conceitos e práticas”. 368p. Campus. 2013.