Um Jogo para o Ensino de Engenharia de Software Centrado na Perspectiva de Evolução

Eduardo Figueiredo[1,2], Cidiane Lobato[1], Klessis Dias[1], Julio Leite[1], Carlos Lucena[1]

[1]Departamento de Informática, PUC-Rio, Rio de Janeiro, Brasil

[2]Computing Department, Lancaster University, Lancaster, UK

{emagno, cidiane, klessis, julio, lucena}@inf.puc-rio.br

Abstract. *Game technology has been used for learning in many educational areas, but there are few attempts in software engineering. To address such problem, this paper presents SimulES, an educational card game that simulates the software engineering process. SimulES comes after a well known game, named Problems and Programmers (PnP), and extends this previous game with software evolution concepts. Using SimulES, a student can take the role of a software project manager and deal with problems which are not sufficiently highlighted by traditional lectures. In addition, this paper also makes an extensive evaluation of PnP.*

Resumo. *A tecnologia de jogos tem sido usada em muitas áreas como ferramenta educacional, mas ainda é pouco comum no ensino de engenharia de software. Para minimizar este problema, este artigo apresenta SimulES, um jogo educacional de cartas que simula o processo de desenvolvimento de software. SimulES foi concebido a partir de uma extensa avaliação do jogo "Problems and Programmers" (PnP), acrescentando a este jogo conceitos de evolução de software. O jogo SimulES permite ao estudante assumir o papel de gerente de projeto e, desta forma, deparar com problemas que não são bem cobertos em aulas tradicionais.*

1. Introdução

O uso de jogos para estimular a curiosidade e prover motivação para o aprendizado é um tema amplamente pesquisado e discutido [deLaet et al. 2005] [Oh e Hoek 2001] [Virvou et al. 2005]. No entanto, na área de engenharia de software (ES) essa estratégia ainda é pouco explorada. Um curso típico de ES consiste de aulas em que conceitos teóricos são passados aos alunos e exercitados por atividades em pequenos exemplos práticos. Apesar de o professor poder explicar assuntos relacionados à gerência de projeto (tal como características humanas), certos problemas de grandes sistemas, com elevado número de pessoal, não são satisfatoriamente cobertos nas atividades práticas.

A motivação desta pesquisa partiu de um estudo no contexto de evolução de software. Hoje em dia, a evolução de software tem se tornado de vital importância devido ao processo incremental de desenvolvimento de sistemas complexos composto de múltiplos ciclos de retro-alimentação [Lehman, 1996]. Em nossa pesquisa, decidimos utilizar o jogo "Problems and Programmers" (PnP) [Baker et al. 2005] [Navaro et al. 2004] como artefato para aplicar conceitos de evolução de software. PnP foi selecionado por apresentar uma série de características interessantes a este estudo. Primeiro, o jogo é

bem conhecido e vem sendo aplicado com sucesso no ensino de ES [Baker et al. 2005]. Segundo, PnP emprega técnicas de ensino comuns a jogos educacionais similares, como por exemplo, simular um projeto de software [deLaet et al. 2005]. Além disso, a simulação adotada no jogo considera importantes fases do processo de desenvolvimento de software, tais como documentação, implementação, inspeção e testes. Entretanto, apesar de amplamente exercitado, o jogo PnP deixa uma importante lacuna por não considerar aspectos modernos de software, como desenvolvimento iterativo e evolução.

Desta forma, este artigo apresenta em detalhes uma avaliação do jogo PnP enfatizando os critérios que afetam a qualidade do jogo para o ensino de ES com ênfase em evolução. A avaliação discute também problemas relacionados a jogabilidade e dinamismo do jogo. Para os problemas identificados, propomos soluções que alteram o formato o jogo, suas regras e artefatos. Em adição, é apresentada uma nova versão do jogo, batizada de SimulES (Simulador de Uso da Engenharia de Software), que incorpora as soluções propostas. O novo jogo é apresentado de forma comparativa a PnP, i.e. baseado no uso de ambos os jogos e em observações referentes ao jogo original.

O restante deste artigo está organizado da seguinte forma. Na Seção 2 é definida a configuração deste estudo, sendo composta dos principais objetivos e de uma extensa avaliação de PnP. O novo jogo SimulES, originado de PnP, é proposto na Seção 3. Esta seção também apresenta uma argumentação de como os problemas identificados em PnP foram ou não sanados no novo jogo. A Seção 4 discute brevemente as limitações deste estudo bem como outros trabalhos e jogos educacionais, destacando as interfaces deste artigo com a literatura existente. Concluímos na Seção 5 com as principais contribuições e futuros desdobramentos desta pesquisa.

2. Contexto do Estudo

Esta seção apresenta os alicerces deste estudo traçando seus principais objetivos e os resultados da avaliação do jogo educacional “Problems and Programmers” (PnP). Tanto a avaliação quanto os objetivos são utilizados para apresentar o jogo proposto (Seção 3). PnP é um jogo de cartas direcionado ao ensino de ES, desenvolvido na Universidade da Califórnia e amplamente utilizado em diversas instituições. Seu objetivo é simular o processo de desenvolvimento de sistemas desde a concepção até a fase de entrega do software [Baker et al. 2005]. Este jogo é principalmente aplicável a estudantes com nível básico de conhecimento em disciplinas de ES, mas conceitos avançados também são apresentados no decorrer do jogo. A idéia dos autores é utilizar tal jogo como ferramenta de apoio durante um curso tipicamente semestral.

2.1. Objetivos

O primeiro passo de nossa pesquisa foi verificar que o jogo PnP praticamente não emprega conceitos de evolução de software. Decidimos então fazer um estudo detalhado deste jogo e, caso necessário, alterar suas regras e estrutura. Ao jogarmos PnP conforme instruções disponíveis no sítio [Problems and Programmers, 2007] identificamos diversos problemas detalhados na Subseção 2.2. Tais problemas podem ser classificados em duas categorias: (i) as técnicas de ensino são limitadas e não permitem aos jogadores adquirirem conhecimento externo ao jogo e (ii) muitos conceitos de engenharia de software utilizados são vagos e/ou obsoletos. Por exemplo, PnP baseia-se em uma abordagem clássica da ES na qual prevalece a visão de um processo de produção bastante linear, o que não reflete a realidade atual. Neste contexto, nossa contribuição

não está centrada apenas em tornar o jogo mais fácil de jogar, mas também em fazê-lo refletir práticas mais modernas de produção de software.

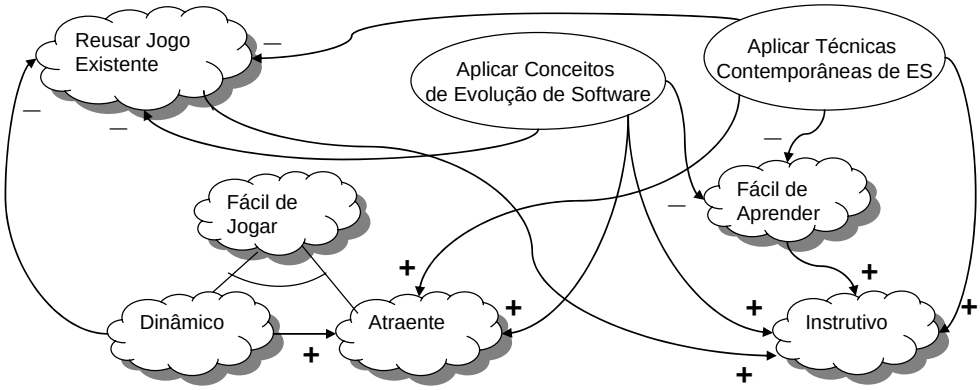


Figura 1. Principais objetivos do experimento

Este estudo possui dois objetivos bem definido, representados como elipses na Figura 1: (i) aplicar conceitos de evolução de software e (ii) utilizar fundamentos mais modernos de ES. Além disso, temos objetivos do tipo "*soft*" [Chung et al. 2000] que refletem requisitos de qualidade, tais como, jogo mais atraente, instrutivo, dinâmico, fácil de usar e fácil de aprender. A Figura 1 utiliza a notação base de modelagem orientada a metas de Mylopoulos, Chung e Yu [1999] para mostrar estes objetivos e suas interferências positivas ou negativas. Por exemplo, o objetivo *Reusar Jogo Existente* interfere negativamente nos seguintes objetivos: *Aplicar Conceitos de Evolução de Software*, *Aplicar Técnicas Contemporâneas de ES* e *Criar um Jogo Dinâmico*. Por outro lado, tal objetivo favorece (interfere positivamente) o objetivo de *Criar um Jogo Instrutivo.*

2.2. Avaliação do Jogo Original

O estudo dos conceitos e regras que sustentam o jogo PnP juntamente com o exercício prático de jogarmos várias vezes, simulando diferentes estratégias e com diversas quantidades de jogadores, nos permitiu identificar inúmeras vantagens e fraquezas deste jogo. Como ponto positivo, o exercício de simulação adotado em PnP auxilia o estudante a compreender lições de boas práticas de ES. Lições estas que certamente serão futuramente lembradas quando o aluno se deparar com um projeto real de software. Em adição, outra característica positiva é ensinar a partir de decisões tomadas de forma equivocada. Desta forma, os participantes entendem, ao fim de uma série de jogadas, quais caminhos devem ser seguidos para eles obterem sucesso em seus projetos.

Apesar de reconhecermos PnP como excelente ferramenta de ensino, qualidade comprovada pela sua aplicação em sala de aula [Baker et al. 2005] [Navaro et al. 2004], levantamos neste estudo questões que podem ser aprimoradas tanto na dinâmica do jogo quanto em seu propósito de simulação do mundo real. As principais limitações de PnP identificadas durante este estudo são listadas e detalhadas a seguir:

*Muito amarrado a um único processo de software (**Problema A**)*. O jogo segue um processo de desenvolvimento de software chamado Modelo Cascata [Royce, 1970] que, apesar de bem conhecido, não é muito utilizado atualmente. Novas abordagens têm sido propostas mais recentemente como o Modelo Espiral [Boehm, 1986] e a Programação Extrema (XP) [Beck, 1999]. Entretanto, o jogo não oferece liberdade ao jogador para escolha de seu próprio processo de desenvolvimento.

*Utilização de cartas muito abstratas e sem informações adicionais (**Problema B**).* No geral, as cartas do jogo não permitem aos jogadores uma clara interpretação de sua mensagem, talvez pela limitada quantidade de informação disponível (e.g. carta de projeto apresentada na Figura 2a). Desta forma, caso o jogador não entenda a mensagem educativa, o jogo não oferece ou aponta para fonte extra de informação.

*Jogo não desperta entusiasmo dos jogadores nas primeiras rodadas (**Problema C**).* Um jogo educativo, bem como qualquer categoria de jogo, deve motivar os participantes a alcançarem seus objetivos. Entretanto, o jogo PnP possui uma dinâmica inicial pouco interessante por não permitir que cartas de problemas sejam jogadas antes dos jogadores possuírem cartas de código. Desta forma, as rodadas iniciais se limitam à compra de cartas sem nenhuma interação entre os participantes.

*Número de jogadores não é limitado, mas a estrutura do jogo dificulta que mais de 4 pessoas participem (**Problema D**).* Quando 6 pessoas jogam PnP, identificamos que muitas cartas de problemas (até 5) são lançadas a cada jogador. Com este elevado número de problemas, torna-se difícil para qualquer participante evoluir e ganhar o jogo.

*Ausência de um tabuleiro para organização da área de projeto do jogador (**Problema E**).* Em nossa experiência, percebemos ser difícil manter as cartas dispostas de forma organizada na mesa. Em poucas rodadas, não era possível saber, por exemplo, se determinada carta era de requisitos ou de desenho; uma vez que estas são idênticas e não existe área definida para dispor tais cartas.

*Ausência de um mapeamento explícito entre os artefatos do jogo e os conceitos de ES aplicados (**Problema F**).* PnP foi proposto para ser usado de forma complementar a aulas tradicionais. Entretanto, o professor que o utiliza não possui um elo que conecte determinado conceito apresentado em aula ao conjunto de artefatos ou cenários do jogo.

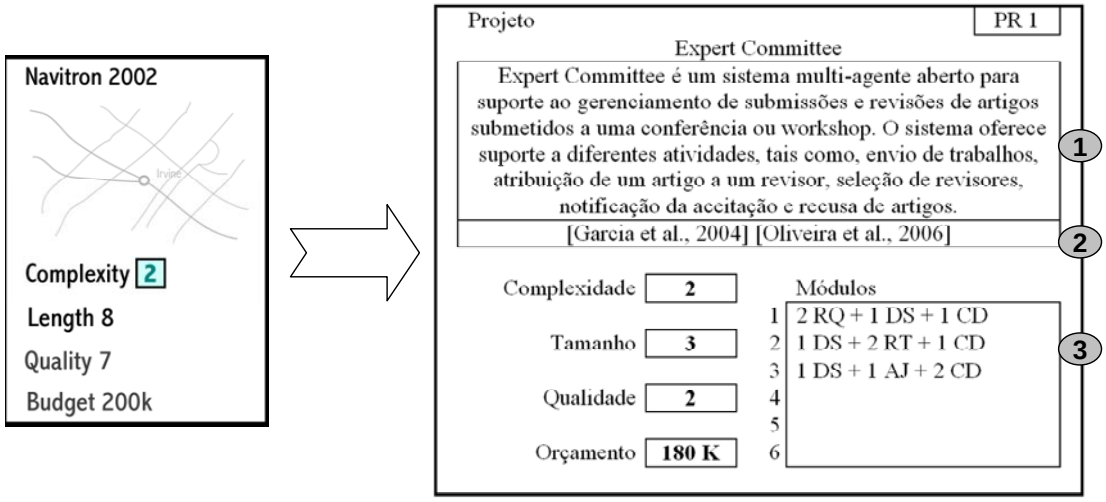
3. Solução Proposta: O Jogo SimulES

Esta seção apresenta o jogo SimulES para ensino de ES com foco em evolução. Este jogo procura resolver os problemas de PnP listados na seção anterior. Como em PnP, o objetivo de SimulES é que jogadores, idealmente alunos, disputem para terminar um projeto de software e o vencedor será aquele que primeiro entregar ao cliente um produto com qualidade adequada. Os recursos do jogo proposto são: cartões de projeto (Subseção 3.1), um tabuleiro (Subseção 3.2), cartas (Subseção 3.3) e um dado. Em relação à PnP, estes recursos foram totalmente reformulados. O dado e o tabuleiro não existiam no jogo anterior e foram incluídos em SimulES para aumentar o seu dinamismo e organizar a área de jogo (tratando os Problemas C e E discutidos na Seção 2.2), respectivamente. Em adição, as cartas foram atualizadas para conceitos e práticas mais modernas de ES e estendidas com informações adicionais (Problema B). Por restrições de espaço, o jogo SimulES é apresentado de forma a contrastar suas características às do jogo predecessor (PnP), informações mais detalhadas sobre suas regras e artefatos encontram-se disponíveis em [Figueiredo et al. 2006] [SimulES, 2007].

3.1. Cartões de Projetos

Os cartões de projetos são muito simples em PnP (Figura 2a) o que motivou a criação de projetos mais bem elaborados de tal forma a deixar a simulação mais realística. Três novos tipos de informações foram adicionados a estes cartões: (1) uma breve descrição textual do projeto, (2) referências para bibliografia relacionada ao tema do cartão e (3) a

forma de composição dos módulos que integram o projeto. Estas informações são descritas a seguir e ilustradas no exemplo de projeto do jogo SimulES apresentado na Figura 2b. Tal projeto descreve um sistema multi-agentes para gerência e automação do processo de revisão de eventos científicos [Garcia et al. 2004] [Oliveira et al. 2006].



(a) PnP (b) SimulES

Figura 2. Exemplos de cartões de projeto (a) no jogo original e (b) em SimulES

1. *Descrição.* A descrição do projeto é um texto em linguagem natural que informa suas principais características. Ela foi adicionada ao cartão para tornar o jogo mais realista e ajudar o jogador a compreender os principais requisitos do sistema (trata o Problema B, Seção 2.2).
2. *Referências*. Abaixo da descrição são colocadas citações para trabalhos publicados (e.g. artigos, livros) relacionadas ao projeto ou a seus principais conceitos educacionais. Uma listagem detalhada das referências também é um artefato em SimulES devendo ser usada pelo professor para incentivar o aluno a buscar informações externas ao jogo (trata os Problemas B e F).
3. *Módulos*. O atributo *Tamanho* do cartão (Figura 2b) indica o número de módulos do projeto. Entretanto, para tornar o jogo mais realista foi introduzida a noção de que diferentes artefatos devem compor os módulos. Por exemplo, o sistema da Figura 2b deve ser composto de três módulos sendo que o primeiro deve conter: duas cartas de requisitos (RQ), uma de desenho (DS) e uma de código (CD). A composição dos outros dois módulos também é ilustrada na Figura 2b.

Além destas novas informações, os cartões de projeto em SimulES também possuem os atributos herdados de PnP, tais como: (i) *Complexidade* que indica quantos pontos de tempo um engenheiro de software precisa gastar para completar um bom artefato; (ii) *Tamanho* que indica quantos módulos integrados devem ser completados para terminar o projeto; (iii) *Qualidade* que representa o quão livre de defeitos deve estar o produto final; e (iv) *Orçamento* que mostra quantidade de dinheiro disponível para gastar com o projeto. O *Orçamento* é uma restrição para contratação de engenheiros de software bem como para o uso de cartas de conceitos (Seção 3.3).

3.2. O Tabuleiro

Com o intuito de solucionar o Problema E apontado como uma limitação de PnP, esta subseção apresenta um tabuleiro para o jogo SimulES. O tabuleiro é uma área na qual cada jogador coloca seus engenheiros de software, representados por cartas que descrevem suas características, em colunas e os artefatos em linhas. Os artefatos podem ser dos seguintes tipos: requisitos, desenhos, códigos, rastros e ajuda aos usuários. Estes dois últimos não existiam em PnP e foram adicionados para trazer conceitos mais atuais de ES ao jogo. Por exemplo, as cartas de rastros são intencionadas para interligar artefatos e contribuir para a gerência por requisitos, que é apoiada na rastreabilidade. O tabuleiro de jogo em um cenário com a contratação de dois engenheiros é representado na Figura 3. As cartas de artefatos são colocadas nas células do tabuleiro, abaixo do engenheiro que as produziu e nas linhas referentes aos seus tipos. Por exemplo, na linha de requisitos da Figura 3 estão dois artefatos feitos por Janaína e um por Carlos.

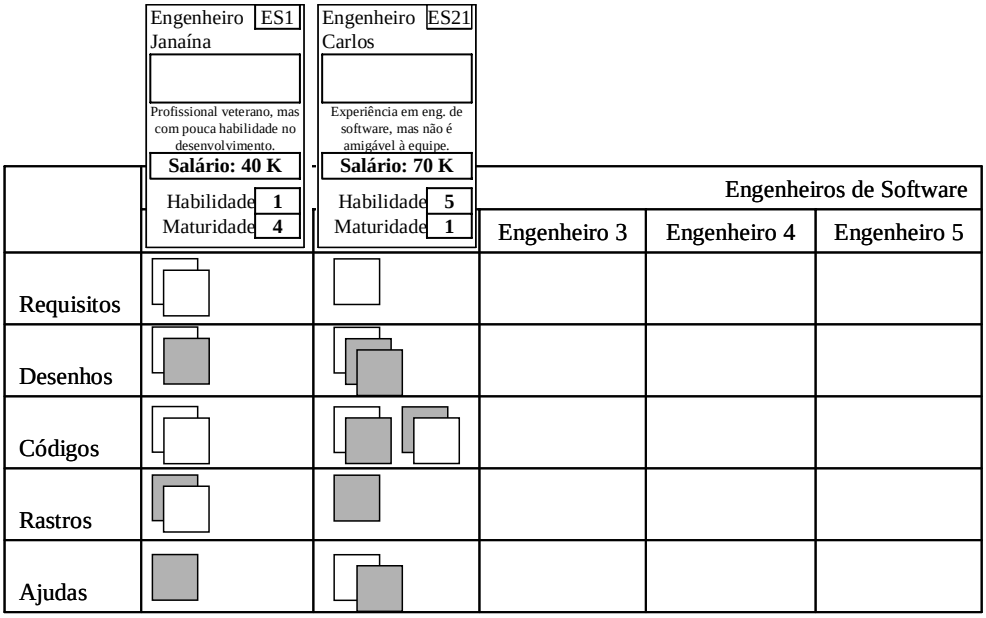


Figura 3. Tabuleiro de jogo com uma configuração de dois engenheiros

3.3. As Cartas

Os artefatos existentes no jogo PnP são as cartas. Apesar de SimulES introduzir novos artefatos como o tabuleiro e o dado, as cartas ainda desempenham o papel principal. Similarmente a PnP, as cartas em SimulES se dividem em quatro categorias: problemas, conceitos, engenheiros de software (programadores em PnP) e artefatos. Todas elas, exceto as cartas de artefatos, possuem nome e código de identificação. O código de identificação foi adicionado em SimulES para possibilitar controle de versão e gerência de configuração. Atualmente, o jogo se encontra na versão 1.0 diponível na página do projeto [SimulES, 2007].

Além do controle de versão e da gerência de configuração, também foi criado o conceito de grupo de cartas. Grupos de cartas são cartas que tratam de um mesmo tema (Tabela 1). Por exemplo, cartas referentes a requisitos recebem o código “RQ”, desenho “DS”, recursos humanos “RH” e outras seguem esse mesmo padrão. A Tabela 1 mostra as seis grandes categorias, suas subdivisões e o número total de cartas em cada categoria. Todos os grupos têm pelo menos 10 cartas para que seu conceito possa ser

bem explorado no jogo. Em adição às existentes em PnP, novas cartas foram criadas para atingir o número mínimo. Esta característica de SimulES permite ao professor fazer o mapeamento direto dos artefatos do jogo aos conceitos de ES envolvidos. Portanto, solucionando o Problema F.

Tabela 1. Agrupamento das cartas por categorias

Categorias	Subcategorias	Nº de Cartas
Gerenciamento	Gerência Técnica, Gerência de Versão e Rastros, Treinamento	11
Recursos Humanos	Personalidade, Recursos Financeiros, Educação	11
Requisitos	Contexto, Problemas ou Alteração, Requisitos Não-Funcionais	12
Desenho	Padrões, *Frameworks*, Qualidade de Desenho	12
Código	Testes e Assertivas, Refatoração	12
Comunicação	Interface com Usuário, Documentação de Ajuda	10

Os quatro tipos de cartas do jogo SimulES (Figura 4) são detalhados a seguir.

a) *Problemas.* Descrevem problemas clássicos de engenharia de software resultantes de falhas no processo de produção. Essas cartas são utilizadas, para criar obstáculos ao progresso dos jogadores adversários. As cartas de problemas possuem os seguintes atributos (Figura 4, a): (i) referências para literatura de apoio; (ii) critério que descreve as condições a serem satisfeitas para que a carta seja lançada; e (iii) efeito no jogador que a receber.

b) *Conceitos.* Descrevem boas práticas de engenharia de software. Essas cartas podem ser utilizadas pelos jogadores para avançarem face ao seu objetivo. Os principais atributos das cartas de conceitos são (Figura 4, b): (i) literatura de apoio; (ii) efeito na configuração do jogo ou tabuleiro do jogador; e (iii) custo (quando presente na carta) que incorre em gastos após o conceito ser aplicado.

c) *Engenheiros de Software.* Principal recurso que o jogador terá para progredir no jogo, pois, são os engenheiros que produzem os artefatos necessários para cumprir o projeto. Estas cartas apresentam as seguintes informações (Figura 4, c): (i) nome e descrição pessoal do engenheiro; (ii) salário que deve respeitar o orçamento do projeto; (iii) habilidade ou número de "pontos de tempo" que um engenheiro é capaz de desempenhar em ações do jogo; e (iv) maturidade que reflete sua tendência em ser um bom trabalhador.

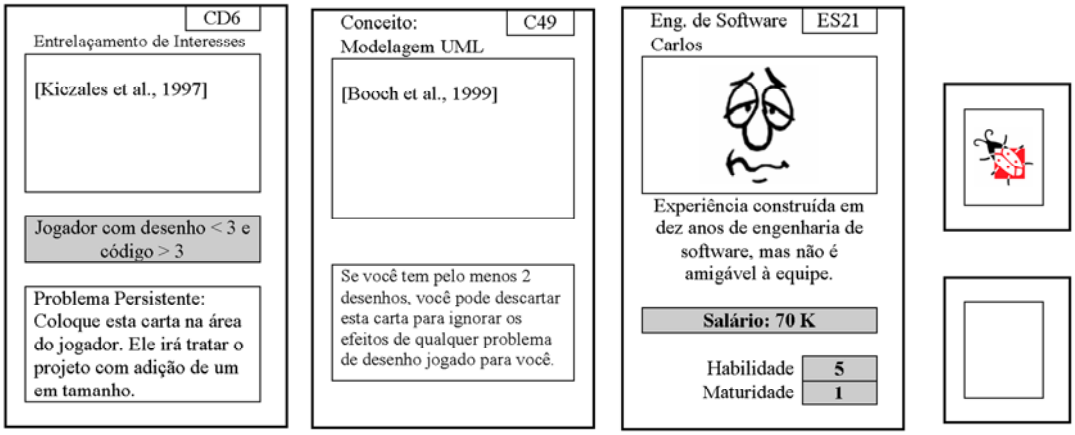


Figura 4. Cartas do jogo: (a) problema, (b) conceito, (c) engenheiro de software e (d) artefatos com e sem defeito

d) *Artefatos*: simbolizam os produtos produzidos pelos engenheiros e podem conter ou não defeitos (Figura 4, d). Além disso, os artefatos podem ter duas cores, branco ou cinza, dependendo de sua qualidade. Apesar de gastarem o dobro dos "pontos de tempo" para serem produzidas, as cartas brancas contêm defeitos na proporção de 5 cartas para 1 defeito enquanto nas cartas cinzas esta proporção é de 3 para 2. Ou seja, com a prática o jogador aprende que geralmente é melhor desenvolver um bom artefato (branco), mesmo que este seja mais trabalhoso.

3.4. A Dinâmica do Jogo

A dinâmica do jogo SimulES apresentada nesta seção é similar à de PnP e, portanto, discutiremos apenas os detalhes mais relevantes. Antes do início do jogo, um cartão de projeto é escolhido aleatoriamente de uma série de projetos disponíveis. As informações deste cartão devem ficar visíveis a todos os jogadores. Em seguida, cada jogador monta seu tabuleiro de jogo e as cartas são separadas em quatro montes: (i) engenheiros de software, (ii) problemas e conceitos, (iii) artefatos brancos e (iv) artefatos cinzas. Diferentemente de PnP, SimulES limita o número de participantes entre 4 e 8. Com o dado escolhe-se quem começa o jogo prosseguindo em sentido horário.

A cada jogada, o jogador da vez lança o dado e retira cartas nos montes dependendo do valor obtido. Por exemplo, se na vez da jogadora Maria o número tirado no dado for 5, Maria pode retirar 3 cartas no monte de problemas e conceitos e 2 (5-3) no de engenheiros de software. As cartas de problemas e conceitos são guardadas na mão do jogador até o momento que ele achar oportuno para jogá-las. As cartas de engenheiros também podem ser guardadas ou então colocadas imediatamente no tabuleiro. O segundo caso indica a contratação do funcionário representado pela carta.

Durante a jogada do jogador, um engenheiro de software pode exercer uma série de tarefas. A habilidade dos engenheiros determina quantos "pontos de tempo" eles têm e, portanto, quantas ações eles podem desempenhar. Por exemplo, as cartas de artefatos são retiradas dos montes, i.e. artefatos produzidos, de acordo com os pontos de habilidade dos engenheiros de software. Além de construir artefatos, os engenheiros têm três outras opções de tarefas: inspecionar artefatos, corrigir defeitos e integrar artefatos em um módulo. Ao fim de sua jogada, o jogador está apto a receber as cartas de problemas de seus adversários. Cada jogador recebe até três cartas de problemas que podem ser lançadas dos três jogadores imediatamente anteriores a ele. O limite de jogadores e de problemas lançados soluciona o Problema D (Seção 2.2).

Em SimulES é dada a liberdade ao jogador para a escolha da abordagem de desenvolvimento de seu interesse, com seus prós e contras (Problema A). Assim, o jogo não é mais limitado ao Modelo Cascata, uma vez que não obriga a utilização do processo totalmente seqüencial e o jogador pode voltar para corrigir problemas nas fases anteriores (o que não é permitido em PnP). Por exemplo, mesmo que esteja na fase de codificação o jogador pode voltar a trabalhar nos requisitos ou desenho do projeto.

Uma vez alcançado o número de módulos necessários para o término do projeto, o jogador pode afirmar que completou o sistema. Neste momento é feita a validação por parte do cliente (papel desempenhado pelo professor da disciplina). Isto significa que alguns dos módulos do projeto são aleatoriamente conferidos e devem estar livres de problemas. O número de módulos a serem conferidos na validação é igual ao valor do atributo *Qualidade* indicado na carta de projeto (Figura 2b). O jogador somente é

declarado vencedor se não for encontrado defeito em nenhum dos artefatos que compõem os módulos conferidos.

4. Trabalhos Relacionados e Limitações do Estudo

A utilização de jogos como forma de melhorar o aprendizado é um tema bem explorado na literatura [Oh e Hoek 2001] [Virvou et al 2005] e mesmo em disciplinas de computação já existem iniciativas para utilização de jogos ou simuladores [deLaet et al. 2005]. Por exemplo, no contexto de engenharia de software, Drappa e Ludewig [2000] apresentam o projeto SESAM constituído principalmente de um simulador para treinamento em projetos de software. Este simulador permite que um usuário assuma o papel de gerente e utilize modelos e regras complexas no desenvolvimento de um sistema. Em comparação a SimulES e PnP, o projeto SESAM não possui características de jogos, tais como entretenimento e competitividade. Além disso, tal simulador é destinado principalmente a profissionais e não a estudantes de computação.

A equipe idealizadora de PnP, também disponibiliza uma versão digital do jogo chamada SimSE [Navaro et al. 2004]. Em sua versão para computador, o jogo permite que um jogador assuma o papel de gerente de projeto e desempenhe atividades como contratar e demitir programadores, associar tarefas, monitorar o progresso, entre outras. Por ser em sua essência adaptação de PnP, o jogo SimSE recai nos mesmos problemas discutidos na Seção 2.2 como limitação ao Modelo Cascata e falta de ligação explicita dos artefatos do jogo com os conceitos de engenharia de software. Em adição, o jogo perde muito de seu atrativo por não permitir competição entre múltiplos jogadores.

Como uma limitação deste trabalho, acreditamos que o jogo SimulES ainda não foi suficientemente exercitado em sala de aula. Além disso, ele foi baseado apenas em melhorias propostas para PnP. Entretanto, o fato de PnP ter sido bastante praticado em sala de aula e ser reconhecidamente útil para o ensino de ES minimiza a falta de experimentação de SimulES. A restrição a PnP é justificável por tal jogo apresentar características comuns a outros jogos educativos, como por exemplo, a simulação do processo de desenvolvimento de software [deLaet et al. 2005]. Finalmente, acreditamos que um estudo detalhado dos aspectos de qualidade geral do jogo SimulES, incluindo questionamento aos alunos e medição, mereça seu próprio artigo.

5. Conclusões e Trabalhos Futuros

Neste artigo foi apresentado um jogo educacional que simula o processo de desenvolvimento de software desde a fase de concepção até a fase de entrega do produto. As contribuições deste artigo podem ser vistas sobre dois aspectos: (i) uma extensa avaliação do bem conhecido jogo PnP para ensino de ES e (ii) um jogo que considera práticas modernas e evolução de software. Na avaliação, apresentamos os principais problemas de PnP. A partir de soluções para tais problemas, o artigo propõe o jogo SimulES que tem como objetivo ensinar práticas mais modernas de ES, tal como evolução de software, bem como ser mais dinâmico e atrativo que seu predecessor.

Na verdade, este trabalho integra esforços anteriores na melhoria do ensino de ES. Por exemplo, com o uso do conceito de evolução na disciplina de Princípios de Engenharia de Software do curso de graduação em Engenharia da Computação da PUC-Rio. Essa disciplina foi modificada em 2001/2002 para focar mais em práticas não tradicionais [Silva et al. 2004], como Programação Extrema (XP). Como trabalho futuro,

deve ser feita uma avaliação quantitativa e/ou qualitativa do jogo SimulES utilizando questionários respondidos por alunos com diferentes graus de conhecimento em ES. É proposta ainda a criação de uma versão digital do jogo utilizando recursos tecnológicos que permitam alta interatividade entre jogadores.

Agradecimentos

Agradecemos aos alunos de Evolução de Software Glória Oliveira e Ritomar Torquato pelas frutíferas discussões sobre o jogo SimulES. Eduardo é financiado pela CAPES (Processo 4216/05-9, Doutorado Pleno no Exterior).

Referências

Baker, A., Navarro, E. and Hoek A. (2005) “An Experimental Card Game for Teaching Software Engineering Processes”. In: Journal of Systems and Software, v. 75, 1-2, pp. 3-16.

Beck, K. (1999) "Extreme Programming Explained". Addison-Wesley Longman.

Boehm, B. (1986) "A Spiral Model of Software Development and Enhancement". In: ACM SIGSOFT Software Engineering Notes, vol. 11, issue 4, pp. 14-24.

Chung, L., Nixon, B., Yu, E. and Mylopoulos, J. (2000) “Non-Functional Requirements in Software Engineering”. Kluwer Publishing.

deLaet, M., Kuffner, J., Slattery, M. and Sweedyk, E. (2005) "Computer Games and CS Education: Why and How". In: Symposium on Computer Science Education, USA.

Drappa, A. and Ludewig, J. (2000) "Simulation in Software Engineering Training". In: International Conference on Software Engineering (ICSE), pp. 199-208, Limerick, Ireland.

Figueiredo, E., Lobato, C., Dias, K., Leite, J. e Lucena, C. (2006) "SimulES: Um Jogo para o Ensino de Engenharia de Software". Relat. Técnico 34/06, Depto de Informática, PUC-Rio.

Garcia, A., Sant'Anna, C., Chavez, C., Silva, V., Staa, A. and Lucena, C. (2004) “Separation of Concerns in Multi-Agent Systems: An Empirical Study”. In: SEMAS, Springer, LNCS 2940.

Lehman, M. (1996) "Laws of Software Evolution Revisited". In: 5th European Workshop on Software Process Technology, LNCS, vol. 1149, pp. 108-124.

Mylopoulos, J., Chung, L. and Yu, E. (1999) "From Object-oriented to Goal Oriented Requirements Analysis". In: Communications of the ACM, vol. 42, no. 1, pp. 31-37.

Navarro, E., Baker, A. and Hoek, A. (2004) “Teaching Software Engineering Using Simulation Games”. In: International Conference on Simulation in Education (ICSIE), California, USA.

Oh, E. and Hoek, A. (2001) "Adapting Game Technology to Support Individual and Organizational Learning". In: Int’l Conf. on SE & Knowledge Eng. (SEKE), p. 347-354, AR.

Oliveira, A., Cysneiros, L., Leite J., Figueiredo, E. and Lucena, C. (2006) “Integrating Scenarios, i*, and AspectT in the Context of Multi-Agent Systems”. In: CASCON, Canada.

“Problems and Programmers Home Page” (2007). Disponível on-line em http://www.problemsandprogrammers.com. Acessado em Fevereiro, 2007.

Royce, W. (1970) "Managing the Development of Large Software Systems", In: IEEE WESCON, IEEE Press, pp. 1-9, San Francisco.

Silva, L., Leite, J. e Breitman, K. (2004) "Ensino de Engenharia de Software: Relato de Experiências". In: Workshop de Educação em Informática (WEI), pp. 994-1005, Salvador.

“SimulES: Simulador de Engenharia de Software” (2007). Disponível on-line em http://www.teccomm.les.inf.puc-rio.br/emagno/simules/. Acessado em Fevereiro, 2007.

Virvou, M., Katsionis, G., Manos, K. (2005). “Combining Software Games with Education: Evaluation of its Educational Effectiveness”. Educational Technology & Society, pp. 54-65.