“Vida de ES”: Protótipo de um Jogo de Cartas para o Ensino de Situações Práticas Vivenciadas por um Engenheiro de Software

Delivelton Teixeira Rodrigues
Lab. de Entretenimento Digital Aplicado - LEnDA
Universidade Estadual de Feira de Santana - UEFS
Feira de Santana, BA, Brasil
delivelton.rodrigues@ifba.edu.br

Victor Travassos Sarinho
Lab. de Entretenimento Digital Aplicado - LEnDA
Universidade Estadual de Feira de Santana - UEFS
Feira de Santana, BA, Brasil
vsarinho@uefs.br

***Resumo*—O ambiente de trabalho de um engenheiro de software contém peculiaridades difíceis de serem praticadas em sala de aula, em especial quando ensinado com uma metodologia tradicional voltada para a explanação teórica do assunto. “Vida de ES” é um jogo de cartas capaz de simular varias situações vividas por um engenheiro de software no seu âmbito de trabalho. Como resultado, através de uma disputa por duas equipes pelo desenvolvimento com melhor desempenho de dois softwares distintos, “Vida de ES” proporciona jogadas capazes de estimular melhores tomadas de decisões necessárias aos atores em um projeto de sistema de software proposto.**

***Index Terms*—engenharia de software, serious game, jogo educativo, simulação, jogo de cartas**

I. Introdução

Apesar das diferentes linhas de pesquisa e de trabalho atualmente aplicadas na informática em si, tem-se na Engenharia de Software (ES) uma posição de destaque como disciplina aplicada em cursos de computação diversos no Brasil e no mundo. ES representa uma linha de estudo da computação que se preocupa em ensinar a lidar com as diferentes etapas requeridas para o desenvolvimento de um software qualquer, variando desde a entender a necessidade de um cliente até a classificação do processo de desenvolvimento e a entrega final do sistema como um todo.

Contudo, durante o ensino de ES, é comum não apresentar ao aluno o ambiente real de atuação da ES que ele irá enfrentar no futuro, uma vez que o mesmo contém peculiaridades difíceis de serem praticadas em sala de aula, em especial quando ensinado com uma metodologia tradicional voltada para a explanação teórica do referido assunto. Como resultado, tem-se uma contradição em não ensinar aspectos práticos em ES, uma vez que “parte dos métodos da engenharia provém da experiência prática, e não apenas da pesquisa científica” [1].

Para situações como essa em que o ensino tradicional tende a não ser satisfatório, e levando em consideração que a “maioria dos cursos de graduação em ciências da computação ainda são frequentemente ministrados por meio de um paradigma antigo que não é adequado para lidar com as preocupações modernas” [2], se faz necessário o uso de metodologias e ferramentas de ensino alternativas, capazes de trazer uma abordagem diferenciada e mais moderna para o aprendizado do discente de ES.

Assim, neste contexto de dificuldades no ensino de situações práticas vivenciadas por engenheiros de software, e considerando o uso de jogos como sendo uma alternativa inovadora que vem sendo aplicada para a capacitação de alunos nos conceitos de ES [3], este artigo apresenta o “**Vida de ES**”, um jogo de cartas que simula através de uma competição entre equipes um ambiente prático de construção de software.

II. Jogo Proposto

Vida de ES é um jogo de cartas idealizado para simular uma disputa de desenvolvimento de software onde duas equipes competem para alcançar o melhor desempenho entre custo, tempo e eficiência na construção dos softwares demandados. O jogo é composto por 3 baralhos com um total de 5 tipos de cartas diferentes (*Software*, *Requisitos*, *Modelagem*, *Desenvolvimento* e *Situações*), um tabuleiro e 2 scores que controlam o desenrolar da partida.

A. Tipos de Cartas do Jogo

Com relação as cartas de *Software* (Fig. 1), estas são utilizadas apenas no início do jogo onde é sorteada uma carta para cada equipe. Cada carta de Software apresenta: em sua parte inferior, a descrição do software a ser desenvolvido; em sua parte superior, um número que representa a quantidade de requisitos necessários para o seu desenvolvimento; e na parte inferior direita, o prazo de entrega esperado contado em semanas.

As cartas de *Requisitos* (Fig. 1) são utilizadas na segunda etapa do jogo, fazendo referência à captura de requisitos da ES. Elas são compostas por um nome, descrição de sua finalidade e um score específico de implementação, e, assim como as cartas de Software, elas também ficam em um único baralho isolado no tabuleiro.

Com relação as cartas de *Modelagem* (Fig. 2), estas descrevem os processos de diagramação, frameworks e padrões de projeto a serem usados no desenvolvimento de sistemas, indicando as vantagen e/ou desvantagens bem como o tempo

Fig. 1. Exemplo de carta de *Software* e carta de *Requisito*.

em semanas de sua implementação. Suas caraterísticas físicas se distinguem pela cor de fundo rosa e um número no canto superior que indica o tempo necessário para sua implementação.

Já as cartas de *Desenvolvimento*, estas possuem cor de fundo azul e indicam o tipo de desenvolvedor e a quantidade de score a ser recebido, através de uma semana de trabalho (uma rodada do jogo). Os versos das cartas de *Modelagem*, *Desenvolvimento* e *Situações* são idênticos e elas fazem parte de um mesmo baralho no jogo, com exceção daquelas classificadas como *Bug* que possui uma coloração preta em seu verso.

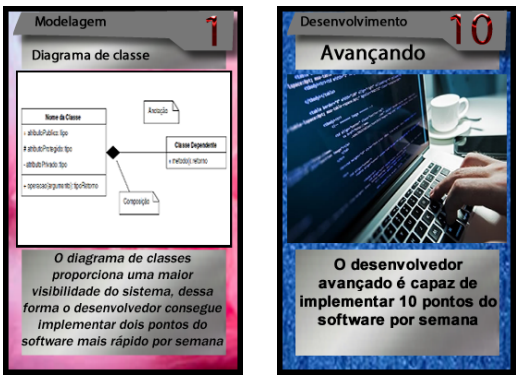

Fig. 2. Exemplo da carta de *Modelagem* e de *Desenvolvimento*.

Para simular a pressão de um ambiente de trabalho e estimular a tomada de decisão por parte dos jogadores, também foram definidas cartas de *Situação* que representam alguns eventos vivenciados na produção de softwares diversos, tais como: a necessidade de troca de um requisito, um possível bug em alguma parte do sistema, a adição de um novo integrante na equipe de desenvolvimento, entre outras situações possíveis.

B. O Tabuleiro

Vida de ES possui um tabuleiro que busca representar um *dashboard* com a finalidade de dar um feedback sobre o que está sendo desenvolvido pela equipe, bem como sua relação com o prazo de entrega indicado na carta de *Software*. Assim, ele é composto por dois lados iguais, sendo um lado para cada equipe. Cada lado é marcado com 6 campos para diferentes tipos de cartas, um campo designado para a carta *Software*, um para as cartas *Requisitos* a serem desenvolvidos, outro para os *Requisitos* já implementados, um para as cartas *Modelagem*, um para as cartas *Desenvolvimento* e outro para os *bugs* encontrados (Fig. 3). Os bugs são as únicas cartas *Situação* que ficam em campo até serem resolvidas, as demais cartas *Situação* são jogadas no tabuleiro e descartadas na mesma rodada. Bugs também se diferenciam no seu saque, uma vez que, por possuir cor de fundo preta, são identificadas antes da vez do jogador. Assim que um carta de bug é puxada, ela deve ir imediatamente para o tabuleiro da equipe que a puxou, podendo ser jogada com a face para cima ou para baixo a depender do processo de software utilizado pela equipe.

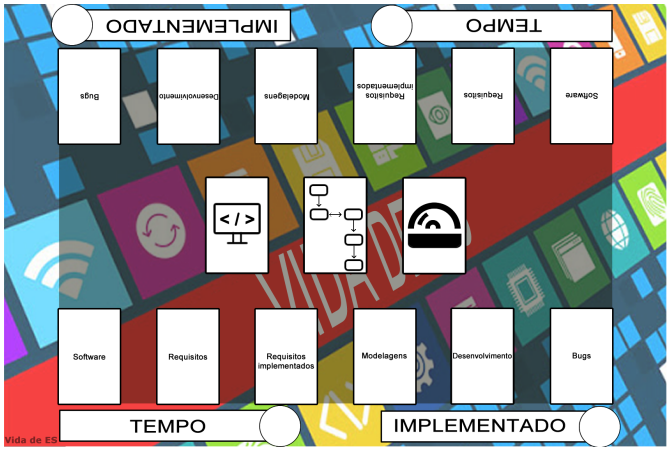

Fig. 3. Tabuleiro do jogo Vida de ES.

C. Scores no Jogo

Para cada lado do tabuleiro, existem 2 campos destinados aos scores de tempo e de implementação de cada carta de Software. Eles servem para representar o que já foi implementado da carta Software por cada equipe, bem como o tempo de implementação gasto nessa tarefa.

O score de implementação é iniciado com o valor zero e, para que a equipe tenha o desempenho ideal, ele deve ter um valor igual a soma dos scores de todos os requisitos que a equipe sorteou pra si no início da partida. Em consequência disso, cada equipe tem um score de implementação final diferente.

Já o score de tempo, este é obtido a partir da carta Software, que vem especificando o tempo em semanas no qual o projeto deve ser entregue. Seu valor inicial é zero e o valor final ideal é o mesmo especificado na carta *Software* que foi sorteada no início da partida.

Enquanto as alterações no score de tempo são constantes durante as rodadas, as alterações no score de implementação acontecem de acordo com o uso das cartas no campo da equipe. De fato, cartas de *Desenvolvimento* indicam desenvolvedores que geram pontos de implementação por rodada, cartas de *Modelagem* podem acelerar o processo de

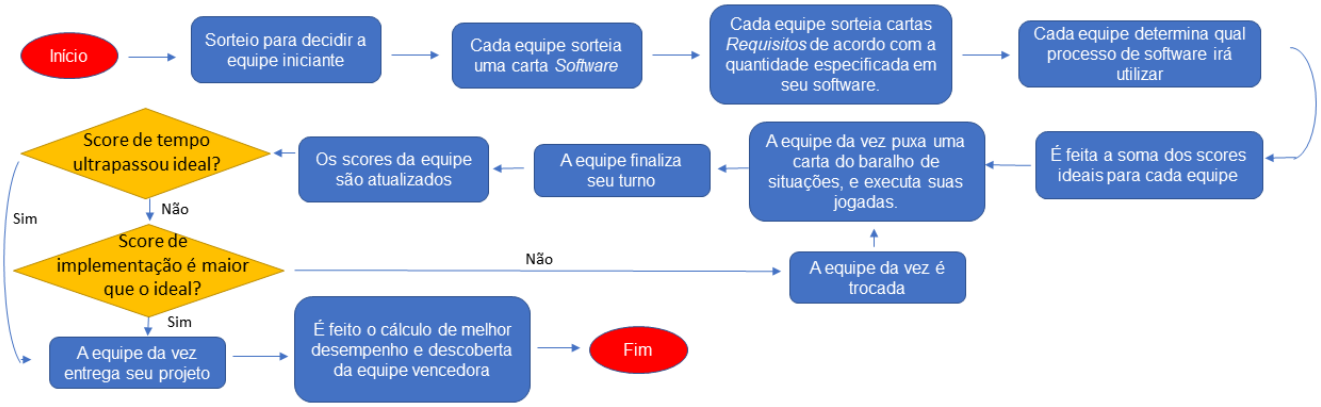

Fig. 4. Fluxograma de execução de uma partida com o jogo Vida de ES.

implementação obtidos pelos desenvolvedores, enquanto que as cartas *Situação* podem desacelerar o desempenho dos desenvolvedores.

O desempenho de cada equipe é calculado com base no tempo gasto pela equipe e na quantidade de software implementado, o qual é representado por:

$$D = 10 - 5 * (TG/TP + (TS - SI)/TS)$$

Onde: D = *Desempenho*, TG = *Score de Tempo Gasto*, TP = *Score de tempo previsto na carta Software*, SI = *Score de Implementação alcançado pela equipe*, TS = *Tamanho do Software (soma dos scores de implementação dos Requisitos)*, com TG menor ou igual que TP e SI menor ou igual que TS.

Como resultado, o desempenho final é obtido pela retirada de pontos por conta da diferença do tempo gasto com relação ao tempo esperado de desenvolvimento, e por conta da diferença do tamanho do software produzido com relação ao tamanho planejado. Ao final da partida, vence a equipe que tiver alcançado o melhor desempenho na construção do seu produto. Ou seja, a equipe que obtiver a pontuação mais próxima de 10 é a vencedora, mesmo que não tenha efetuada a "entrega" completa de seu projeto de software.

D. Fluxo de uma Partida

O Vida de ES foi projetado para ser jogado por pelo menos duas equipes de 1-3 integrantes. O fluxo de uma partida tem como etapa inicial o sorteio de uma carta *Software* e o sorteio de cartas *Requisitos* na quantidade descrita pela carta Software. Em seguida, cada equipe deve escolher o modelo de processo que julgar mais adequado para desenvolvimento do seu software sorteado (Fig. 4).

Durante a rodada, cada integrante da equipe deve sacar na sua vez uma carta do baralho de situações (exceto na primeira rodada do jogo onde todos sacam 3 cartas), e pode jogar uma única carta no tabuleiro passando posteriormente a vez para o próximo integrante da equipe. Isso se repete até o último integrante da equipe, o qual após jogar sua carta no tabuleiro deve passar a vez para a equipe adversária. O saque de carta é obrigatório para cada jogador, porém jogar uma carta no tabuleiro não é obrigatório, exceto quando o jogador saca uma carta bug, que deve ser colocada imediatamente no tabuleiro e a vez deve ser passada para o próximo integrante da equipe ou para a equipe adversária.

A dinâmica do Vida de ES define três modelos de processo de software possíveis de serem escolhidos: *Cascata*, *Scrum* e *Espiral*. Cada um desses modelos contêm regras distintas e cada equipe deve declarar antes de seu primeiro saque qual modelo será utilizado por toda a partida.

Levando em consideração as características do modelo Cascata, onde os principais subprocessos devem ser executados em estrita sequência através de pontos de controle bem definidos [1], caso a equipe escolha o modelo cascata, ela deve cumprir as seguintes regras:

- Os campos do tabuleiro devem ser utilizados de maneira sequencial, onde só é possível jogar cartas de *Desenvolvimento* após o campo de cartas de *Modelagem* estar ocupado. Não se pode jogar cartas de *Modelagem* após utilizar cartas de *Desenvolvimento*;
- As cartas no campo *Bug* devem ser jogadas viradas para baixo e a equipe que estiver jogando utilizando o modelo cascata só pode virar as mesmas de face para cima depois que o campo de carta de desenvolvimento estiver ocupado, simulando assim a descoberta de um bug após o desenvolvimento.

Por outro lado, caso a equipe opte pelo modelo Scrum, no qual são realizadas reuniões de acompanhamento diárias de curta duração nas quais se discute o que foi feito desde a última reunião e o que precisa ser feito até a próxima [4], e onde "as dificuldades encontradas e os fatores de impedimento (*bottlenecks*) são identificados e resolvidos" [4], tem-se as seguintes regras definidas no jogo:

- As cartas podem ser jogadas em qualquer sequência no tabuleiro;
- Os jogadores da equipes no começo de cada rodada podem trocar cartas com os colegas de equipe, simulando as reuniões entre membros da equipe;
- As cartas jogadas no campo *Bug* não precisam ficar com a face para baixo, simulando assim as dificuldades

encontradas durante o processo e a resolução das mesmas a qualquer momento do processo.

Por fim, caso a equipe escolha utilizar o modelo Espiral, onde "o produto é desenvolvido em uma série de iterações e cada nova iteração corresponde a uma volta na espiral" [1], a qual "permite construir produtos em prazos curtos, com novas características e recursos que são agregados na medida em que a experiência descobre sua necessidade" [1], tem-se as seguintes regras a serem seguidas:

- Assim como no modelo Cascata as cartas devem ser jogadas em sequência no tabuleiro, porém isso deve ocorrer conforme a implementação de cada requisito, sendo um por vez;
- As cartas jogadas no campo Bug não precisam ficar com a face voltada para baixo.

A escolha do modelo de processo a ser usado deve ser feita antes da primeira rodada e não pode ser alterada durante a partida. Cada grupo pode escolher um processo distinto, levando em consideração o tamanho do seu software e o respectivo prazo de entrega, conforme indicados nas cartas de Software e de Requisitos.

Para softwares maiores e com poucos requisitos o modelo Scrum se destaca pela possibiliade de trocas de cartas entre os integrantes e a não obrigatoriedade de seguir uma determinada sequência. Porém, quando se tem uma quantidade grande requisitos a gerência através do Scrum tende a ficar mais complicada no jogo. Em contrapartida, o modelo Espiral se destaca com um melhor desempenho em softwares com grande quantidade de requisitos, tendo uma sequência de fases para cada requisito mais simples de se gerenciar e mais rápida de se implementar. Quando o software exige o desenvolvimento de poucos e pequenos requisitos o modelo cascata se destaca perante os outros modelos por conta da sua objetividade.

III. Resultados Obtidos

Com relação a assimilação de conceitos de ES apresentados no jogo proposto, atividades de Concepção e Análise de Requisitos em um projeto de software são realizadas logo na primeira fase do jogo, onde cada equipe sorteia um software a ser construído, define os requisitos a serem implementados, bem como o tempo necessário para a sua implementação. Com relação a fase de Construção, esta é representada com uso do baralho que contêm as cartas de Desenvolvimento, Modelagem e Situações, onde os jogadores encontram propriedades de definição de modelos, de desenvolvimento de código e de possíveis situações correlacionadas à profissão de um engenheiro de software. Em se tratando da fase de Testes, Vida de ES possui o campo de testes, onde os erros (cartas de Bug) são identificados e corrigidos. E com relação a fase de finalização e entrega do projeto, esta é representada pelo fim da partida, quando todos os requisitos são implementados e sem bugs por uma das equipes do jogo. Nesse momento o cálculo é feito com base no tempo gasto e no tempo predefinido no inicio da partida, e a equipe que tiver o melhor desempenho vence a partida. A partida também pode ser finalizada quando uma das equipes gasta todo o seu tempo disponível. Nesse momento a equipe é declarada como derrotada independente da pontuação da outra equipe, simulando assim uma perda de contrato devido ao não cumprimento do prazo de projeto.

Vida de ES também foi inicialmente aplicado em um curso de modalidade integrado de Tecnologia da Informação na disciplina de Projeto de Software de um colégio de ensino técnico para 24 alunos numa faixa etária média de 16 anos. A aplicação ocorreu com o intuito de avaliar o conhecimento dos alunos, a qual foi realizada em dois dias em um formato de torneio, onde equipes anteriormente formadas em outras atividades de ensino continuaram com a mesma formação durante o torneio.

Como foi a primeira experimentação do jogo algumas regras tiveram que ser alteradas no decorrer do torneio, tornando-o mais consistente de modo a manter o jogo mais próximo da realidade representada pelos seus conceitos. Ao fim do torneio, cada integrante respondeu um questionário com cinco questões a respeito do jogo, bem como campos para a coleta de opiniões sobre possíveis melhorias do mesmo.

IV. Considerações finais

Este artigo apresentou Vida de ES, um jogo de cartas capaz de simular várias situações vividas por um engenheiro de software no seu âmbito de trabalho. Para tal, foram descritos os elementos e regras gerais do jogo, bem como os aspectos de ES diretamente relacionados com as dinâmicas de jogo descritas.

Com relação a experimentação do jogo em si, Vida de ES apresentou resultados interessantes de usabilidade, jogabilidade e imersão nos testes preliminares realizados com estudantes. A assimilação dos conceitos de ES abordados e as regras do jogo também atingiram um sincronismo capaz de projetar uma simulação promissora de um ambiente de trabalho de um engenheiro de software. Porém, mesmo com estes resultados positivos, algumas melhorias ainda se fazem necessárias para tornar o jogo ainda mais satisfatório, a exemplo da construção de uma "calculadora digital" capaz de calcular os scores das equipes de forma automática, agilizando assim o desenrolar de cada partida a ser realizada.

Como trabalhos futuros, pretende-se implementar uma versão digital para o jogo proposto, a qual permita também a personalização de cartas por parte de seus jogadores. A avaliação do jogo proposto em outras turmas relacionadas ao ensino de conceitos de ES também será realizadas em um futuro próximo.

Referências

[1] W. de Pádua Paula Filho, *Engenharia de software*. LTC, 2003, vol. 2.

[2] P. Parsons, "Preparing computer science graduates for the 21st century," *Teaching Innovation Projects*, vol. 1, no. 1, pp. 1–4, Feb. 2011. [Online]. Available: https://ojs.lib.uwo.ca/index.php/tips/article/view/3552

[3] L. Fernandes, C. Maria, and C. Werner, "Sobre o uso de jogos digitais para o ensino de engenharia de software," p. 4, 09 2020.

[4] M. dos Santos Soares, "Metodologias ágeis extreme programming e scrum para o desenvolvimento de software," *Revista Eletrônica de Sistemas de Informação*, vol. 3, no. 1, 2004.