Tester's Quest: Um Jogo de Tabuleiro Cooperativo para o Ensino de Teste de Software

Uma Abordagem Lúdica para Explorar a Pirâmide de Testes

Elias Torres
Centro de Informática
Universidade Federal da Paraíba
João Pessoa, PB, Brasil
eliasvictor16.contato@gmail.com

João Lima
Centro de Informática
Universidade Federal da Paraíba
João Pessoa, PB, Brasil
joaovictoroliveira13@hotmail.com

Maria Alves
Centro de Informática
Universidade Federal da Paraíba
João Pessoa, PB, Brasil
mariasagurgel@gmail.com

Arthur Ribeiro
Centro de Informática
Universidade Federal da Paraíba
João Pessoa, PB, Brasil
arthur.oliveira.rj.ni@gmail.com

Carlos Costa
Centro de Informática
Universidade Federal da Paraíba
João Pessoa, PB, Brasil
carloscosta8001@gmail.com

Yuska Aguiar
Centro de Informática
Universidade Federal da Paraíba
João Pessoa, PB, Brasil
yuska@ci.ufpb.br

RESUMO

O Tester's Quest é um jogo de tabuleiro, inspirado no jogo Pandemic, desenvolvido e aplicado na disciplina de Teste de Software enquanto ferramenta educacional para explorar conceitos relativos aos níveis da pirâmide de teste e suas características. Sua mecânica promove a cooperação entre os jogadores quando estes assumem o papel de especialistas em qualidade de software para identificar e combater bugs a partir de recursos limitados de projetos de software. O jogo é composto por um tabuleiro representando a pirâmide de testes, cartas que representam eventos, bugs e power-ups — que simulam situações reais do desenvolvimento de software, contextualizados ao nível de teste no qual os jogadores se encontram na pirâmide. O Tester's Quest foi aplicado de forma exploratória para refinamento em duas turmas. Foi perceptível que a proposta reforça a correspondência entre o conteúdo didático alvo e a mecânica do jogo, favorecendo o aprendizado.

PALAVRAS-CHAVE

Teste de Software, Jogos Educacionais, Gamificação, Jogos Autorais, Tester's Quest

1 Introdução

A manutenção do engajamento dos alunos durante o processo de aprendizagem é um dos principais desafios enfrentados no contexto educacional. O modelo de ensino tradicional, centrado em aulas expositivas e na passividade do aluno, mostra-se frequentemente insuficiente para manter o engajamento, promover aprendizagem significativa e atender às necessidades diversificadas dos estudantes, conforme criticado por Barr e Tagg [1]. Os autores destacam ainda um paradoxo recorrente: mesmo alunos com excelentes notas acadêmicas, muitas vezes demonstram incapacidade de aplicar o conhecimento adquirido fora dos contextos específicos em que foram ensinados. Essa limitação expõe a fragilidade de um modelo que prioriza a memorização em detrimento da construção de competências aplicáveis.

Essa lacuna no processo de ensino-aprendizagem torna-se especialmente crítica quando se considera a formação de profissionais na área da computação. Nessas carreiras, é essencial que os egressos não apenas compreendam os conceitos teóricos, mas sejam capazes de aplicá-los na prática em contextos reais e colaborativos. Segundo as diretrizes estabelecidas no documento Software Engineering 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering [13] elaborado pela IEEE e ACM, espera-se que os estudantes desenvolvam competências como pensamento crítico, trabalho em equipe, comunicação eficaz e aplicação prática do conhecimento técnico. Essa orientação está em consonância com as Diretrizes Curriculares Nacionais para os cursos de Computação no Brasil, estabelecidas pelo Ministério da Educação [5], que também enfatizam a importância da formação integral e da articulação entre teoria e prática no desenvolvimento de soluções computacionais. A ausência de estratégias pedagógicas que promovam essas habilidades pode comprometer diretamente a preparação dos alunos para os desafios do mercado e para sua atuação como profissionais reflexivos e eficazes.

Partindo da contestação das deficiências do modelo tradicional de ensino, torna-se imperativa a busca por alternativas pedagógicas capazes de superar esses entraves e promover uma aprendizagem plena. Nesse contexto, modelos que rompam com a lógica expositiva e incorporem os princípios dos Quatro Pilares da Educação propostos pela UNESCO [7] — aprender a conhecer, fazer, conviver e ser — apresentam-se como caminhos promissores. Dentre esses modelos, destaca-se a gamificação, definida por Deterding et al. [8] como o uso de elementos de design de jogos em contextos que não são jogos buscando aumentar o engajamento, a motivação e a participação ativa dos estudantes no processo de aprendizagem.

Embora muitos estudos foquem em soluções digitais para o contexto de ensino-aprendizagem com gamificação, os jogos analógicos — como jogos de tabuleiro — também se destacam como alternativas viáveis para aplicação educacional. Essas propostas se alinham ao conceito de Computação Desplugada, que busca ensinar princípios de computação por meio de atividades interativas realizadas sem o uso de computadores promovendo uma aprendizagem mais acessível, contextualizada e participativa, como proposto por Curzon et al. [4]. Além de dispensarem recursos tecnológicos, essas atividades favorecem a interação social direta, a colaboração entre

os participantes e o desenvolvimento de estratégias coletivas — elementos que enriquecem o processo de ensino-aprendizagem.

Esses benefícios tornam a gamificação uma estratégia pedagógica particularmente eficaz para disciplinas técnicas com elevado grau de abstração, como por exemplo Teste de Software. Embora essencial para assegurar a qualidade de sistemas, o ensino dessa disciplina frequentemente se restringe à transmissão passiva de conceitos teóricos e classificações, dificultando a internalização prática pelos discentes, conforme apontado por Nascimento et al. [12]. Conforme demonstrado pelos autores, os alunos comumente percebem as atividades de teste como "tediosas e custosas", evidenciando uma lacuna entre a importância da disciplina e sua percepção discente. Ainda segundo o estudo, 94,3% dos alunos que participaram da pesquisa preferem uma abordagem gamificada, sinalizando maior receptividade a métodos que promovem participação ativa.

Nesse contexto, o desenvolvimento de um jogo de tabuleiro educativo surge com um duplo propósito: além de funcionar como um recurso didático, também é fruto da experiência dos autores no próprio processo de aprendizagem e elaboração do material. Assim, a criação do jogo representa tanto uma ferramenta para facilitar o ensino quanto uma vivência prática que aprofunda a compreensão dos conceitos e desafios inerentes ao teste de software.

A partir dessa iniciativa, o presente artigo apresenta o desenvolvimento do Tester's Quest, um jogo de tabuleiro cooperativo concebido como recurso didático para o ensino de Teste de Software. Desenvolvido por estudantes da disciplina ao longo de quatro encontros presenciais complementados por atividades extraclasse com apoio da professora responsável, a proposta busca integrar, de maneira sutil e contextualizada, os conceitos teóricos relacionados à pirâmide de testes e seus diferentes níveis à prática, permitindo que os estudantes experimentem os princípios e desafios do processo de testes por meio das próprias mecânicas do jogo. Dessa forma, em vez de uma exposição direta de conteúdos, o jogo favorece a internalização dos conceitos de forma intuitiva estimulando o engajamento.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta trabalhos relacionados que utilizam jogos como recurso para o ensino de Teste de Software, contextualizando a proposta do Tester's Quest frente às contribuições existentes. A Seção 3 descreve os aspectos metodológicos adotados para o desenvolvimento do jogo. Na Seção 4, são detalhados os conceitos e os elementos que compõem o Tester's Quest. A Seção 5 apresenta e discute os resultados obtidos com a aplicação do jogo. Por fim, a Seção 6 traz as considerações finais e sugestões para o futuro.

2 Trabalhos Relacionados

A gamificação tem se consolidado como uma alternativa promissora ao modelo tradicional de ensino. Diversos trabalhos têm explorado essa abordagem, aplicando-a tanto na forma de jogos digitais quanto analógicos com o objetivo de tornar o processo de aprendizagem mais motivador e significativo. Nesta seção, são descritos trabalhos que propõem o uso de jogos educacionais aplicados ao ensino de teste de software e áreas correlatas destacando suas características e resultados.

IslandTest [14] é um jogo educativo desenvolvido com tecnologias web que combina elementos lúdicos — como arrastar objetos, resolver enigmas e desativar bombas — com quizzes para engajar os jogadores no aprendizado dos tipos de teste de software. Inspirado no seriado Lost, o jogo utiliza um enredo imersivo no qual o jogador assume o papel de um estudante que precisa aplicar conceitos de testes (unidade, integração, sistema, validação, entre outros) para solucionar desafios em uma ilha misteriosa. Além da proposta narrativa, o jogo estimula uma competição saudável ao ranquear o desempenho dos usuários incentivando a repetição das atividades e a superação dos resultados anteriores. Em relação aos resultados, foi constatado, a partir de uma avaliação com 85 alunos da graduação, um retorno bastante positivo quanto à motivação, usabilidade e percepção de aprendizagem: a maioria dos estudantes relatou altos níveis de satisfação, engajamento e relevância do conteúdo, destacando aspectos como jogabilidade, clareza conceitual e contribuição para o desenvolvimento profissional.

GameTest [11] é um jogo educacional desenvolvido com tecnologias web que introduz uma narrativa inspirada na saga Harry Potter. Nele, o jogador acompanha a trajetória de João, um jovem estudante recém-admitido na fictícia "SoftMun" — uma escola voltada ao ensino de software — que precisa recuperar seu computador perdido ao longo de uma jornada repleta de desafios relacionados a testes de software. O jogo é single-player e estruturado em níveis de dificuldade progressiva (fácil, médio e difícil), nos quais o jogador responde a quizzes com feedback imediato a cada acerto ou erro. A proposta foi avaliada por meio de um questionário aplicado a 31 participantes que opinaram sobre diversos aspectos da sua experiência no jogo. Os resultados indicaram uma recepção positiva, destacando-se a percepção de relevância do jogo no contexto da disciplina de Teste de Software.

O Debug [3] é um jogo de cartas desenvolvido para ensinar conceitos de testes de software como níveis (unidade, integração, sistema, aceitação) e tipos de teste (caixa-preta, caixa-branca, caixa-cinza) de forma lúdica e colaborativa. Inspirado no Uno, o jogo simula o processo de desenvolvimento e teste de software incentivando a reflexão sobre a importância de testes precoces e contínuos para garantir a qualidade do produto. Como forma de avaliação, o jogo foi aplicado em uma turma da disciplina de Teste de Software composta por 39 estudantes durante uma atividade prática em sala de aula na qual diversas equipes desenvolveram jogos educativos. Ao final da dinâmica, Debug foi eleito o jogo preferido entre os participantes.

O iTest Learning [6] foi desenvolvido para auxiliar no ensino do planejamento de testes de software, preenchendo uma lacuna identificada na literatura quanto à falta de ferramentas lúdicas para essa fase específica. Desenvolvido em Java, o jogo simula a elaboração de um plano de testes para projetos hipotéticos, abordando itens como escopo, tipos de teste, níveis de teste, ferramentas e artefatos. Embora não detalhe avaliações quantitativas, o jogo foi projetado para integração em disciplinas acadêmicas com armazenamento de dados para análise docente.

O Jogo das 7 Falhas [9] é um jogo educacional voltado ao ensino de testes de caixa-preta, no qual o jogador deve identificar falhas em funcionalidades de software utilizando técnicas como análise de classes de equivalência e valor-limite. A proposta estimula o raciocínio lógico e a aplicação prática dos conceitos de teste em um ambiente lúdico e desafiador. O jogo foi avaliado em três turmas da disciplina de Engenharia de Software de um curso de Ciência

da Computação. Os resultados indicaram que, além de motivar os alunos, o jogo foi percebido como eficaz para o aprendizado. Em avaliação qualitativa, 87% dos participantes consideraram a atividade agradável e 78% relataram ter gostado, reforçando seu potencial como ferramenta de apoio ao ensino.

BlackBox [15] é um jogo sério de espionagem desenvolvido para ensinar técnicas de teste funcional de software, com foco em estratégias de caixa-preta, como classes de equivalência, análise de valor limite e geração de casos de teste. No jogo, os jogadores assumem o papel de agentes em uma narrativa envolvente e devem resolver missões utilizando conhecimentos de teste para impedir os planos de uma organização criminosa. O jogo busca engajar os alunos por meio de desafios progressivos e uma ambientação lúdica. Em avaliação realizada com 39 estudantes da disciplina de Programação Avançada de Sistemas, o jogo obteve resultados positivos quanto à motivação, imersão, diversão e percepção de aprendizado. Mais de 75% dos alunos relataram sentimentos positivos nos quesitos de atenção, satisfação e competência.

Os trabalhos abordados evidenciam como a gamificação tem sido explorada de diferentes formas no ensino de Teste de Software, por meio de jogos digitais, analógicos, narrativas interativas e simulações. Embora cada proposta tenha suas particularidades, todas compartilham o objetivo comum de promover uma aprendizagem mais ativa, contextualizada e motivadora. Nesse panorama, o Tester's Quest contribui ao incorporar elementos cooperativos e aspectos visuais baseados na pirâmide de testes, buscando reforçar, de maneira intuitiva, a importância de uma abordagem estruturada no processo de verificação e validação de software.

A Tabela 1 apresenta uma síntese comparativa dos jogos educacionais discutidos. Nela, são destacados aspectos centrais de cada proposta, como a dinâmica de interação entre os participantes e o escopo de conteúdo abordado, permitindo uma visualização clara das similaridades e distinções em relação ao Tester's Quest.

Tabela 1: Visão geral dos jogos

Nome	Modalidade	Escopo Central
IslandTest	Individual	Tipos de testes
GameTest	Individual	Conceitos iniciais
Debug	Co-op./Comp.	Processo, níveis e tipos
iTest	Individual	Planejamento de testes
7 Falhas	Individual	Teste de caixa-preta
BlackBox	Individual	Testes funcionais
Tester's Quest	Co-op	Pirâmide de testes

Nota: "Co-op." refere-se a jogos cooperativos e "Comp." a jogos competitivos

3 Metodologia

O Tester's Quest foi desenvolvido ao longo de quatro aulas de 120 minutos cada como parte de uma dinâmica proposta no semestre 2024.1 na disciplina de Teste de Software do curso de Ciência da Computação da Universidade Federal da Paraíba (UFPB). A atividade foi realizada em grupos de estudantes, e tinha como objetivo de criar jogos de tabuleiro que explorassem o conceito discutido mais recentemente em sala: Níveis e Tipos de Teste. O grupo responsável pelo desenvolvimento do Tester's Quest era composto por sete integrantes do curso.

A **primeira aula** foi dedicada à fase de idealização. Nessa etapa, os membros da equipe dividiram-se para pesquisar diferentes jogos de tabuleiro tanto cooperativos quanto competitivos. O processo envolveu a leitura de descrições de jogos, a visualização de vídeos explicativos sobre suas dinâmicas e a anotação de mecânicas que pudessem ser adaptadas ao contexto da disciplina. Dentre os jogos analisados, Pandemic, criado por Matt Leacock e publicado originalmente pela Z-Man Games em 2008 [10], destacou-se como principal referência. Nesse jogo cooperativo, os participantes assumem papéis de especialistas em saúde pública com o objetivo de conter surtos de doenças ao redor do mundo por meio do gerenciamento de ações, recursos e eventos aleatórios representados por cartas — uma lógica que inspirou a ideia de um jogo colaborativo voltado à detecção e correção de bugs em projetos de software. Em relação às mecânicas presentes, o Pandemic ainda foi base para algumas delas tais como um baralho de cartas cuja finalidade é fornecer habilidades aos jogadores e um outro destinado a intensificar a problemática adicionando mais bugs.

A escolha pelo Pandemic foi orientada por um conjunto de requisitos relacionados ao tipo de experiência que se desejava proporcionar com o jogo. Entre os critérios considerados, destacam-se: (i) a presença de uma mecânica cooperativa, que promove o trabalho em equipe, (ii) a presença de elementos de imprevisibilidade, como eventos aleatórios e surgimento de novos problemas, que remetem à natureza dinâmica dos testes de software, (iii) a clareza dos objetivos e das condições de vitória e derrota, facilitando a compreensão das metas pelos jogadores e (iv) o alto fator de rejogabilidade, possibilitado pela aleatoriedade dos elementos de jogo, o que permite que mesmo jogadores experientes vivenciem novas experiências e enfrentem desafios inéditos a cada partida.

Com um ponto de partida definido, a **segunda aula** foi dedicada ao amadurecimento da ideia inicial, à definição das principais mecânicas do jogo e à avaliação da viabilidade das propostas de cada membro. Também foram preenchidos formulários fornecidos pela professora, cujo objetivo era validar quais conceitos relacionados ao tema estavam sendo efetivamente contemplados pelo projeto. Foi durante essa aula que se definiu que a pirâmide de testes — com seus quatro níveis (unitário, integração, sistema e aceitação) — seria o conceito central do jogo. Essa escolha permitiu ancorar a experiência lúdica em um modelo consagrado da engenharia de software facilitando a associação entre o jogo e os conteúdos da disciplina. A partir dessa definição, foram estabelecidos os três principais baralhos que comporiam a dinâmica central do jogo: cartas de evento, cartas de power-up e cartas de bug. Cada tipo de carta teve um protótipo inicial desenhado em papel, contendo seus elementos essenciais e possíveis efeitos no jogo. Além disso, surgiu a ideia da seção de treinamento — originalmente pensada como uma mecânica em que o jogador desistia de jogar por duas rodadas consecutivas e, ao fim delas, ganhava 3 pontos de projeto desde que o processo não fosse interrompido. No entanto, essa abordagem foi posteriormente revisada após sessões de teste, pois deixava o jogador inativo por um período excessivo comprometendo sua experiência.

Com as ideias bem definidas na segunda aula, grande parte do desenvolvimento a partir desse ponto passou a ocorrer em atividades

extraclasse. O design do tabuleiro e das cartas foi realizado utilizando a plataforma Canva com o apoio de alguns assets disponíveis gratuitamente na internet. Para a primeira versão do jogo, foram criadas 15 cartas para cada um dos três baralhos principais. Houve cuidado para garantir que os conteúdos e mecânicas da maioria das cartas estivessem alinhados com a temática da disciplina contribuindo para o caráter educativo da proposta.

Na **terceira aula**, alguns dos jogos elaborados pela turma já começaram a ser testados. No entanto, como o design da primeira versão do Tester's Quest ainda não havia sido finalizado, o grupo realizou um teste preliminar utilizando a versão rascunhada do jogo. Apesar das limitações visuais e estruturais dessa versão inicial, a sessão foi fundamental para colher feedbacks importantes que orientaram ajustes e simplificações posteriores. Um dos principais pontos revisados foi o mecanismo de revelação das cartas de bug. A ideia original previa que algumas cartas — especialmente as de níveis superiores da pirâmide de testes — exigissem múltiplas rolagens de dado para serem reveladas. Por exemplo, uma carta de nível de sistema poderia demandar duas rolagens bem-sucedidas com valor mínimo de 4 em cada uma. Contudo, durante os testes, essa abordagem se mostrou excessivamente complexa e pouco intuitiva, o que comprometeu o ritmo do jogo. Assim, optou-se por simplificar a mecânica: todas as cartas passaram a requerer apenas uma única rolagem de dado com valor igual ou superior ao custo de revelação indicado. Essa decisão contribuiu para tornar a jogabilidade mais fluida e acessível.

Com os ajustes realizados nas mecânicas do jogo, a **quarta aula** foi dedicada a uma rodada de testes mais estruturada na qual integrantes de outros grupos puderam experimentar a primeira versão completa do Tester's Quest. Desta vez, com a parte visual totalmente finalizada, foi possível colher feedbacks mais qualificados. Durante a atividade, cada grupo visitante recebia uma explicação rápida das mecânicas principais, e pelo menos um dos autores do jogo permanecia à mesa para acompanhar a partida, tirar dúvidas e garantir o bom andamento da experiência.

A Figura 1 apresenta um fluxograma que sintetiza o processo de concepção e desenvolvimento do Tester's Quest, resumindo as principais atividades realizadas em cada uma das quatro aulas da disciplina.

Figura 1: Fluxo das quatro aulas dedicadas ao desenvolvimento do Tester's Quest

4 Tester's Quest

4.1 Ideia e Caracterização

Inspirado no jogo de tabuleiro Pandemic, Tester's Quest importa o aspecto cooperativo associado ao gerenciamento de recursos buscando emular de maneira lúdica o cotidiano de uma equipe de desenvolvimento com ênfase na etapa de teste de software.

A estratégia de teste de software adotada pelo jogo Tester's Quest baseia-se em um modelo conceitual expandido da pirâmide de testes proposta por Mike Cohn em seu livro Succeeding with Agile: Software Development Using Scrum [2]. No modelo original, o autor propõe uma estrutura em níveis priorizando os testes unitários, seguidos pelos testes de integração e, em menor proporção, pelos testes de interface. A adaptação presente no jogo utiliza os níveis sugeridos no Guide to the Software Engineering Body of Knowledge Version 4.0 [16] substituindo o nível de interface por um nível de sistema — que abrange aspectos mais amplos da aplicação — e acrescentando o nível de aceitação responsável por verificar se o comportamento do sistema atende aos requisitos do cliente.

A principal mecânica derivada desse modelo refere-se ao crescente custo de solucionar os problemas à medida que eles são encontrados em níveis superiores da pirâmide. Por exemplo, quando um problema é detectado no nível de aceitação, sua correção é mais onerosa, especialmente se ainda houver falhas não resolvidas nos níveis inferiores. Assim, o jogo incentiva os jogadores a priorizarem a resolução de falhas da base para o topo, promovendo de forma intuitiva os benefícios de se adotar essa abordagem estruturada de testes. Essa lógica de jogo reforça o princípio de que identificar e resolver problemas precocemente — nos níveis mais baixos — é mais eficiente e evita a propagação de falhas para as camadas superiores.

Em relação ao gerenciamento de recursos, a equipe deve administrar os pontos de projeto que funcionam como a moeda principal do jogo. Esses pontos são utilizados para solucionar bugs, comprar power-ups e lidar com eventos aleatórios negativos que podem impactar o andamento do projeto. Além disso, os jogadores devem se revezar nas idas à seção de treinamento, onde o membro que for deverá responder duas perguntas relacionadas a testes de software em seu próximo turno. Caso acerte ambas, a equipe ganha 3 pontos de projeto adicionais, reforçando o incentivo ao aprendizado e valorizando o conhecimento técnico dos jogadores.

A Figura 2 apresenta uma versão simplificada do tabuleiro de Tester's Quest ilustrando como essas mecânicas foram dispostas visualmente.

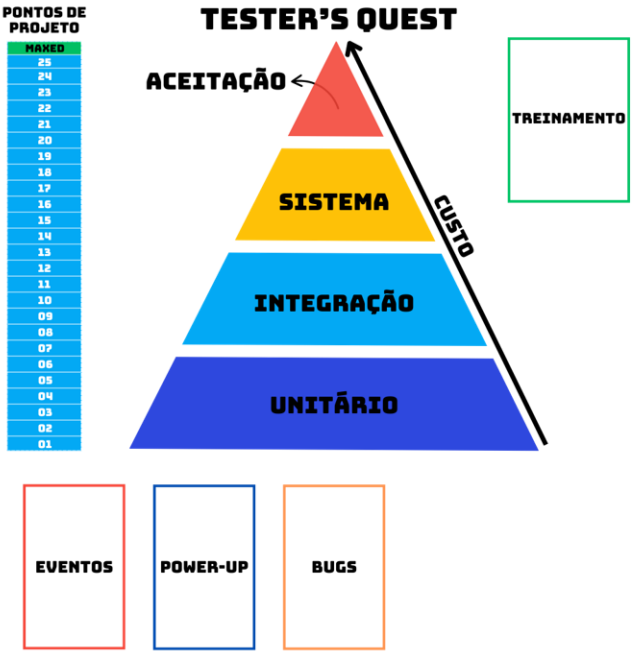

Figura 2: Tabuleiro do Tester's Quest

4.1.1 Conceitos Abordados no Jogo. A concepção do Tester's Quest está fundamentada em conceitos centrais da disciplina de Teste de Software, que são explorados de forma contextualizada ao longo da partida. Dentre os principais, destacam-se:

a) **Pirâmide de Testes**: Estrutura central do jogo, com quatro níveis — teste unitário, integração, sistema e aceitação — conforme o SWEBOK [16].
b) **Custo da correção de bugs por nível**: Quanto mais alto o nível, maior o custo de correção.
c) **Tipos e impactos de bugs**: Simulados pelas cartas de bug específicas de cada nível.
d) **Gestão de recursos em testes**: Representado pela dinâmica de pontos de projeto.
e) **Conceitos teóricos gerais**: Trabalhados por meio das perguntas da seção de treinamento, que abordam tópicos como técnicas de teste, níveis e tipos de teste, elaboração de casos de teste, entre outros.

4.2 Cartas

4.2.1 Cartas de Bug. As cartas de bug são classificadas em quatro categorias correspondentes aos níveis que ocupam da pirâmide de testes: unitário, integração, sistema e aceitação. No início da partida, elas são posicionadas no tabuleiro com o verso voltado para cima de acordo com seu respectivo nível. Cada carta possui um valor de revelação que representa a dificuldade de identificação do bug — esse valor deve ser atingido ou superado com uma rolagem de dado para que a carta possa ser revelada.

Uma vez revelada, a carta é virada e exibe suas informações completas que incluem: um título, o custo em pontos de projeto necessário para sua resolução, uma descrição contextual e o efeito adverso que o bug causa no jogo enquanto não for corrigido. A Figura 3 apresenta dois exemplos de cartas de bug, ilustrando diferentes níveis da pirâmide e evidenciando suas variações de complexidade e impacto no jogo.

Figura 3: Exemplo de cartas de bug dos níveis de aceitação e unitário

4.2.2 Cartas de Evento. As cartas de evento são puxadas no início do turno de cada jogador (exceto nos casos em que o jogador opta por ir à seção de treinamento). Essas cartas introduzem elementos aleatórios ao jogo, podendo trazer consequências positivas ou negativas, o que contribui para a imprevisibilidade das partidas e aumenta o fator replay. As cartas ficam organizadas em um deck próprio no tabuleiro posicionadas com o verso voltado para cima — dessa forma, os jogadores não sabem de antemão se o próximo evento será favorável ou desfavorável. A Figura 4 apresenta dois exemplos de cartas de evento: uma com efeito negativo e outra com impacto benéfico.

Figura 4: Exemplo de cartas de evento negativo e positivo

4.2.3 Cartas de Power-up. As cartas de power-up também estão organizadas em um deck próprio com o verso voltado para cima. Elas podem ser adquiridas logo após a ativação da carta de evento mediante o pagamento de 2 pontos de projeto. Diferentemente dos eventos, os power-ups não têm ativação imediata — o jogador pode mantê-los em mãos e decidir o melhor momento para utilizá-los permitindo um uso estratégico e oportuno durante a partida. Embora o conteúdo da carta só seja revelado após a compra, o jogador tem a garantia de que o efeito será benéfico trazendo vantagens que auxiliam o progresso da equipe ou mitigam os efeitos negativos em curso. A Figura 5 apresenta três exemplos de cartas de power-up.

Figura 5: Exemplo de cartas de power-up

4.3 Regras

4.3.1 Preparação do Jogo. Os jogadores devem se posicionar em um círculo ao redor do tabuleiro e escolher um dos quatro personagens disponíveis — idealmente deve se jogar com 3 ou 4 pessoas. A escolha é meramente cosmética, sem impacto nas regras ou habilidades no jogo.

Com todos devidamente posicionados, são distribuídas as cartas de bugs nos níveis da pirâmide da seguinte forma: 6 cartas no nível unitário, 4 no nível de integração, 2 no nível de sistema e 1 no nível de aceitação. As cartas restantes formam um deck que pode ser utilizado para introduzir novos bugs ao decorrer do jogo. Além disso, as cartas de evento e power-ups devem ser embaralhadas e atribuídas aos seus respectivos campos no tabuleiro para serem usadas no decorrer do jogo.

4.3.2 Dinâmica de Turno. A equipe começa o jogo com **15 pontos de projeto** e pode acumular o **máximo de 25**. No início de cada turno, o jogador tem duas opções:

(1) Ir ao **treinamento** — desde que já tenha passado ao menos duas rodadas desde a última vez que participou dele.
(2) Seguir o **fluxo normal do turno**, que consiste nas três seguintes etapas:
   a) Puxar uma carta de evento com ativação imediata;
   b) Decidir se deseja adquirir um power-up;
   c) Ou jogar o dado para tentar revelar um bug ou resolver algum outro que já esteja revelado — **não é permitido fazer as duas coisas no mesmo turno**.

Optar pelo treinamento significa renunciar ao fluxo normal do turno. Nesse caso, o jogador se prepara para responder duas perguntas relacionadas à disciplina de Teste de Software no turno seguinte. Se for bem-sucedido, sua equipe recebe 3 pontos de projeto como recompensa. Após participar do treinamento, o jogador deverá aguardar pelo menos duas rodadas para poder utilizá-lo novamente incentivando o revezamento entre os participantes. Além disso, só pode haver uma pessoa no treinamento por vez.

4.3.3 Condições de Vitória e Derrota. A equipe vence a partida se conseguir solucionar todos os bugs antes que os pontos de projeto se esgotem. Por outro lado, o jogo é perdido caso a equipe atinja zero pontos de projeto em qualquer momento da partida.

5 Aplicações e Resultados

Após o desenvolvimento do Tester's Quest, o jogo foi aplicado em dois contextos distintos no âmbito da disciplina de Teste de Software ofertada no curso de Ciência da Computação da Universidade Federal da Paraíba. O objetivo dessas aplicações foi observar o comportamento dos alunos durante as partidas, verificar a coerência entre as mecânicas propostas e os objetivos pedagógicos e identificar possíveis melhorias a partir da interação dos jogadores com o jogo.

5.1 Primeira Aplicação: Turma 2024.1

A primeira aplicação do Tester's Quest foi realizada durante o semestre letivo de 2024.1, como parte da quarta aula prática da disciplina de Teste de Software — conforme descrito na seção de Metodologia. Nessa oportunidade, o jogo foi testado por integrantes de outros grupos, com a mediação de ao menos um dos autores para suporte durante a partida.

Os resultados foram bastante positivos, especialmente no que diz respeito ao balanceamento geral da partida. Os jogos duraram entre 25 e 45 minutos, apresentando uma progressão dinâmica e envolvente. As cartas de evento contribuíram significativamente para o ritmo do jogo, alternando momentos de tranquilidade e pressão, o que gerou reações entusiasmadas por parte dos jogadores. Contudo, foi identificado um ponto de melhoria relevante: as cartas de power-up não estavam sendo utilizadas com frequência, uma vez que seus efeitos não compensavam o custo de aquisição em pontos de projeto. Essa percepção levou a uma reformulação posterior dessas cartas, tornando-as mais atraentes e impactantes no contexto da partida.

Para avaliar a recepção do jogo, a professora distribuiu uma **cartela de avaliação** aos participantes, na qual os alunos podiam pontuar diversos aspectos do jogo, sugerir melhorias e indicar os jogos que mais se destacaram. Como mostra a Figura 6, o Tester's Quest obteve um resultado interessante, sendo um jogo bem avaliado.

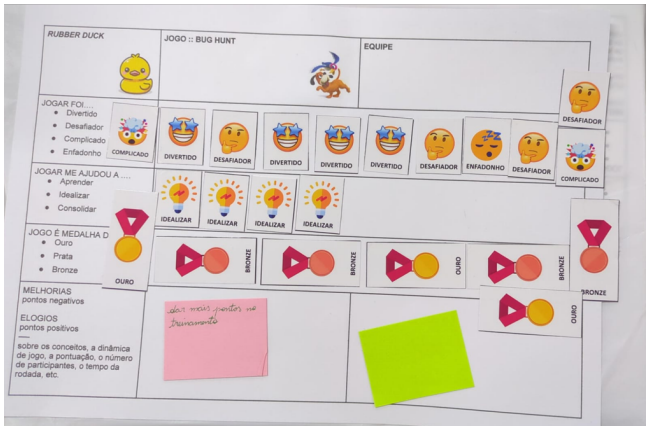

Figura 6: Cartela de Avaliação. O nome "Bug Hunt" se trata do nome provisório que foi posteriormente alterado para "Tester's Quest"

5.2 Segunda Aplicação: Turma 2024.2

Além da atividade estruturada, o Tester's Quest foi reaplicado no semestre seguinte (2024.2), a convite da professora responsável pela disciplina, como uma atividade extracurricular. Nessa ocasião, cerca de 12 estudantes tiveram oportunidade de experimentar o jogo ao longo de 120 minutos, tempo correspondente ao período regular da aula. A maioria dos participantes não possuía conhecimento prévio sobre teste de software, tendo seu primeiro contato com o conteúdo durante a disciplina.

O objetivo da aplicação foi inspirar os alunos no processo de criação dos seus próprios jogos didáticos, uma vez que o desenvolvimento de jogos é uma atividade avaliativa recorrente proposta pela docente. Embora essa nova aplicação não tenha envolvido coleta formal de dados, foi possível observar o divertimento dos participantes com manutenção do tempo médio das partidas. A professora acompanhou ativamente a dinâmica, fazendo observações pontuais e reforçando a relevância da atividade como ferramenta de apoio ao ensino.

Vale ressaltar que, nesse momento, as cartas de power-up ainda não estavam atualizadas, uma vez que as novas versões não haviam sido impressas. Entretanto, essa aplicação foi crucial para a identificação de um problema recorrente relacionado à mecânica da seção de treinamento: alguns jogadores utilizavam repetidamente essa opção, participando pouco das demais ações do jogo. Como resposta, foi implementada a regra de que o jogador só pode retornar ao treinamento após duas rodadas consecutivas de espera, promovendo assim um maior revezamento entre os membros da equipe e uma experiência de jogo mais equilibrada

5.3 Limitações e Pontos de Melhoria

Embora os resultados obtidos nas duas aplicações do Tester's Quest tenham sido majoritariamente positivos, algumas limitações foram identificadas ao longo do processo de desenvolvimento e aplicação do jogo. Uma limitação do Tester's Quest está na forte dependência da mecânica de treinamento para a exploração dos conteúdos técnicos mais específicos da disciplina. Apesar do jogo conseguir abordar de forma lúdica conceitos gerais da pirâmide de testes e a dinâmica de resolução de bugs, os detalhes mais aprofundados — como diferenças entre tipos de teste e técnicas específicas de validação — são trabalhados principalmente por meio das perguntas aplicadas durante o treinamento. Dessa forma, parte do sucesso pedagógico do jogo fica condicionado à qualidade e à diversidade dessas perguntas bem como à mediação ativa do instrutor para garantir que os principais conceitos sejam efetivamente assimilados pelos jogadores.

Outro ponto que merece atenção refere-se à padronização visual do jogo. Embora o formato e fontes utilizados na elaboração das cartas tenha sido padronizado para manter a consistência na apresentação das informações, o estilo das ilustrações pode variar entre os diferentes tipos de cartas. Essa variação decorre da utilização de recursos visuais disponíveis no Canva, que, apesar de facilitarem a prototipagem rápida, resultaram em uma mistura de estilos artísticos que alguns usuários podem perceber como desconexa. Embora essa heterogeneidade não comprometa a funcionalidade do jogo, recomenda-se um refinamento visual para futuras versões.

Por fim, apesar dos ajustes realizados no balanceamento das cartas de power-up e na mecânica de treinamento, a versão revisada do Tester's Quest ainda não foi testada em ambiente acadêmico. Portanto, não há dados disponíveis sobre o impacto dessas mudanças na experiência dos jogadores e na eficácia pedagógica do jogo. No entanto, acredita-se que tais melhorias possam contribuir para uma experiência mais equilibrada e envolvente ou, no pior dos casos, mantenham o nível de engajamento e aprendizado já observado nas versões anteriores. Testes futuros serão fundamentais para validar essas hipóteses.

6 Considerações Finais e Possibilidades Futuras

O desenvolvimento de Tester's Quest surgiu como uma tentativa de explorar os benefícios da gamificação através de uma ferramenta lúdica voltada ao ensino de conceitos relacionados à pirâmide de testes buscando promover uma aprendizagem significativa por meio da cooperação e da simulação de desafios. O jogo foi desenvolvido em um ambiente educacional colaborativo, testado em diferentes contextos e continuamente aprimorado a partir de feedbacks coletados durante sua aplicação.

Os resultados observados apontam para um aumento no engajamento dos alunos, na participação ativa durante as aulas e na facilidade de assimilação dos conceitos. Essa percepção está alinhada com os achados de Nascimento et al. [12], que demonstram como abordagens gamificadas no ensino de Teste de Software elevam a motivação dos discentes e contribuem para a retenção de conteúdos abstratos. Tais evidências reforçam a relevância da gamificação como estratégia pedagógica, em consonância com a crítica de Barr e Tagg [1], que defendem a substituição do ensino centrado na simples transmissão de conteúdos por modelos que priorizem a construção ativa do conhecimento por meio da experiência. Apesar das limitações ainda presentes, como a dependência da mecânica de treinamento para a exploração de conteúdos mais específicos e a necessidade de maior refinamento visual, o jogo mostrou-se funcional, bem recebido e com potencial de replicação em turmas futuras.

Além dos benefícios observados entre os jogadores, o processo de desenvolvimento do Tester's Quest também foi extremamente enriquecedor para os autores enquanto estudantes da disciplina de Teste de Software. Durante a criação do jogo, fomos desafiados a compreender com profundidade os diferentes níveis e tipos de teste para que as mecânicas e elementos lúdicos fossem coerentes com os conceitos técnicos. Essa necessidade nos levou a estudar de maneira mais ativa, debatendo interpretações, propondo metáforas e avaliando como representar conteúdos abstratos em ações concretas dentro do jogo. A construção coletiva ainda favoreceu o desenvolvimento de habilidades como trabalho em equipe, comunicação e senso crítico, fundamentais na formação de profissionais da área de computação.

Como proponentes do jogo, observar as partidas conduzidas por outros colegas também foi um exercício valioso de escuta e análise. A maneira como os jogadores reagiam às regras, interagiam entre si e tomavam decisões revelou nuances que muitas vezes não havíamos previsto. A experiência de ver o jogo "ganhar vida" nas mãos de outras pessoas evidenciou a importância da clareza nas instruções, do equilíbrio nas mecânicas e da adequação do

conteúdo ao público-alvo. Essas observações não apenas orientaram melhorias no projeto, mas também ampliaram nossa compreensão sobre o papel do feedback no processo de desenvolvimento de produtos educacionais.

Como possibilidades futuras, considera-se a criação de um banco ampliado e categorizado de perguntas para a seção de treinamento e a elaboração de um guia didático para apoiar professores interessados em aplicar o jogo. Além disso, outras melhorias avaliadas incluem o redesenho da mecânica de compra dos power-ups. Em vez de manter as cartas ocultas, propõe-se a implementação de uma loja rotativa, na qual três cartas de power-up ficam visíveis aos jogadores e são atualizadas a cada três rodadas. Essa mudança permitiria decisões mais estratégicas por parte da equipe estimulando a análise de riscos e recompensas com base nas necessidades da partida. Também está em análise a introdução de um sistema de classes para os personagens do jogo. Com isso, a escolha de personagem deixaria de ser meramente cosmética e passaria a influenciar mecanicamente a partida, atribuindo habilidades específicas a cada um deles. Por exemplo, certos personagens poderiam ter mais facilidade para resolver determinados tipos de bugs, o que promoveria uma camada adicional de estratégia e personalização às partidas.

DISPONIBILIDADE DE ARTEFATO

Os artefatos desenvolvidos ao longo deste trabalho, incluindo o tabuleiro, as cartas (evento, power-up e bug), além do manual de instruções estão disponíveis para consulta e uso livre com fins educacionais. Acesse os materiais em: Artefatos.

REFERÊNCIAS

[1] Robert B. Barr and John Tagg. 1995. From teaching to learning: A new paradigm for undergraduate education. *Change: The Magazine of Higher Learning* 27, 6 (Nov./Dec. 1995), 12–26. doi:10.1080/00091383.1995.10544672

[2] Mike Cohn. 2015. *Succeeding with Agile: Software Development Using Scrum.* Pearson, India.

[3] Carolina Costa, Yuska Aguiar, Luiz Souza, Jayanne Morais, and Pedro Silveira. 2024. Card Game as an Educational Strategy in Software Testing: Debug a game to explore content on Levels and Types of Testing. In *Anais do XXIII Simpósio Brasileiro de Qualidade de Software* (Bahia/BA). SBC, Porto Alegre, RS, Brasil, 605–612. https://sol.sbc.org.br/index.php/sbqs/article/view/32988

[4] Paul Curzon, Peter W. McOwan, Nicola Plant, and Laura R. Meagher. 2014. Introducing teachers to computational thinking using unplugged storytelling. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (Berlin, Germany) *(WiPSCE '14)*. Association for Computing Machinery, New York, NY, USA, 89–92. doi:10.1145/2670757.2670767

[5] Conselho Nacional de Educação (Brasil). 2016. *Resolução CNE/CES nº 5, de 16 de novembro de 2016.* Retrieved July 22, 2025 from https://portal.mec.gov.br/docman/novembro-2016-pdf/52101-rces005-16-pdf/file Diário Oficial da União, Seção 1, 17 nov. 2016, pp. 22–24.

[6] Virgínia Farias de Oliveira Sousa. 2012. iTest Learning: um jogo para o apoio ao ensino de testes de software. (2012). http://repositorio.ufc.br/handle/riufc/25265 TCC (graduação em Sistemas de Informação) – Universidade Federal do Ceará, Campus Quixadá.

[7] Jacques Delors. 1996. *Learning: The Treasure Within: Report to UNESCO of the International Commission on Education for the Twenty-First Century.* UNESCO Publishing, Paris, France.

[8] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. 2011. From game design elements to gamefulness: defining "gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments* (Tampere, Finland) *(MindTrek '11)*. Association for Computing Machinery, New York, NY, USA, 9–15. doi:10.1145/2181037.2181040

[9] Lucio Diniz and Rudimar Dazzi. 2011. Jogo Digital para o Apoio ao Ensino do Teste de Caixa-Preta. In *Anais do X Simpósio Brasileiro de Qualidade de Software* (Curitiba). SBC, Porto Alegre, RS, Brasil, 231–245. doi:10.5753/sbqs.2011.15398

[10] Matt Leacock. 2025. *Pandemic.* Retrieved May 20, 2025 from https://www.zmangames.com/game/pandemic/

[11] Elizieb Luiz, Alana Pinheiro, Alysson Milanez, Maria Silva, and Pedro Silva. 2024. GameTest: Um jogo para praticar teste de software. *Informática na Educação: teoria & prática* 27 (01 2024), 1–18. doi:10.22491/1982-1654.138735

[12] Eduardo Nascimento, Roberta Coelho, Charles Madeira, Kláudio Medeiros, Lucas Silva, and Carlos Oliveira Neto. 2024. Definição e Avaliação de uma Abordagem Gamificada para o Ensino de Teste de Software. In *Anais do XXXVIII Simpósio Brasileiro de Engenharia de Software* (Curitiba/PR). SBC, Porto Alegre, RS, Brasil, 455–465. doi:10.5753/sbes.2024.3523

[13] The Joint Task Force on Computing Curricula. 2004. *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering.* Technical Report. New York, NY, USA.

[14] Rafael Queiroz, Fabrício Pinto, and Paulo Silva. 2019. IslandTest: jogo educativo para apoiar o processo ensino-aprendizagem de testes de software. In *Anais do XXVII Workshop sobre Educação em Computação* (Belém). SBC, Porto Alegre, RS, Brasil, 533–542. doi:10.5753/wei.2019.6658

[15] Nayara Ribeiro, Rosiane Vasconcelos, Davi Viana, and Luis Rivero. 2017. Avaliando a Viabilidade do BlackBox em Sala de Aula: Um Jogo Sério para Ensino de Teste Funcional de Software. In *Anais do XXVIII Simpósio Brasileiro de Informática na Educação (SBIE 2017), VI Congresso Brasileiro de Informática na Educação (CBIE 2017)*. 817–826. https://www.researchgate.net/publication/320996858

[16] Hideaki Washizaki. 2024. *Guide to the Software Engineering Body of Knowledge (SWEBOK), Version 4.0.* IEEE Computer Society, Los Alamitos, CA.