GABRIEL GONÇALVES MOREIRA

UM JOGO EDUCACIONAL PARA ENSINO DO PROCESSO DE TESTE DE SOFTWARE

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Ciências.

Curitiba
2021

GABRIEL GONÇALVES MOREIRA

UM JOGO EDUCACIONAL PARA ENSINO DO PROCESSO DE TESTE DE SOFTWARE

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Ciências.

Área de concentração: Ciência da Computação

Orientadora: Profa. Dra. Sheila Reinehr

Co-orientador: Prof. Dr. Frederick van Amstel

Curitiba
2021

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná

Sistema

Moreira, Gabriel Gonçalves
M838j Um jogo educacional para ensino do processo de teste de software / Gabriel
2021 Gonçalves Moreira ; orientadora: Sheila Reinehr ; co-orientador: Frederick van Amstel. – 2021.
xvi, 185 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2021
Bibliografia: f. 125-133

1. Software – Testes. 2. Jogos educativos. 3. Jogos eletrônicos. I. Reinehr, Sheila. II. Amstel, Frederick van. III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. IV. Titulo.

CDD 23. ed. – 005.14

Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central
Margareth Jansson Zanetti – CRB 9/1117

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

Curitiba, 14 de março de 2023.

22-2023

DECLARAÇÃO

Declaro para os devidos fins, que **GABRIEL GONÇALVES MOREIRA** defendeu a dissertação de Mestrado intitulada "**UM JOGO EDUCACIONAL PARA ENSINO DO PROCESSO DE TESTE DE SOFTWARE**", na área de concentração Ciência da Computação no dia 16 de novembro de 2021, no qual foi aprovado.

Declaro ainda, que foram feitas todas as alterações solicitadas pela Banca Examinadora, cumprindo todas as normas de formatação definidas pelo Programa.

Por ser verdade firmo a presente declaração.

Prof. Dr. Emerson Cabrera Paraiso
Coordenador do Programa de Pós-Graduação em Informática

DEDICATÓRIAS

Dedico todos os meus esforços a meus pais, Antônio José Moreira Sombra e Maria Juciane Gonçalves, e ao meu irmão, Antônio Gustavo Gonçalves Sombra.

AGRADECIMENTOS

As Professoras Dra. Sheila Reinehr e Dra. Andreia Malucelli e ao Professor Dr. Frederick van Amstel, pela excelente orientação e acompanhamento do trabalho.

Aos professores participantes da banca examinadora Dr. Carlos Silla e Prof. Dr. Fábio Binder pelo tempo, pelas valiosas colaborações e sugestões.

Aos colegas de curso, principalmente do grupo de pesquisa GPES, pelas reflexões, críticas e sugestões fornecidas.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil – (CAPES) – Código de Financiamento 001.

“Uso apenas as células cinzentas. A sorte, deixo-a para os outros.”
(Hercule Poirot)

Agatha Christie

RESUMO

Apesar da reconhecida importância, há uma clara lacuna de conhecimento entre o que é ensinado na universidade e o que é tratado na indústria para tópicos relacionados a testes de software, o que gera uma grande carência no mercado por profissionais especializados. Uma das principais razões para isso pode ser o ensino de teste de software na graduação, que apresenta diversos problemas que vão desde a estruturação dos cursos que não abordam com profundidade testes de software, como aspectos motivacionais relacionados ao aprendizado do tema. Apesar de ser o mais utilizado no ensino de testes no Brasil, o método tradicional de ensino não consegue simular e pode não facilitar nem motivar o aprendizado do processo de teste e suas etapas. Com isso, os estudantes de graduação podem não entender o processo e seus componentes e nem se motivarem para aprender sobre o conteúdo. Os jogos educacionais são uma abordagem que pode facilitar o ensino do processo de teste de software e promover a motivação dos estudantes. Apesar de existirem vários jogos educacionais para ensino de testes de software, nenhum jogo aborda o processo de teste de software como um todo, com seus papéis, artefatos e atividades. Este trabalho apresenta o desenvolvimento de um jogo educacional para apoiar e motivar o ensino do processo de teste de software. Para o desenvolvimento do jogo foram feitos diversos *playtestings* para verificar se a nova proposta era capaz de motivar e ensinar o conteúdo proposto. Como forma de avaliar o cumprimento dos objetivos da pesquisa, foi feito um experimento com 16 participantes e foi possível verificar um ganho de conhecimento específico dos participantes após a experiência com o jogo. Além disso, efeitos positivos foram observados em relação aos aspectos motivacionais percebidos pelos jogadores com o jogo para o aprendizado do tema.

Palavras-chaves: teste de software, jogos educacionais, ensino de teste de software, processo de teste de software, jogos sérios.

ABSTRACT

Despite its recognized importance, there is a clear knowledge gap between what is taught at university and what is treated in the industry for topics related to software testing, which creates a great shortage in the market for specialized professionals. One of the main reasons for this may be the teaching of software testing at undergraduate level, which presents several problems that range from courses structuring that do not address software testing in depth as motivational aspects related to test learning. Despite being the most used approach in software tests teaching in Brazil, the traditional teaching method cannot simulate and may not facilitate or motivate the learning of the test process and its steps. As a result, undergraduate students may not understand the process and its components, nor be motivated to learn about the content. One approach that can facilitate teaching the software testing process and promote student motivation is educational games. Although there are several educational games for teaching software testing, no game addresses the entire software testing process, with its roles, artifacts and activities. This work presents the development of an educational game to support and motivate the teaching of the software testing process. For the development of the game, several playtestings were carried out to verify if the new proposal was able to motivate and teach the proposed content. To assess the fulfillment of the research objectives, an experiment was carried out with 16 participants, and it was possible to verify a gain in specific knowledge of the participants after the experience with the game. Furthermore, positive effects were observed in relation to the motivational aspects perceived by players with the game for content learning.

Keywords: software testing, educational games, software test teaching, software test process, serious games.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

DSR	Design Science Research
FEDS	Framework for Evaluation in Design Science
GEQ	Game Engagement Questionnaire
ISTQB	International Software Testing Qualification Board
RUP	Rational Unified Process

CAPÍTULO 1 - INTRODUÇÃO

"O começo de todas as ciências é o espanto de as coisas serem o que são."

Aristóteles

A qualidade de software é importante para o sucesso da empresa que o produz, bem como para a satisfação dos clientes e consequente aceitação do produto. A falta de qualidade de software pode resultar em prejuízos financeiros, insatisfação dos usuários/clientes, danos ao meio ambiente e, inclusive, pôr a vida de pessoas em perigo (LAPORTE, et al., 2007).

Uma das técnicas mais utilizadas para garantir a qualidade de um software é a de testes de software. Testar um software é "o processo de executar um programa com o objetivo de encontrar erros" (MYERS, et al., 2004) antes de distribuí-lo aos seus usuários. Ainda, caracterizando testes de software, o SWEBOK (IEEE Computer Society, 2014) apresenta a seguinte definição:

> Teste de software consiste na verificação dinâmica de que um programa fornece comportamentos esperados em um conjunto finito de casos de teste, selecionados adequadamente a partir do domínio de execução geralmente infinito

A atividade de teste de software deve ser realizada em todas as fases do processo de desenvolvimento, porque falhas podem estar presentes em todas elas (EKWOGE, et al., 2017). Acompanhando o desenvolvimento do software, o processo de teste de software deve ser realizado a fim de dar maiores garantias de que os testes de software atingem seus objetivos, ou seja, garantem uma maior qualidade ao sistema testado.

Apesar da reconhecida importância do assunto (DELAMARO, et al., 2016); (SILVA, et al., 2012), testes de software é testes de software é uma das áreas mais

desafiadoras. Em um questionário com 78 estudantes realizado por (OGUZ & OGUZ, 2019) verificou que testes de software é a área de conhecimento considerada mais desafiadora pelos estudantes. Mais preocupante do que isso, testes de software foi também uma das áreas em que mais estudantes informaram não ter conhecimento sobre.

Como consequência da visão desafiadora e pouco conhecimento, estudos apontam que um *gap* de conhecimento entre o que é ensinado e o que é aplicado na indústria. (GAROUSI, et al., 2019) analisou 33 artigos para classificar as principais áreas de engenharia de software que apresentam *gaps* de conhecimento entre o que é trabalhado na indústria e o que é ensinado na academia. Ao ranquear as áreas analisadas nos artigos, testes de software aparece como uma das áreas de mais importância e também com um dos maiores *gaps* de conhecimento, revelando uma necessidade da indústria por profissionais com mais conhecimento sobre testes.

Dentre os assuntos mais requeridos para os profissionais da indústria, destaca-se a capacidade de gerenciamento e execução do processo de teste. Taipale, Smolander e Kälviäinen (2005) mostraram que o processo de teste de software ficou em terceiro lugar dentre nove tópicos que especialistas em teste de software da indústria ordenaram como mais importantes na época. Com resultados similares, em um *survey* de opinião realizado mais recentemente com 246 profissionais da indústria de software do Canadá, Turquia, Dinamarca, Áustria e Alemanha, o gerenciamento das atividades do processo de teste foi o tópico considerado mais desafiadora pelos praticantes (GAROUSI, et al., 2017).

Apesar da requisição da indústria por profissionais mais capacitados no gerenciamento e execução das atividades do processo de teste, este é um dos tópicos menos abordados em cursos de teste de software. Em um mapeamento sistemático da literatura realizado por Gaurousi, et al. (2020) o processo de testes foi o tópico menos abordado por cursos educacionais. De forma similar Paschoal e De Souza (2018) observaram que menos da metade dos professores de graduação abordam o processo de teste em suas disciplinas, tornando este tópico o terceiro menos ensinados pelos discentes.

1.1 Motivação

É clara a lacuna de conhecimento entre o que é ensinado na universidade e o que é tratado na indústria para tópicos relacionados a processos de software e habilidades pessoais, como mostra Scataton et al (2018). Em relação a testes de software, há uma dificuldade no entendimento do processo e estágios dos testes, o que pode levar à desmotivação dos estudantes para o aprendizado desse assunto (VALLE, et al., 2017). Uma razão para isso pode estar relacionada às deficiências no ensino e motivação para ensinar tais tópicos para os estudantes na graduação.

Há vários problemas relacionados ao ensino de testes na graduação, como a estruturação dos cursos. Com a análise dos currículos de algumas das principais universidades brasileiras e do exterior, Valle, Barbosa e Maldonado (2015) observaram que, com raras exceções, essa atividade é abordada apenas em disciplinas de Engenharia de Software e/ou Fundamentos da Programação, não havendo uma disciplina especificamente dedicada ao ensino de testes de software.

Clegg Rojas e Fraser (2017) também criticam a estruturação dos cursos e apontam que o modo como os engenheiros de software são ensinados é um dos fatores que contribui para que testes de software não tenham um papel tão importante no dia a dia desses profissionais. Segundo os autores, isso ocorre porque o ensino de Engenharia de Software é mais focado na parte criativa de aspectos de design e codificação do que no ensino da atividade de testes de software.

Outro problema relacionado ao ensino de teste tem relação com desmotivação dos estudantes para o aprendizado de testes. Muitos estudantes de computação se sentem desmotivados para aprender conteúdos relacionados a testes de software. Jia e Yang (2013) destacam alguns dos problemas que podem afetar a motivação dos estudantes em relação ao aprendizado de testes de software,: (i) Desconexão entre teoria e prática; (ii) O ensino em sala de aula e o conteúdo do trabalho nos negócios não são os mesmos; (iii) A falta de experiência de teste dos alunos pode levá-los a não conseguir compreender o processo e as etapas do teste; (iv) A experiência de desenvolvimento dos alunos pode ser insuficiente para que eles compreendam o desenvolvimento orientado a testes; (v) O ensino e a prática em sala de aula não estão intimamente relacionados.

A desmotivação sobre o aprendizado de testes pode ser também pela natureza das atividades de teste, que, assim como inspeções, podem ser consideradas atividades “Destrutivas” (ALHAMMAD & MORENO, 2018). A principal razão para essa visão da natureza dos testes pode ser porque eles acabam revelando problemas e erros a serem corrigidos, o que não acontece com atividades “Construtivas”, como codificação e *design*.

Além da natureza pouco motivadora e “destrutiva” dos testes (MYERS, et al., 2011); (ALHAMMAD & MORENO, 2018), existem outros desafios que impactam na motivação e no ensino de testes, como uma provável ineficiência da abordagem tradicional de ensino (SMITH, et al., 2012), que é a abordagem mais utilizada em disciplinas de graduação no ensino de testes no Brasil (PASCHOAL & DE SOUZA, 2018). O uso de métodos tradicionais de ensino, focando apenas na apresentação de slides, em aulas expositivas onde o estudante é um agente passinho no seu processo de aprendizagem pode não ser o mais efetivo para o ensino-aprendizagem de testes de software.

Por conta de restrições de tempo e dificuldade em simular cenários, o modelo tradicional de ensino pode não ser capaz de criar oportunidades para aprendizagem prática em testes de software. Além disso, com o uso das novas tecnologias, ocorreram diversas mudanças nas relações e na maneira de pensar e aprender (SÁNCHEZ, 2014), o que influencia diretamente no modo como os estudantes aprendem. Dessa forma, manter o interesse e a concentração dos estudantes durante o processo de ensino-aprendizagem se tornou um desafio para muitos professores que continuam a adotar o modelo de ensino tradicional. Portanto, a aplicação do modelo tradicional pode não ser suficiente para preparar os estudantes de Ciência da Computação e áreas afins para o ambiente real do mercado.

Para mitigar os desafios, algumas abordagens alternativas foram propostas para estimular o comportamento desejado e aumentar a motivação e o aprendizado dos estudantes, como métodos ativos, como nos trabalhos de (PASCHOAL & SOUZA, 2018), Martinez (2018) e Clarke et al (2011) e jogos educacionais, tais como os apresentados nos trabalhos de Buffardi e Valdivia (2018), Valle et al (2017) e Beppe et al (2018).

Uma outra abordagem que é também um dos métodos ativos de aprendizagem e ganhou espaço nas últimas décadas foi a gamificação, que deriva diretamente da popularização e popularidade dos jogos, e vem tomando espaço no meio educacional (WERBACH & HUNTER, 2012). Quando direcionada à educação, em forma de jogos educacionais ou utilizando mecânicas de jogos em plataformas de ensino, a gamificação é usada para aumentar o prazer e o engajamento do jogador durante o processo de aprendizagem, mantendo-o interessado e motivado a continuar aprendendo (KAPP, 2012).

O objetivo por trás do uso de jogos educacionais para a aprendizagem é aproveitar a motivação intrínseca de um jogador para jogar um jogo e incentivar a aprendizagem de algum conteúdo (MAYER, 2011). O nível de motivação intrínseca também tem consequências cognitivas, provocando maior atenção dos jogadores.

Tendo em vista que teste de software não é uma atividade muito atraente para os estudantes e levando em conta também o pouco tempo que é dedicado para que o conteúdo de testes seja apresentado, a utilização de gamificação, trazendo comportamentos de jogos, é uma estratégia promissora para potencializar o interesse e a motivação dos estudantes quando aprendendo o conteúdo, assim como jogos educacionais.

A fim de prover atividades práticas e que aproximem mais o ensino do ambiente de desenvolvimento de software, vários jogos educacionais foram criados para abordar conteúdos de Engenharia de Software (BATTISTELLA & VON WANGENHEIM, 2016), alguns deles focados especificamente no ensino de testes de software. Utilizando o mapeamento sistemático de (VALLE, et al., 2015) e complementando com revisões bibliográficas da literatura, vários jogos educacionais para o ensino de testes de software foram encontrados e a visão geral deles é apresentada no *Quadro 1* e serão detalhados individualmente no Capítulo 3.

Quadro 1. Jogos educacionais para ensino de testes de software. Fonte: o Autor.

JOGO	AUTORES	LOCAL DE PUBLICAÇÃO
U-TEST	(SILVA & THIRY, 2010)	Defesa de mestrado
JETS	(SILVA, et al., 2011)	RENOTE
Jogo das 7 falhas	(DINIZ & DAZZI, 2011)	Simpósio Brasileiro de Informática na Educação-SBIE
iTest Learning	(FARIAS, et al., 2012)	Simpósio Brasileiro de Engenharia de Software

TestEG	(OLIVEIRA & COSTA, 2013)	Monografia de graduação
Code hunt	(TILLMANN, et al., 2014)	Proceedings of the 7th International Workshop on Search-Based Software Testing
iLearnTest	(RIBEIRO & PAIVA, 2014)	Defesa de mestrado
JoVeTest	(BARBOSA, et al., 2016)	IX Fórum de Educação em Engenharia de Software
BlackBox	(ARAUJO, et al., 2017)	Simpósio Brasileiro de Informática na Educação-SBIE
Code Defenders	(CLEGG, et al., 2017)	IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)
Testing Game	(VALLE, et al., 2017)	2017 IEEE Frontiers in Education Conference (FIE)
Bug Hide-seek	(BUFFARDI & VALDIVIA, 2018)	IEEE Frontiers in Education Conference (FIE)
GreaTest	(BEPPE, et al., 2018)	Simpósio Brasileiro de Engenharia de Software
IslandTest	(QUEIROZ, et al., 2019)	Anais do XXVII Workshop sobre Educação em Computação

Apesar de existirem vários jogos educacionais para ensino de testes de software, nenhum jogo trata sobre o processo de teste de software como um todo, com seus papéis, artefatos e atividades. Algumas atividades individuais de alguns papéis e algumas atividades do processo de testes são abordadas pelos jogos existentes. Além disso, observando alguns trabalhos que abordam alguns tópicos do processo de teste, é possível ver que esses jogos abordam o conteúdo muitas vezes com mecânicas limitadas a perguntas e respostas. A utilização dessas mecânicas pode não motivar os estudantes a aprenderem o conteúdo como também pode não ensinar, uma vez que é necessário que o estudante já saiba do conteúdo para que ele responda as perguntas corretamente.

Dessa forma, os jogos existentes não são capazes de fornecer conhecimento suficiente sobre todas as atividades, papéis, artefatos e problemas que podem surgir durante o processo de teste de software e nem de obter efeitos motivadores para o aprendizado de tais tópicos. Com isso, um jogo onde os estudantes possam simular as atividades do processo pode melhorar o ensino neste tópico e prepará-los melhor para as atividades de testes que eles terão que desempenhar na indústria.

1.2 Objetivos

Este trabalho tem por objetivo **desenvolver um jogo educacional para apoiar o ensino do processo de teste de software**, permitindo que os jogadores possam imaginar a execução do processo de teste de uma maneira lúdica e competitiva.

Pretende-se que o jogo possibilite a imaginação de como pode e como deve ocorrer o processo de teste de software, abordando as atividades, artefatos e papéis presentes no processo, além de problemas e soluções que podem emergir ao longo da execução do processo. O jogo a ser desenvolvido não visa substituir nem excluir a participação de professores e instrutores do processo de ensino-aprendizado do estudante, mas fortalecer, por meio da atividade lúdica, o ensino sobre o processo de testes.

1.2.1 Objetivos específicos

Para se alcançar o objetivo geral proposto neste trabalho, os seguintes objetivos específicos devem ser considerados:

- Identificar estratégias para o ensino do processo de teste de software;
- Implementar um jogo educacional que apoie no ensino do processo de teste de software.
- Avaliar o ganho de conhecimento sobre o processo de teste e software e os efeitos motivacionais do jogo nos jogadores.

1.3 Justificativa

Jogos educacionais têm sido aplicados com frequência nos processos de ensino e aprendizagem da engenharia de software (ES), como mostram Xie, Tillmann e De Halleux (2013). Eles são uma estratégia para minimizar as dificuldades do ensino da ES, que aborda mais aspectos teóricos, uma vez que podem colocar o jogador em situações que envolvem componentes do mundo real, normalmente ausentes nos exercícios de salas de aula (SAVI, 2011). Além disso, têm grande capacidade de divertir pessoas e ao mesmo tempo incentivar o aprendizado por meio de ambientes interativos e dinâmicos (ZENG, et al., 2020).

Considerando a importância do processo de teste para uma maior garantia da qualidade, a falta de jogos que abordem de forma completa o conteúdo do processo

de teste e a capacidade de jogos educacionais em promover o ensino de teste de software, este trabalho é direcionado tanto aos estudantes de graduação em Computação e áreas afins, como também a professores de teste que podem utilizar tais jogos como ferramenta de suporte para ensino do conteúdo.

Do ponto de vista dos estudantes, ao fornecer um ambiente lúdico, motivador e que aborda de forma imaginativa o processo, é esperado que os estudantes que tiverem contato com o jogo sejam capazes de conhecer as atividades do processo, distinguir as responsabilidades dentro do processo, avaliar as situações para tomar decisões sobre problemas que podem ocorrer durante o processo e ter mais motivação quando aprendendo sobre o conteúdo de testes.

Do ponto de vista dos professores de teste de software, o resultado deste trabalho fornecerá mais uma forma de ensinar testes de software, despertando mais e aumentando a motivação dos estudantes para o aprendizado do conteúdo da disciplina. Com tais resultados é esperado que haja uma maior motivação para aplicação de ferramentas, como jogos, para o ensino de conteúdos de testes, considerando que poucos docentes utilizam jogos para ensino de conteúdos de teste de software (PASCHOAL & DE SOUZA, 2018).

1.4 Delimitação de escopo

O jogo a ser criado deverá englobar apenas atividades do processo de teste de software. Para um treinamento efetivo com um jogo educacional, é preciso definir competências a serem treinadas no domínio de jogo (ABNT, 2001). As competências a serem desenvolvidas pelo jogo são as seguintes:

- C1. Organizar as principais atividades que compõem o processo de testes de software;
- C2. Distinguir os principais artefatos de teste para as atividades em que devem ser produzidos;
- C3. Separar as responsabilidades de cada papel do processo de teste;
- C4. Analisar algumas possíveis situações/problemas que comumente ocorrem durante o processo.

Não será do escopo desse projeto conteúdos relacionados às técnicas e critérios para a criação dos casos de teste, os tipos de teste e níveis de teste, uma

vez que esses conteúdos não sobre o processo de teste, apesar de serem relacionados ao processo.

1.5 Estrutura do documento da tese

O Capítulo 1 oferece ao leitor um panorama geral sobre o contexto no qual se insere este trabalho de pesquisa. Neste primeiro capítulo foi apresentado a importância da atividade de testes para a qualidade do software e os problemas que existem no ensino desse tópico de Engenharia de Software nas disciplinas de graduação. Também foram apresentadas algumas abordagens que estão sendo usadas para o ensino desse tópico, como jogos educacionais e métodos ativos de ensino e os benefícios que a combinação das duas abordagens já mostrou ter no ensino de outros conteúdos. Como objetivo do estudo, foi proposto a criação de um jogo que possa ser usado em conjunto com os métodos ativos de ensino para ensino do processo de testes de software.

O Capítulo 2 apresenta o referencial teórico descrito no Capítulo 1, focando nos temas de teste de software, especificamente nos fundamentos, níveis, critérios, medidas e no processo de teste de software. Ainda no Capítulo 2 são apresentados conceitos, características e os benefícios para a educação dos jogos educacionais.

O Capítulo 3 apresenta os trabalhos relacionados a este, que de alguma forma abordam o ensino de teste de software. No Capítulo 3 também são apresentados o funcionamento e as características de cada proposta de jogo educacional, além de, se existir trabalho que tenha sido encontrado apresentando, também são apresentados a forma de avaliação de cada proposta. Por fim, também são apresentadas as características que distinguem o jogo a ser desenvolvido neste trabalho das demais propostas encontradas.

O Capítulo 4 apresenta a estrutura metodológica planejada para a execução do das atividades da pesquisa, que vão desde o estabelecimento dos objetivos da proposta até o desenvolvimento e avaliação do jogo. Levando em consideração as características da pesquisa e seus objetivos, foi escolhida a *Design Science Research* (DSR) como método de pesquisa, tomando como base a definição de atividades definida por (PEFFERS, et al., 2007) para guiar a execução desta pesquisa.

O Capítulo 5 apresenta o artefato (jogo educacional) criado por esta pesquisa, que foi chamado de ProTesters. Neste capítulo é apresentado todo o material de jogo, suas mecânicas e como que as mecânicas estão relacionadas com os objetivos educacionais do jogo. Além disso, como forma de facilitar o entendimento das discussões dos *playtestings* que serão apresentadas no Capítulo 6, são apresentadas também neste capítulo a visão geral das versões utilizadas em cada *playtesting.*

O Capítulo 6 apresenta as avaliações que foram feitas com o jogo em suas várias versões. Neste capítulo são apresentados as sessões de *playtestings,* que foram realizadas antes do experimento com o artefato no seu ambiente final de uso e tinha o objetivo de identificar melhorias ainda durante o desenvolvimento e criação do artefato. Também é apresentado o experimento de avaliação final do jogo com estudantes de graduação. Este capítulo também apresenta os resultados de análises quantitativas e qualitativas geradas pelas respostas dos estudantes aos questionários e discussões levantadas nas avaliações. Por fim, com os resultados das análises, este capítulo mostra o cumprimento dos objetivos da pesquisa em relação a aprendizibilidade e motivação dos estudantes.

O Capítulo 7 conclui este trabalho, apresentando as considerações finais, que inclui as conclusões e as contribuições de pesquisa e as ameaças à validade deste trabalho.

1.6 Considerações sobre o capítulo

Este capítulo de introdução apresentou a importância da atividade de testes para a qualidade do software, alguns dos problemas que existem no ensino desse tópico e algumas abordagens que estão sendo usadas para solucionar o ensino desse tópico em disciplinas de graduação. Algumas das abordagens mais encontradas foram jogos educacionais, que, em conjunto com métodos ativos de ensino apresentam benefícios no aprendizado dos estudantes. No próximo capítulo serão apresentados conceitos sobre testes de software e seus tópicos e jogos educacionais.

CAPÍTULO 2 - REVISÃO DA LITERATURA

“Ninguém é tão ignorante que não tenha algo a ensinar.
Ninguém é tão sábio que não tenha algo a aprender.”

Blaise Pascal

Nesta seção são apresentados os principais conceitos e pesquisas relacionados ao tema deste trabalho, identificados por meio de uma revisão da literatura. Na subseção 2.1 são apresentados os principais conceitos, terminologias, técnicas e critérios relacionados aos testes de software. Na subseção 2.2 são apresentados conceitos e aplicações de jogos educacionais.

2.1 Testes de software

Executar testes é “o processo de executar um programa com a intenção de encontrar erros” (MYERS, et al., 2004). Não é possível garantir que um software está livre de erros, mas, se os testes forem bem executados, a maioria dos erros pode ser descoberta e corrigida e o software pode ser entregue com maior nível de confiabilidade. No SWEBoK (IEEE, 2014) teste de software é uma área do conhecimento específica e está organizada nas seguintes subáreas: Fundamentos, Níveis, Técnicas, Medidas e Processo. Cada uma das subáreas do teste de software será apresentada nas subseções a seguir, seguindo a ordem apresentada pelo SWEBoK.

2.1.1 Fundamentos dos testes de software

O padrão IEEE 610.12 (IEEE Computer Society, 1991) apresenta definições dos principais termos utilizados no contexto de teste de software, sendo eles:

- Defeito (*Fault*) é um passo, processo ou definição de dados incorretos, por exemplo, um comando incorreto;

- Engano (*Mistake*) é a ação humana que causa um resultado incorreto, por exemplo, uma decisão incorreta tomada pelo programador;
- Erro (*Error*) é o resultado incorreto na execução de um programa;
- Falha (*Failure*) é a produção de uma saída incorreta conforme a especificação.

Outros termos que são muito utilizados pelos engenheiros de software no ambiente de testes de software são (DELAMARO, et al., 2016):

- Domínio de Entrada: é o conjunto de todos os possíveis valores que podem ser utilizados para executar um programa;
- Dado de Teste: é um elemento do domínio de entrada de um programa que é escolhido para a execução de um determinado teste;
- Caso de Teste: é um par formado por entradas, um dado de teste do domínio do programa, e por uma saída esperada para a execução do teste;
- Conjunto de Testes: é o conjunto de todos os casos de teste utilizados durante a atividade de teste;
- Oráculo: é o instrumento que decide se a saída obtida pela execução de um teste coincide com a saída esperada. Esse oráculo pode ser uma ferramenta automatizada ou o próprio testador.

2.1.2 Níveis de testes

Há quatro níveis de teste de software que variam em relação à granularidade do código que é testado: teste de unidade, teste de integração, teste de sistema e teste de regressão.

O nível de teste mais baixo é o **teste de unidade** que visa testar os componentes do sistema de forma isolada (PFLEEGER, 2004). O teste de unidade é um teste que deve ser executado pelo programador em um ambiente de testes e deve demonstrar que o programa atende ao conjunto de requisitos especificados. O teste de unidade tem como objetivo testar isoladamente as menores partes que constituem um sistema, pois, assim, é mais fácil identificar qual unidade (por exemplo, método, função ou classe) não está funcionando de maneira correta (DELAMARO, et al., 2016).

O segundo nível de testes de software é o **teste de integração**. Na medida em que os módulos do sistema são desenvolvidos e precisam trabalhar com outros módulos, é preciso verificar se a interação entre eles funciona de maneira adequada, não ocasionando erros (DELAMARO, et al., 2016). Para isso, é necessário elaborar casos de teste que verifiquem as interações e as interfaces entre as unidades do sistema, testando como as unidades interagem entre si e se a interação gera resultados corretos.

O **teste de sistema** verifica se todas as funcionalidades implementadas no sistema estão corretas em relação ao que foi especificado nos artefatos e documentos de requisitos, ou seja, verifica se o sistema como um todo foi implementado de maneira correta em relação ao que foi especificado (DELAMARO, et al., 2016). O teste de sistema envolve a integração de todos os componentes implementados para a criação de uma versão do sistema. Esse tipo de teste verifica a compatibilidade de todos os componentes, analisando se eles interagem de maneira correta entre si por meio de suas interfaces (SOUZA; FALBO; VIJAYKUMAR, 2015).

Além dos três níveis anteriores, há ainda o **teste de regressão**. Esse nível de teste não é realizado durante o processo de desenvolvimento de um software, mas durante o processo de manutenção. Nesse teste é verificado se as modificações, causadas pela alteração ou edição de unidades do sistema, ocasionam a introdução de defeitos. Com isso faz-se o teste de regressão para comprovar que os novos requisitos implementados estão funcionando como o esperado e que eles não inseriram erros nos requisitos que já haviam sido implementados anteriormente (DELAMARO, et al., 2016)

2.1.3 Técnicas de testes

A princípio, os testes têm potencial para revelar todos os defeitos do sistema. Entretanto (DAHL, et al., 1972) afirma que os testes em programas podem ser usados para mostrar a presença de *bugs*, mas nunca a ausência. Isso ocorre, pois, considerando que o conjunto de dados de entrada do programa pode ser muito grande ou infinito, criar todos os casos de teste para todas as entradas e saídas pode ser muito custoso. De modo geral, é impossível criar casos de teste para testar todas as entradas e saídas que um programa pode gerar.

Por ser impossível de garantir a inexistência de defeitos em um software, os testes são utilizados na prática para garantir certa qualidade do produto desenvolvido, mas não que ele não possui defeitos (DELAMARO, et al., 2016); WAZLAWICK, 2013). Por isso, a atividade de testes possui critérios que fornecem algumas diretrizes para a criação de um conjunto reduzido de casos de teste que irão testar o sistema. Os critérios dividem as entradas do programa em diferentes subdomínios. Ao dividir a entrada em subdomínios é possível selecionar apenas algumas entradas de cada subdomínio e assim garantir a cobertura do código-fonte.

Os critérios de teste de software, em nível de programa, podem ser classificados em três diferentes técnicas: funcional, estrutural e baseado em defeitos. O que distingue essas técnicas é a fonte utilizada para criar os requisitos de teste (DELAMARO, et al., 2016). Dependendo de qual artefato se deseja utilizar (e.g. especificação ou código-fonte), é utilizada uma técnica e critérios diferentes para auxiliar na criação de um conjunto eficiente de casos de teste.

Teste funcional

O teste funcional (também conhecido como teste caixa-preta) tem como princípio utilizar a especificação dos requisitos funcionais do sistema para criar os casos de teste, não considerando os detalhes de implementação. Por isso, para esse tipo de teste, o código-fonte do sistema testado é visto como uma caixa preta, na qual são fornecidas as entradas e é verificado se as saídas geradas estão em conformidade com os objetivos especificados (KHAN; SADIQ, 2011; (DELAMARO, et al., 2016); JUNIOR et al., 2012). Nessa técnica o software é verificado pelo ponto de vista do usuário.

Os critérios mais conhecidos da técnica de teste funcional são particionamento por equivalência, análise do valor limite e grafo de causa-efeito. O particionamento por equivalência divide o domínio de entrada de dados de um programa em classes, considerando que, pela especificação do programa, dados em uma mesma classe são tratados da mesma maneira pelo programa. Com isso, qualquer entrada de uma classe pode representar todas as demais entradas daquela classe, pois elas devem se comportar da mesma maneira no programa, ou seja, cada elemento dentro de uma classe tem, teoricamente, o mesmo potencial de revelar um defeito quanto os demais,

pois eles são equivalentes do ponto de vista da especificação. Com isso, o conjunto de dados de entrada que pode ser considerado pelo programa pode ser diminuído para apenas uma entrada para cada classe de equivalência.

O critério análise do valor limite tem como premissa que os casos de teste que exploram os limites entre as classes de equivalência têm maiores chances de encontrar defeitos. Tais casos de teste são formados por entradas que estão exatamente sobre, acima ou abaixo das classes de equivalência. Dessa forma, a análise do valor limite utiliza as classes identificadas pelo particionamento por equivalência, mas adiciona algumas regras para selecionar os elementos das fronteiras entre as classes, e não de forma aleatória dentro das classes (DELAMARO, et al., 2016).

O grafo de causa-efeito tem o objetivo de suprir as limitações dos critérios de análise do valor limite e particionamento por equivalência, que não exploram combinações das entradas dos casos de teste. O grafo de causa-efeito identifica as causas (entradas) e efeitos (resultados) na especificação. Daí cria-se um grafo booleano ligando as causas com os seus respectivos efeitos no programa. Cada nó, tanto causa como efeito, tem um valor 0 (zero) ou 1 (um). Zero é um estado não verdadeiro e 1 é um estado verdadeiro, que existe para o sistema. Deve-se adicionar anotações ao grafo, de “AND” e “OR”, que delimitam as combinações das causas e efeitos que são possíveis e impossíveis em relação às restrições sintáticas ou do ambiente. A partir do grafo, deve-se gerar uma tabela com cada causa e o efeito esperado e assim derivar os casos de teste (DELAMARO, et al., 2016).

Teste estrutural

O teste estrutural (também conhecido como teste caixa-branca) tem como princípio utilizar o código-fonte do sistema para criar os casos de teste, ou seja, o testador utiliza a estrutura e os aspectos de implementação para a seleção dos casos de teste. De modo a simplificar o entendimento e a visualização da estrutura do programa, a maioria dos critérios de teste estrutural utilizam o Grafo do Fluxo de Controle (GFC) (MYERS, et al., 2011). O GFC se utiliza de nós para apresentar blocos disjuntos de comandos e arcos para representar os diferentes fluxos de execução. Como se trata da execução de um programa, o GFC é um grafo orientado, possuindo

um único nó de entrada e um de saída. A partir do GFC, os critérios do teste estrutural determinam diretrizes para selecionar os casos de teste, garantindo a cobertura do código por meio da execução dos diversos caminhos possíveis no grafo.

Os critérios mais conhecidos da técnica de teste estrutural são critérios baseados em complexidade, baseados em fluxo de controle e baseados em fluxo de dados. Os critérios baseados na complexidade utilizam informações sobre a complexidade do programa. Um dos critérios mais conhecidos é o critério de McCabe, que utiliza a complexidade ciclomática para fornecer uma medida quantitativa da dificuldade para conduzir esse tipo de teste.

Os critérios baseados em fluxos de controle utilizam características de controle da execução do programa, tais como comandos ou desvios para derivar os casos de teste (DELAMARO, et al., 2016). Os principais critérios dessa categoria são:

- Todos-Nós: exige que cada comando do programa testado seja executado pelo menos uma vez, ou seja, exige que a execução do programa passe, ao menos uma vez, em cada vértice do GFC;
- Todas-Arestas: exige que cada aresta (desvio de fluxo de controle) do programa seja exercitada pelo menos uma vez;
- Todos-Caminhos: exige que todos os caminhos possíveis do programa sejam executados.

Os critérios baseados em fluxos de dados concentram-se nas definições e usos de variáveis para derivar os casos de teste. Esses critérios são mais adequados para certas classes de defeitos, como os defeitos computacionais. A seguir são apresentados os principais critérios baseados em fluxos de dados (RAPPS & WEYUKER, 1982):

- Todas-Definições: exige que cada definição de uma variável seja executada pelo menos uma vez, não importa se é por c-uso ou p-uso;
- Todos-Usos: exige que toda associação entre uma definição de variável e todos seus usos (c-uso e p-uso) sejam exercitadas, por pelo menos um caminho livre de definição;
- Todos-DU-Caminhos: exige que todas as associações entre uma definição de variável e seus subsequentes usos (c-uso e p-uso) sejam exercitadas por todos os caminhos livre de definição e de laços.

Um problema relacionado ao teste estrutural é a impossibilidade de se determinar automaticamente se um caminho é executável ou não, pois não existe algoritmo que decida se um dado caminho é executável e forneça os valores que executem esse caminho (VERGILIO, et al., 1993). Sendo assim, ainda é necessária a intervenção de um testador para determinar quais caminhos são não executáveis para o programa que está sendo testado.

Teste baseado em defeitos

A técnica de teste baseada em defeitos utiliza informações sobre os tipos de erros mais comuns cometidos durante o processo de desenvolvimento de um software (DEMILLO, et al., 1978). A técnica foca nos erros comuns que um programador ou projetista poderiam cometer durante o desenvolvimento do software e nas abordagens que podem ser usadas para detectar a ocorrência de tais erros (DELAMARO, et al., 2016).

A principal técnica do critério baseado em defeitos é a análise de mutantes. A análise de mutantes descrita por (DEMILLO, et al., 1978) se baseia em duas hipóteses:

- Hipótese do programador competente: assume que programadores experientes escrevem programas corretos ou muito próximos do correto. Assumindo a validade desta hipótese, pode-se afirmar que erros são introduzidos nos programas por meio de pequenos erros que, embora não causem erros sintáticos, alteram a semântica do programa e, consequentemente, podem conduzir o programa a um comportamento incorreto.
- Efeito de acoplamento: assume que erros complexos estão relacionados a erros simples. Um conjunto de casos de teste capazes de revelar erros simples é capaz de revelar erros complexos. Então, aplica-se uma mutação de cada vez em um programa P em teste. Ou seja, cada mutante contém apenas uma transformação sintática em relação ao original.

Utilizando as ideias das hipóteses do programador competente e do efeito de acoplamento, o testador deve fornecer um programa P a ser testado e um conjunto T de casos de teste que será avaliado em relação à capacidade de detectar mutantes.

Os testes do conjunto T são executados no programa e se resultados incorretos forem obtidos, então um erro foi encontrado e o teste termina com o erro encontrado sendo reportado. Se nenhum erro foi encontrado, o programa ainda pode conter erros que o conjunto T não conseguiu revelar. A partir disso, algumas mutações são aplicadas ao programa P, dando origem aos programas mutantes P_1, P_2 … , P_n, cada um diferindo de P em apenas uma mutação (DEMILLO, et al., 1978).

2.1.4 Medidas de teste

Várias medidas que envolvem a atividade de testes estão presentes na literatura, uma vez que teste é uma das principais atividades da garantia da qualidade. Dentre algumas das mais importantes para o processo e garantia da qualidade do produto, é possível destacar as medidas de cobertura de código, *fault seeding*, escore de mutação e densidade de defeitos.

Cobertura de código é uma medida comumente utilizada como critério de adequação. Na prática, a cobertura de código é baseada na porcentagem de código-fonte do programa que foi testado pelos casos de teste elaborados, ou seja, na porcentagem de instruções que foram executadas pelos testes. Entretanto, há outras formas de medir a cobertura dos casos de teste além do código-fonte, como cobertura de ramificações e cobertura de caminhos. A cobertura de ramificações requer que todos as estruturas de controle do programa (isto é, estruturas lógicas e de repetição) sejam testadas pelo menos uma vez. Já a cobertura de caminhos requer que todos os caminhos possíveis da entrada até a saída de um programa sejam executados (ZHU, et al., 1997).

Fault seeding é uma medida utilizada para verificar a capacidade que os testes projetados têm em revelar os defeitos em um programa. Ela faz com que alguns defeitos recorrentes sejam intencionalmente inseridos no código-fonte do programa e os testes sejam executados. Com isso, é esperado que os testes sejam capazes de identificar não só todos os defeitos artificiais que foram inseridos como também defeitos verdadeiros que não foram identificados anteriormente. Uma alta confiança nos resultados pode ser obtida se os resultados dos testes no código com *fault seeding* se assemelharem aos resultados obtidos com o código-fonte original. Com isso, é obtida uma medida de confiança da capacidade dos testes em revelar defeitos.

A análise de mutação é um critério que utiliza *fault seeding* de uma maneira mais segura, inserindo apenas um defeito por mutante, o que torna mais fácil a remoção do defeito após a aplicação da técnica, mas também é mais custosa (MARCHETTO, et al., 2007).

O escore de mutação é uma medida que, assim como *fault seeding*, avalia a capacidade que os testes elaborados têm em revelar defeitos em um programa (WALSH, 1985).

Densidade de defeitos é uma medida para identificar a quantidade de defeitos presentes em relação ao tamanho do programa, normalmente medido em milhares de linhas de código (Kloc) (HATTON, 1997).

2.1.5 Processo de teste

O processo de teste integra os conceitos, estratégias, técnicas e medidas de teste em um processo controlado que apoia as atividades de teste de software. O processo de teste descreve os passos a serem executados, definindo uma maneira organizada de se planejar e executar os casos de teste, e quanto trabalho, tempo e recursos serão necessários para a realização das atividades de teste.

O processo de teste de software tem papéis que dividem as responsabilidades da equipe de testes durante a realização das atividades do processo. Os principais papéis do processo de teste são (RIOS & MOREIRA, 2013):

- Testador de Software: é o responsável pela execução dos casos de teste e dos *scripts* de teste;
- Analista de Teste de Software: é o responsável pela modelagem e elaboração dos casos de teste e *scripts* de teste;
- Arquiteto de Teste de Software: é o responsável pelas atividades relacionadas com a infraestrutura do teste, escolhendo as ferramentas e preparando a equipe para executar os testes no ambiente;
- Líder da Equipe de Teste: é o responsável pela liderança de um projeto de teste, podendo ser relacionado a um projeto de desenvolvimento ou manutenção de um sistema.

De acordo com Rios e Moreira (2013), o ciclo de vida do processo de teste, mostrado na Figura 1, é composto por 6 atividades, sendo que quatro delas ocorrem

de forma sequencial e as outras duas ocorrem de forma paralela em relação às demais. O Plano de Teste é o principal artefato que guia todo o processo de teste, definindo os níveis, técnicas e tipos de testes que serão executados. O Plano de Teste descreve o escopo, abordagem e os recursos, além da agenda das atividades de teste (IEEE Computer Society, 1998).

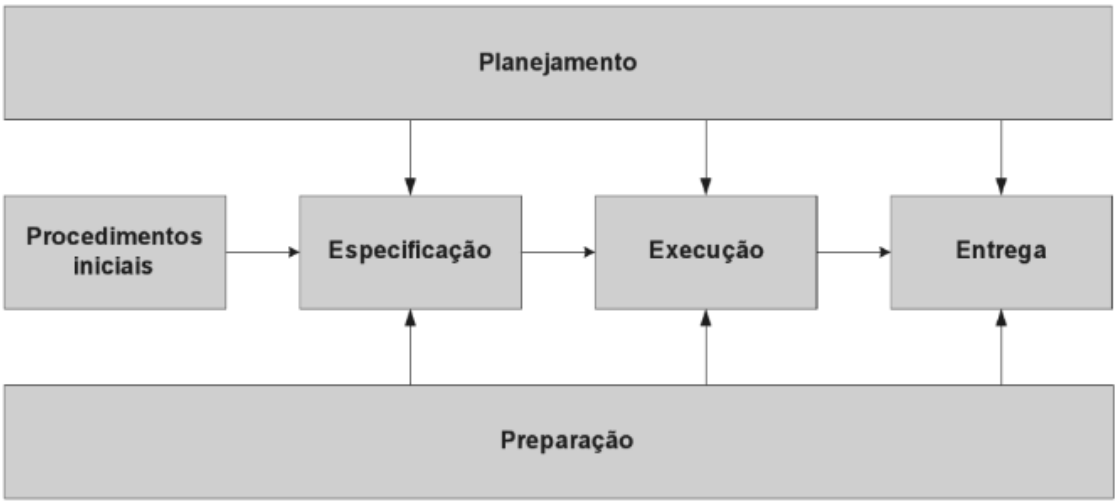


Figura 1. Etapas do processo de teste por Rios e Moreira (2013).

Na etapa de **Procedimentos Iniciais** é estabelecido um acordo com as partes envolvidas com o desenvolvimento do sistema sobre o objetivo do processo de testes. Também são definidos a equipe responsável pela execução do processo, as responsabilidades de cada, o plano preliminar de trabalho, avaliação dos riscos, níveis de serviços e outros itens considerados relevantes para o sucesso do projeto.

Planejamento dos Testes é a atividade focada na elaboração da estratégia de teste e o plano de teste, que serão utilizados a fim de minimizar os principais riscos do negócio e fornecer os caminhos para as próximas atividades. No plano de testes, também devem ser definidos os critérios e as técnicas que serão utilizados para testar o software e, assim, obter uma previsão do esforço, tempo e recursos que serão necessários para realizar as atividades planejadas.

Preparação é a atividade com foco na preparação do ambiente de testes, o que inclui hardwares, softwares, pessoal, ferramentas e documentos. Essa atividade é importante para que as demais atividades de teste sejam executadas facilmente e de maneira correta pela equipe de teste.

A **Especificação** ou Projeto dos Casos de Teste é a atividade em que os casos de teste, *scripts* e cenários de teste para testar o software serão definidos. A atividade de especificação dos testes deve seguir os níveis e critérios que foram definidos no plano de teste. Além disso, nesta atividade devem ser realizadas inspeções na documentação do sistema.

A **Execução dos Testes** é a atividade na qual os casos de teste, *scripts* e cenários de teste que foram especificados são executados no software a fim de encontrar erros. Os resultados obtidos com a execução dos testes devem ser documentados. Inspeções nos documentos do sistema também podem ser realizadas nesta atividade.

Na **Entrega** é feita a conclusão do processo de teste com a entrega do sistema para o ambiente de produção. Toda a documentação de testes elaborada durante o processo deve ser recolhida para a elaboração de um relatório gerencial com as conformidades e não conformidades encontradas.

Os testes de software são importantes para todo desenvolvimento de software. Alguns processos, como o IBM Rational Unified Process (RUP) (KRUCHTEN, 2004) trata testes como uma das principais disciplinas do processo, sendo realizada desde o início até o fim do desenvolvimento do software. A disciplina de testes, seguindo o processo descrito pelo RUP, contém os seguintes papéis:

- Gerente de teste: planejar, gerenciar e resolver problemas que atrapalhem os esforços de teste;
- Analista de teste: identificar e definir os requisitos de teste, monitorar a cobertura dos testes elaborados e avaliar a qualidade geral após a aplicação dos testes;
- *Designer* de teste: definir a abordagem de teste, as técnicas apropriadas, ferramentas e guias para a realização dos testes; e
- Testador: conduzir os testes e registrar os resultados de teste.

O processo de teste do RUP contém 6 atividades principais. A Figura 2 apresenta o fluxo das atividades da disciplina de testes do RUP. A atividade **Definir missão da avaliação** tem como objetivo identificar o foco e definir os objetivos dos testes.

A atividade **Verificar abordagem de testes** tem como objetivo demonstrar que as técnicas empregadas facilitarão o esforço de teste, produzindo resultados precisos.

Verificar a estabilidade da *build* é verificar se a *build* (versão compilada do sistema em uma iteração) está estável o suficiente para que as atividades de teste sejam realizadas.

Arquivar missão aceitável é a atividade de fornecer aos *stakeholders* resultados da avaliação dos testes.

Testar e avaliar envolve implementar os testes, avaliar a build com a execução dos testes criados e gerar o relatório de incidentes encontrados.

Melhorar ativos de teste tem como objetivo manter e melhorar os ativos de teste que podem ser usados para os testes da iteração atual e das demais iterações que virão ocorrer.

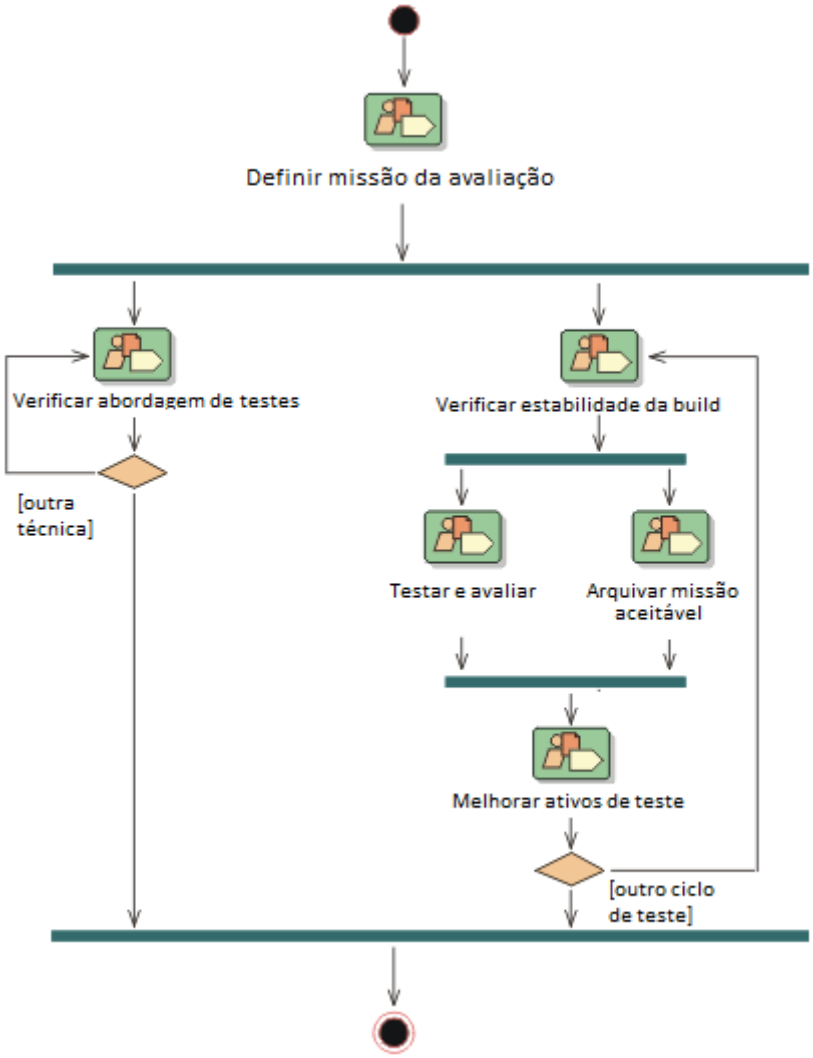


Figura 2. Fluxo de atividades da disciplina de testes do RUP adaptado de (RATIONAL SOFTWARE CORP, 2021).

2.2 Jogos educacionais

Com o aumento do uso das tecnologias digitais, diversas mudanças ocorreram nas relações e na maneira de pensar (SÁNCHEZ, 2014), o que influenciou no modo como estudantes estudam e aprendem. Com o rápido desenvolvimento da tecnologia e a penetração social dos jogos, a demanda das pessoas por *"edutainment"*, um educação acompanhada de entretenimento está aumentando (ZENG, et al., 2020). A principal razão disso são as mudanças que tecnologias digitais trouxeram para a vida cotidiana. Os estudantes não são iguais aos estudantes de antes (PRENKSY, 2012), uma vez que mudaram radicalmente nos últimos anos, sendo mais acostumados a consumir mídias digitais (TAYLOR & PARSONS, 2011) e a experimentar as

tecnologias digitais como espaços sociais contínuos, ativos e participativos (WRIGHT, et al., 2014).

Uma das formas de acompanhar as mudanças ocorridas foi a introdução de jogos na educação. Os jogos estão sendo aplicados como estratégia de ensino e aprendizagem dirigidos a um público-alvo caracterizado como geração *gamer*, já nascem e crescem cercados de tecnologia. Os jogos educacionais são jogos que não se destinam apenas ao entretenimento ou passatempo de seus jogadores, mas são desenvolvidos, conforme o próprio nome sugere, com propósitos educacionais. Eles são jogos destinados para ensino de algum conteúdo, para expandir os conhecimentos de seus jogadores, simular atividades práticas em um ambiente não real, dentre outros. A diferença trazida pelo uso de jogos educacionais é que eles facilitam a aprendizagem de conceitos e ideias ao mesmo tempo que motivam e proporcionam prazer aos jogadores. Isso permite que os jogadores obtenham conhecimentos sobre o conteúdo combinados com a diversão proporcionada pelo jogo (BAKKER, et al., 2015).

Os jogos educacionais podem ser utilizados com alguns objetivos, como reforçar conhecimentos já adquiridos, facilitar a aprendizagem de algum conteúdo, desenvolver habilidades dos estudantes, entre outros (AMARAL, et al., 2013). Os jogos educacionais fornecem motivações para que os jogadores tenham um aprendizado por experiência, onde eles podem enfrentar novos desafios e estimular suas próprias habilidades (HSU, et al., 2013).

Os jogos podem ser classificados como não digitais ou digitais. Os digitais são caracterizados por serem representados em forma de elementos gráficos interativos jogados por meio de um dispositivo virtual, como um celular ou computador. Já os jogos não-digitais são caracterizados por serem representados ou envolverem objetos físicos e palpáveis, podendo ser jogados com cartas, tabuleiros, objetos esportivos, dentre outros (LUCCHESE & RIBEIRO, 2009).

A experiência a ser descrita neste trabalho se concentra nos jogos educacionais voltados para o ensino de testes de software em disciplinas de Ciência da Computação e áreas afins. Há várias propostas de jogos digitais e não digitais para ensino de testes de software. A realização do trabalho se concentrará na criação e avaliação de um jogo educacional para ensino do processo de testes de software.

2.2.1 Benefícios dos jogos educacionais

Uma das principais vantagens de ser apenas um jogo, e não uma aplicação real, é a minimização do fracasso dos jogadores, o que o torna mais adequado para o ensino de estudantes do que treinamentos práticos. A realização de treinamentos práticos tradicionais tem custos elevados, pode envolver riscos reais e tem demonstrado ter efeitos de aprendizagem e motivação inferiores do que treinamentos com jogos educacionais (BHAGAT, et al., 2016); (ROSENTHAL, et al., 2011). Simulando um cenário real, por exemplo, é possível permitir que os jogadores se arrisquem e experimentem ações difíceis de testar em um cenário real, onde o custo do fracasso seria muito maior (MATTAR, 2010); (MEDEIROS & SCHIMIGUEL, 2012).

Zeng, Parks & Shang (2020) citam quatro principais benefícios do uso de jogos educacionais:

- Estimular a motivação para aprender – como jogos têm um potencial de estimular a motivação e manter a atenção dos jogadores durante horas, é possível usar de mecânicas e design de jogos para ensinar;
- Melhorar resultados de aprendizagem - jogos educacionais não apenas têm efeitos motivacionais, mas também podem melhorar significativamente o desempenho acadêmico (HWANG, et al., 2013);
- Criar ambientes de aprendizagem – jogos educacionais podem criar ambientes de aprendizagem altamente interativos, com riscos controlados e que simulam situações reais, o que permite a tentativa e erro dos jogadores e a experiência de diversas situações;
- Promover a transformação de métodos de aprendizagem – jogos educacionais podem criar ambientes livre de erros a um baixo custo, o que permite que os jogadores possam ter experiências simuladas de situações do mundo real em ambiente controlados.

A utilização de jogos educacionais tem mostrado impacto positivo no processo de ensino aprendizagem em diversas áreas. A alguns dos principais benefícios que eles podem trazer para o processo de ensino-aprendizagem (SAVI, 2011, pp. 54-55):

- Efeito motivador - com desafios motivadores e bem planejados, é possível capturar a atenção e entusiasmo dos estudantes e direcioná-los para o aprendizado de um determinado assunto;

- Facilitador do aprendizado - por meio de recursos gráficos, os jogos educacionais conseguem facilitar o entendimento do conteúdo a ser ensinado. Além disso, por meio da simulação de cenários reais, os jogos conseguem melhorar o pensamento estratégico dos estudantes para resolução de problemas;
- Aprendizado por descoberta - por conta do *feedback* imediato, os estudantes podem aprender por meio da tentativa e erro. Juntando a isso um ambiente livre de riscos reais, os jogadores se sentem mais livres para tomar novas decisões, estimulando a experimentação de novos cenários;
- Experiência de novas identidades - pelo fato de que em muitos jogos o jogador tem que assumir a identidade de um personagem do jogo, é possível obter o aprendizado com as competências e conhecimentos associados ao personagem assumido;
- Socialização - jogos educacionais podem servir como agentes mediadores de socialização, permitindo que jogadores troquem informações, compartilhem problemas e se ajudem para resolver um problema comum;
- Coordenação motora - há vários jogos que promovem o desenvolvimento de capacidades motoras e habilidades espaciais;
- Comportamento expert - ao muito se jogar um jogo com desafios educacionais, o jogador ganha um sentimento de expertise sobre o tema abordado por meio da prática constante.

2.3 Considerações sobre o capítulo

Neste capítulo de revisão bibliográfica foram apresentados os conceitos relacionados a jogos educacionais, seus benefícios e utilização. Também foram apresentados os principais fundamentos, técnicas, níveis e medidas de testes, e o processo de teste de software. Tais tópicos de teste são abordados por vários jogos que tentam de uma forma lúdica facilitar o aprendizado de tais tópicos por seus jogadores, que em muitos casos são estudantes de graduação. No próximo capítulo são apresentados os jogos educacionais para ensino dos tópicos de teste de software que foram encontrados na revisão da literatura feita pelo autor deste trabalho.

CAPÍTULO 3 - TRABALHOS RELACIONADOS

> “Ao brincar, a criança assume papéis e aceita as regras próprias da brincadeira, executando, imaginariamente, tarefas para as quais ainda não está apta ou não sente como agradáveis na realidade.”
>
> Lev Vygotsky

Nesta seção são destacados os principais trabalhos relacionados ao estudo. Há várias propostas de jogos educacionais com a intenção de auxiliar no ensino-aprendizagem de testes de software. Alguns dos jogos que podem ser mencionados como trabalhos relacionados são: U-TEST; JETS; Jogo das 7 falhas; iTest learning; TestEG; Code hunt; iLearnTest; JoVeTest; BlackBox; Code defenders; Bug Hide-seek; Testing game; GreaTest; e IslandTest. Para uma melhor organização e entendimento, esta seção está dividida em subseções para cada jogo educacional identificado seguindo a ordem em que foram apresentados.

3.1 U-TEST

O U-TEST é um jogo digital de simulação para apoio ao ensino de testes de software, focando especificamente em testes de unidade (SILVA & THIRY, 2010). No jogo, o jogador tem o papel de um testador responsável por escrever testes de unidade para testar funções de um sistema hipotético. Para criar os testes de unidade, o jogador deve aplicar técnicas para a seleção de dados de entrada dos testes, como particionamento em classe de equivalência e análise do valor limite.

Inicialmente, o jogador tem o papel de um programador *freelancer*, podendo estar associado a um projeto ou não. Enquanto não tem um projeto, o jogo mostra alguns projetos que o jogador pode escolher para trabalhar, apresentando a descrição do projeto, o nível de dificuldade e a remuneração ao finalizar os testes no projeto. Após a escolha de um projeto, diversos desafios relacionados à verificação das

unidades de código já desenvolvidas são apresentados ao jogador. A Figura 3 apresenta uma das telas com o desafio sobre classes de equivalência, em que o jogador deve selecionar os valores que delimitam as classes de teste. Ao fim de cada desafio, um guru de testes questiona o jogador sobre suas respostas ao desafio. Mesmo se o jogador concluir corretamente o desafio, se ele errar as respostas ao guru, ele pode receber alguma desqualificação. Se o jogador apresentar desempenho abaixo do esperado, o guru surge para o jogador, mas dessa vez, apresentando dicas.

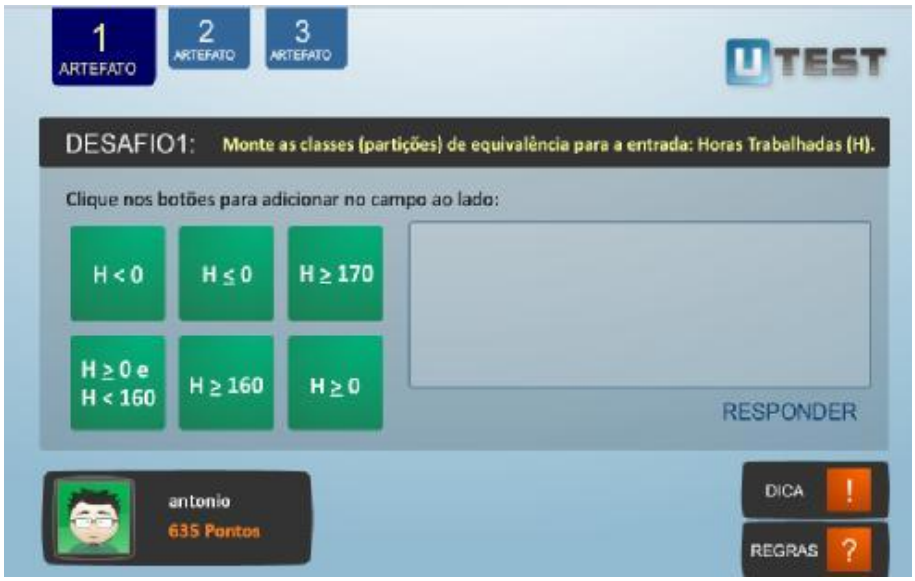


Figura 3. Tela de desafio sobre classes de equivalência do U-TEST por Silva e Thiry (2010).

No final do projeto, o jogo apresenta o desempenho do jogador, informando, inclusive, algumas críticas para problemas identificados e parabenizando o jogador por concluir o projeto. Caso o jogador tenha atingido uma das três melhores pontuações no jogo, o jogador recebe uma medalha e um troféu. Após isso, também é apresentado o hall da fama onde são disponíveis as pontuações dos demais jogadores e a classificação do jogador atual.

Os objetivos do jogo são fazer com que os jogadores reconheçam e entendam os principais conceitos de testes de software e que possam entender e aplicar as técnicas de particionamento em classe de equivalência e análise do valor limite.

Foram realizados três experimentos com o U-TEST com um total de 30 estudantes de graduação. O primeiro experimento foi feito com um grupo experimental, que teve experiência com o jogo e um de controle, que teve experiência com um jogo placebo para ensino de teste. O segundo experimento foi feito com um

grupo experimental, que teve experiência com o jogo e um de controle, que resolveu exercícios em sala como forma de aprender testes. O terceiro experimento do jogo foi realizada apenas com a aplicação do jogo. Para cada experimento foi feito a aplicação de pré e pós-teste e a aplicação de um questionário de avaliação do jogo. Enquanto o jogo, de modo geral, foi bem avaliado pelos estudantes, seus resultados só mostraram ter resultado significativo no primeiro experimento, quando comparado ao jogo placebo. Por fim, um dos principais pontos de insatisfação dos estudantes em relação ao jogo foi em relação às informações que são dadas durante o jogo. Apesar disso, os autores afirmam que já existe versões onde esse problema foi corrigido.

3.2 JETS

O JETS (Jogo da Equipe de Teste de Software) é um jogo sério 3D multiplayer que visa apoiar o ensino-aprendizagem de testes de software em cursos de graduação em Ciência da Computação e áreas afins (SILVA & MÜLLER, 2012). O jogo é classificado como do gênero simulador e simula as interações da equipe de testes de uma empresa de desenvolvimento de softwares.

O jogo é constituído de quatro fases e em cada fase o jogador assume um papel diferente dentro de uma equipe de testes da empresa. Na primeira fase, o jogador ingressa no jogo com o papel de Testador de Software e deve executar casos de teste prontos. A Figura 4 apresenta a tela no cenário de testador do jogo. Na segunda fase, o jogador assume o papel de Analista de Testes e deve elaborar os casos de teste a partir de interfaces e métodos determinados. Na terceira fase, o estudante assume o cargo de Arquiteto de Testes e deve definir o ambiente de testes e escolher as ferramentas de teste apropriadas. Na quarta fase do jogo, o estudante assume o cargo de Líder da Equipe de Teste e será responsável pela alocação de recursos, acompanhamento das atividades de teste, verificando prazos e custos do processo de teste de software de um projeto.

Figura 4. Cenário da fase de testador de software do JETS por Silva e Müller (2012).

Os desafios de cada fase, bem como as respectivas pontuações, poderão ser editados pelo professor. O JETS é um jogo de apoio ao ensino-aprendizagem, logo, o jogo não exclui a necessidade de um instrutor. O objetivo do jogo é aprimorar o conhecimento da estratégia de testes de software e fornecer uma visão geral de como as atividades de teste podem ser realizadas.

O jogo foi aplicado com 15 estudantes do curso de Ciência da Computação. O experimento do jogo foi feito com dois grupos um grupo experimental, que teria a experiência com o jogo e um grupo de controle, que resolveria exercícios em sala. Para o experimento foram aplicados pré e pós-teste e um questionário de percepção do jogo. Os resultados de pré e pós-teste do experimento não apresentam resultados de melhora nos conhecimentos dos estudantes, principalmente porque, segundo os autores, muitos estudantes não realizaram o pós-teste. Já o questionário de percepção do jogo apresenta resultados positivos dos estudantes, mesmo considerando que poucos estudantes tiveram experiência com o jogo.

3.3 Jogos das 7 falhas

O Jogo das 7 Falhas é um jogo digital *single-player* que consiste em descobrir falhas em funcionalidades de um software (DINIZ & DAZZI, 2011). No jogo, o jogador assume o papel de um testador em uma equipe de teste de software de uma empresa fictícia chamada Diniz Quality Assurance. O jogador deve descobrir as falhas

existentes em cada funcionalidade testada, correlacionando-as com uma classe de equivalência ou um valor-limite.

O jogo possui dois níveis de complexidade: baixa e média. Em cada nível existe uma funcionalidade com 7 falhas a serem descobertas. No nível 1, o jogador deve encontrar as 7 falhas em no máximo 25 minutos. Já no nível 2, o jogador deve encontrar as 7 falhas em no máximo 40 minutos. O jogador só pode jogar no nível 2 se tiver completado o nível 1. Caso o tempo de um nível termine antes do jogador ter encontrado todas as falhas da funcionalidade, o jogador é eliminado. Caso descubra as falhas das duas funcionalidades a tempo, ou seja, conclua os níveis 1 e 2 no tempo estimado, o jogador é vencedor.

Antes de iniciar um nível, sete falhas são sorteadas aleatoriamente dentre 33 possíveis para a funcionalidade que está sendo testada no nível. Dessa forma, o jogador tem mais desafios e fica mais motivado para jogar o jogo novamente, pois cada jogada pode haver desafios diferentes. Para identificar as falhas, o jogador deve criar casos de testes elaborados por meio das técnicas de classe de equivalência e análise de valor-limite. Se o caso de teste originar uma falha, será exibida uma mensagem de que uma falha foi descoberta e que é necessário informar a classe de equivalência ou o valor-limite que a originou, como apresentado na Figura 5. Caso um caso de teste não origine uma falha, uma mensagem correspondente ao resultado esperado e um feedback informando a classe de equivalência ou valor-limite o caso de teste corresponde.

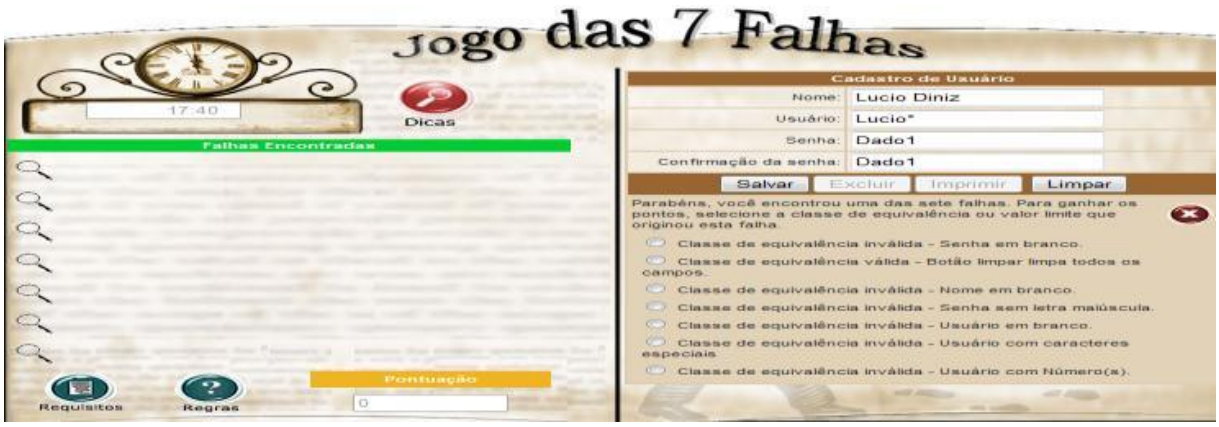


Figura 5. Tela do Jogo das 7 Falhas após o jogador encontrar uma falha por Diniz e Dazzi (2011).

Ao gerar um caso de teste que origina uma falha e escolher corretamente a classe de equivalência ou valor-limite que originou a falha, o jogador ganha 10 pontos. Ao escolher a classe de equivalência ou valor-limite errada, o jogador perde 10 pontos. Além disso, a pontuação do jogador também leva em conta o tempo de sobra para cada nível. Um ponto é incrementado na pontuação do jogador a cada intervalo de 10 segundos que sobrar para cada nível.

O Jogo das 7 Falhas foi aplicado com 21 estudantes de graduação em Ciência da Computação. O experimento foi feito com dois grupos: um experimental, que teve experiência com o jogo e um grupo de controle, cujos participantes responderam exercícios sobre teste caixa-branca. No experimento foram aplicados pré e pós-teste para avaliar o ganho de conhecimento dos estudantes após o tratamento de cada grupo. Com a aplicação de testes de hipótese nos resultados de pré e pós-teste, foi verificado que houve um ganho significativo dos estudantes que tiveram experiência com o jogo em comparação com os que não tiveram. Também foi aplicado um questionário para avaliar a percepção dos jogadores sobre o jogo. A avaliação da percepção do jogo foi bem positiva, embora não seja apresentada mais nenhuma informação sobre os pontos positivos do jogo.

3.4 *iTest Learning*

O *iTest Learning* é um jogo de simulação para a plataforma Web com o objetivo de apoiar o ensino de testes de software, focando na fase de planejamento dos testes (FARIAS, et al., 2012). O jogo provê um ambiente simulado para realizar o planejamento de testes de software de um projeto hipotético.

O jogo possui três níveis de dificuldade: fácil, médio e difícil. No início do jogo, diversos projetos são apresentados ao jogador para que ele escolha um. Dependendo do nível escolhido, um conjunto diferente de projetos é apresentado ao jogador. Após o jogador escolher um projeto, o jogo apresenta a descrição do projeto escolhido. A partir da descrição do projeto, o jogador deve realizar as 6 fases de planejamento das atividades de teste. Na Fase 1, ilustrada na Figura 6, o jogador deve escolher os itens do projeto a serem testados. Na fase 2, o jogador deve definir os tipos de teste que serão realizados, como teste de Interface, de Usabilidade, Segurança, Estresse etc. Na fase 3 o jogador deve definir por quais níveis de testes o projeto passará, como

Unidade, Integração, de Sistema etc. Na fase 4 o jogador deve definir os critérios de aceitação para que um teste executado seja aprovado ou não. Na fase 5 o jogador deve escolher as ferramentas que serão utilizadas no processo de testes. E na fase 6 o jogador deve indicar quais artefatos devem ser gerados ao longo do processo de testes, como o Plano de Testes, Especificação de Casos de Testes, Relatório de Testes etc. O jogador também pode tirar dúvidas em relação aos conceitos abordados a qualquer momento por meio da funcionalidade de ajuda.

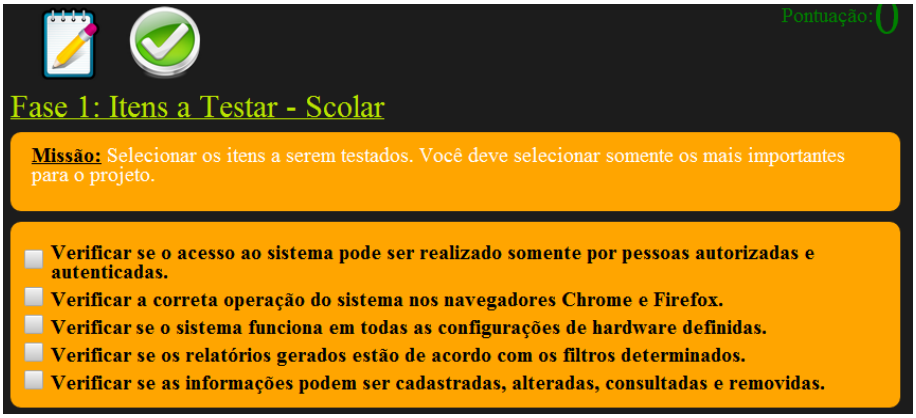


Figura 6. Descrição do projeto escolhido pelo jogador no iTest Learning por Farias et al. (2012).

Cada projeto do jogo tem configurações diferentes que exigem abordagens diferentes para os testes. A cada escolha correta tomada no planejamento dos testes para as configurações do projeto, o jogador ganha pontuação. Ao finalizar o planejamento e submeter suas escolhas, o jogo dá um feedback para o jogador, mostrando seus erros e/ou acertos. No feedback também é exibida uma proposta de como poderia ser o planejamento dos testes para o projeto, uma vez que não há como definir uma resposta absoluta. Além disso, também é possível o jogador visualizar o *ranking* do jogo e verificar sua classificação em relação aos outros jogadores.

O jogo não aborda todos os itens de planejamento de testes. Os itens de planejamento cobertos no jogo são: escopo do projeto, itens de teste, tipos de teste, níveis de teste, ferramentas de teste e artefatos de testes.

Bezerra et al. (2014) apresenta os resultados de aplicação de uma versão atualizada do iTest Learning. O experimento com o jogo contou com 39 estudantes dos cursos de Engenharia de Software, Ciências da Computação e Sistemas de Informação. Para o experimento, foi aplicado o questionário proposto por Savi,

Wangenheim e Borgatto (2011). Em relação a avaliação com o questionário, o jogo foi bem avaliado em relação ao aprendizado percebido pelos estudantes. Entretanto, os aspectos de diversão, desafio e imersão ainda indicam necessidades de melhorias, principalmente em relação a interface do jogo.

3.5 TestEG

O TestEG (*Test Educational Game*) é um jogo computacional *single-player* para apoiar o ensino-aprendizagem de assuntos que envolvem testes de software (OLIVEIRA & COSTA, 2013). O jogador assume o papel de Gerente de Testes e deve auxiliar uma equipe de testes para realizar as atividades de testes, tirando dúvidas, fornecendo treinamento aos testadores e respondendo perguntas sobre teste de software.

No início do jogo, o jogador recebe um orçamento inicial e deve selecionar três testadores para compor a sua equipe. A Figura 7 apresenta a tela de seleção do personagem do jogador. Ao longo do jogo, o jogador deve auxiliar sua equipe, solucionando suas dúvidas e fornecendo informações necessárias para que eles realizem suas atividades. O jogador também pode fornecer treinamento para os testadores, verificar o desempenho deles e ler conteúdo sobre testes de software. Ao treinar um funcionário, o nível de habilidade dele sofre um aumento de acordo com o treinamento executado.

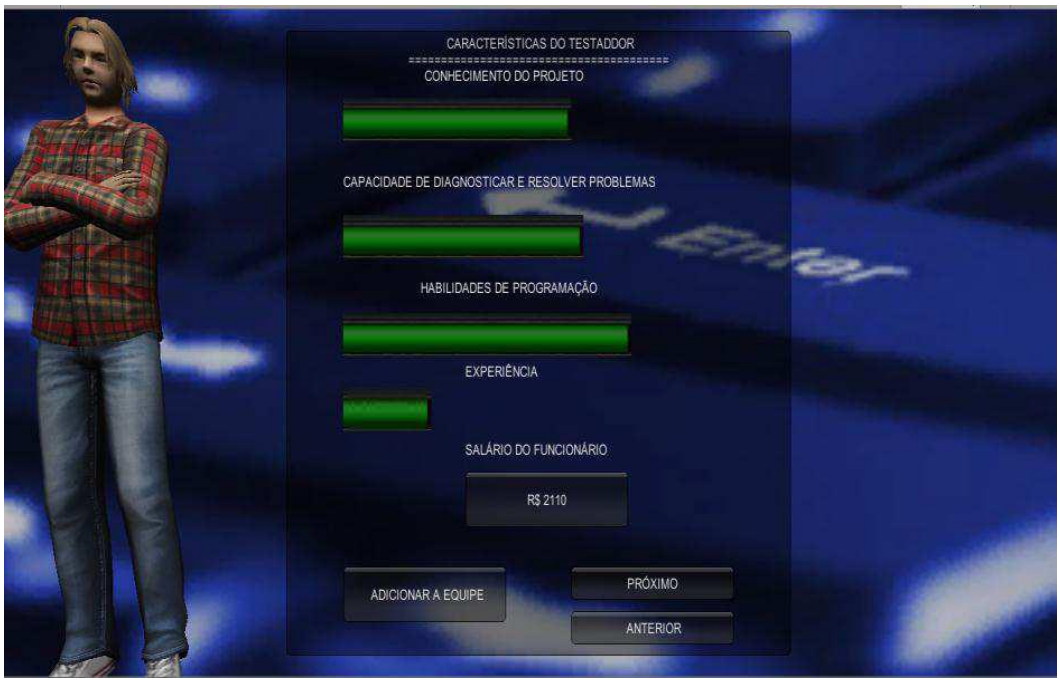


Figura 7. Tela de escolha de personagem do jogo TestEG por Oliveira e Costa (2013).

A cada dois minutos (tempo equivalente a um mês no jogo), um valor relativo aos gastos com os funcionários é retirado do orçamento do jogador. Além disso, para testar as habilidades do jogador em gerenciar uma equipe de testes, o jogador deve responder 10 perguntas em um período de 10 minutos, sem que o orçamento acabe. O nível de dificuldade das perguntas é baseado no nível de habilidade dos funcionários contratados. Quanto mais habilidosos forem os funcionários, mais fáceis são as perguntas. O jogador perde o jogo caso seu orçamento acabe antes de ter respondido as 10 perguntas.

O jogo é dividido em dois papéis: o do usuário, que são os jogadores, e o administrador. Antes de começar o jogo, o administrador deve criar as perguntas que serão feitas aos usuários, informando a descrição da pergunta, os itens de resposta, a resposta correta e a pontuação da pergunta. Cada pergunta tem pontuação que corresponde ao seu nível de dificuldade: perguntas no nível fácil variam de 0 a 100; no nível médio de 101 a 300; e no nível difícil de 301 a 500. A cada resposta correta, o jogador ganha os pontos correspondentes à pergunta. Para cada resposta errada, 10% da pontuação da pergunta são subtraídos da pontuação do jogador e um minuto são acrescentados ao tempo do jogador. Além disso, a cada resposta há uma atualização nas habilidades do funcionário que fez sua solicitação: se o jogador

responde corretamente, o funcionário ganha habilidade; caso contrário, o funcionário perde habilidade.

Ao final, é apresentada a pontuação do jogador, que é baseada tanto nas respostas corretas quanto nas habilidades dos funcionários ao final do jogo. Além disso, é apresentado o *ranking* dos três jogadores com maior pontuação no jogo.

O experimento com o jogo foi feito com a aplicação do modelo de avaliação proposto por Savi, Wangenheim e Borgatto (2011). O jogo foi aplicado com 127 participantes, dos quais 52 responderam ao questionário de avaliação. A avaliação com o jogo foi positiva em todos os aspectos avaliados, apesar de algumas melhorias na jogabilidade e desafios terem sido levantadas pelos participantes.

3.6 *Code Hunt*

Code Hunt é um jogo sério que não só cria internamente os valores para guiar a geração de testes, mas também oferece experiências divertidas ao usuário onde a geração de testes baseada em pesquisa é simulada manualmente (TILLMANN, et al., 2014). No jogo, o jogador não cria casos de teste, mas deve criar um código que passe em casos de teste pré-definidos para a atividade que ele está realizando.

O jogo é estruturado em setores que contém uma série de níveis. Os setores são organizados em tópicos sequenciais, como “operações aritméticas”, “*loops*”, “condicionais”, “*string*” e “*cyphers*”. Em cada nível, o jogador deve implementar um algoritmo em um editor de texto apresentado em uma janela do jogo. Entretanto, o jogador não tem a descrição do algoritmo a ser implementado, apenas os casos de teste que foram elaborados para testar o algoritmo. O algoritmo a ser implementado tem parâmetros de entrada e um tipo de valor de retorno especificado pelo nível. De início, o jogador não tem nenhuma informação sobre se o que ele implementou está certo ou não. Para saber se ele atingiu o algoritmo correto, ele deve enviar seu código para avaliação.

Durante a avaliação, o código é compilado e comparado com o código correto. O resultado da avaliação pode ser um erro de compilação ou discrepâncias e concordâncias com o código correto, assim como apresentado na Figura 8. Se houver discrepâncias, alguns casos de teste são apresentados para o jogador para ajudá-lo

a implementar o algoritmo correto. Se o código compilar e houver apenas concordâncias com o código correto, o jogador ganha o nível.

Figura 8. Página principal do Code Hunt mostrando o resultado dos testes por Tillmann et al. (2014).

Ao completar com sucesso um nível, o jogo atribui um número inteiro para classificar o código do jogador. As classificações são:

- 1 indica que o código é muito mais longo do que os outros códigos enviados;
- 2 significa que o código está na média; e,
- 3 significa que o código é significativamente menor do que os demais.

As classificações visam estimular que os jogadores implementem soluções simples e corretas.

Ao final, o jogador pode visualizar seu progresso final no jogo, baseado na classificação de seus algoritmos em cada nível, e é possível ver os 15 jogadores com maior pontuação no jogo.

Não foram encontrados resultados em relação ao aprendizado com o jogo, embora Bishop et al. apresentam alguns benefícios que o jogo pode trazer ao ensino de testes, como o instrutor ter acesso às versões de código de estudante durante suas tentativas, o que pode facilitar o entendimento dos problemas de aprendizado do estudante por parte do instrutor. O jogo também promove a junção de conteúdos de

programação e testes de software. E o jogo provê dicas, que podem ajudar os jogadores a melhorarem suas habilidades de programação.

3.7 *iLearnTest*

O *iLearnTest* é um jogo online para facilitar a aprendizagem e o treino dos conhecimentos na área de testes de software, a fim de captar a atenção, motivar e incentivar o envolvimento dos estudantes ao jogar o jogo (RIBEIRO & PAIVA, 2014). O *iLearnTest* não foi feito para substituir o ensino, mas para ser uma opção ao ensino de testes de software e como uma ferramenta para aumentar o interesse dos estudantes durante o estudo de assuntos do tema.

O jogo foi feito com o objetivo de ajudar os estudantes na preparação para o exame de certificação no nível *Foundation* do ISTQB (*International Software Testing Qualification Board*). A qualificação no nível *Foundation* destina-se a qualquer indivíduo envolvido nas atividades de testes de software, independentemente da função desempenhada. O corpo de conhecimento para este nível de certificação é estruturado em seis capítulos: Fundamentos de testes; Testes por meio do Ciclo de Vida de software; Técnicas Estáticas; Técnicas de Concepção de testes; Gestão de testes; e Ferramentas de Suporte aos testes.

Como o objetivo é abordar os conteúdos da certificação, na tela inicial, o jogo é dividido em seis plataformas, cada uma abordando o conteúdo de um dos capítulos da certificação. Ao posicionar o personagem do jogador em uma plataforma, o jogo apresenta o título do capítulo e a pontuação obtida pelo utilizador até ao momento naquele capítulo em relação ao total de pontos que ele pode obter, conforme representa a Figura 9.

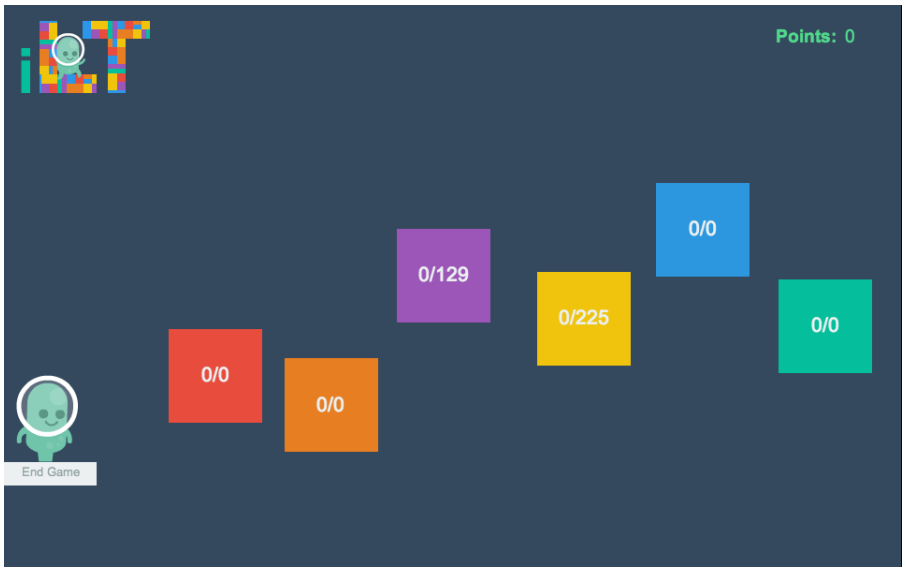


Figura 9. Tela de escolha de seção do iLearnTest por Ribeiro e Paiva (2014).

Após selecionar uma seção, são apresentados os objetivos de aprendizagem do capítulo e depois é exibido o conteúdo teórico importante para os desafios daquele capítulo. Cada capítulo no jogo apresenta desafios diferentes que avaliam os conteúdos relacionados ao capítulo selecionado.

Apesar do conteúdo do jogo ser bem definido pela certificação, uma vez que o corpo de conhecimento a ser tratado é bastante extenso, foram selecionados apenas os conteúdos do capítulo três (Técnicas Estáticas) e do capítulo quatro (Técnicas de concepção de testes). O desenvolvimento do jogo focado nestes capítulos é uma prova de conceito para que depois se possa estender e cobrir os demais capítulos do corpo de conhecimento do ISTQB.

O experimento com o iLearnTest foi feito com 12 estudantes divididos em três grupos. Os participantes do grupo A tinham que aprender sobre o conteúdo de testes apenas com o material do ISTQB. Os participantes dos grupos B e C tinham que aprender o conteúdo apenas com o iLearnTest. Foram aplicados pré e pós-testes com questões baseadas no nível *Foundation* do ISTQB para os participantes de todos os grupos. Inicialmente foi feita uma comparação com os resultados de pré e pós-teste dos participantes dos grupos A e B. Os resultados mostraram que houve apenas uma leve melhora nos resultados de pós-teste na comparação dos grupos. Após melhorias no jogo, houve a aplicação do jogo com o grupo C que apresentou uma melhora significativa no pós-teste, o que indica que o jogo é eficaz para o ensino do nível

Foundation do ISTQB. Não foram encontradas informações sobre uma nova versão do jogo na literatura.

3.8 JoVeTest

O JoVeTest, apresentado na Figura 10, é um jogo online que segue a mesma dinâmica do popular jogo da velha (BARBOSA, et al., 2016). O jogo pode ser jogado por dois jogadores, cada um possuindo um símbolo (X ou O). Alternadamente, os jogadores devem posicionar seu símbolo com o objetivo de fazer uma combinação do seu símbolo em uma linha, coluna ou diagonal. A diferença do JoVeTest para o jogo da velha convencional é que, para ter o direito de posicionar seu símbolo, o jogador deve responder corretamente a uma pergunta múltipla escolha sobre teste de software.

Figura 10. Tela de jogo do JoVeTest por Barbosa, Neves e Neto (2016).

Ao iniciar o jogo, os jogadores se deparam com a tela de configurações iniciais, onde devem informar um nome de jogador para cada símbolo. Além disso, os jogadores podem configurar os temas das questões que eles querem responder. Após isso, o jogo informa de qual jogador é a vez de jogar.

O jogador que tem a vez deve então selecionar uma posição do tabuleiro para colocar seu símbolo. Ao selecionar uma posição, uma questão dos temas selecionados é sorteada e apresentada ao jogador para que ele responda. Se ele

responder corretamente à questão, ele pode posicionar seu símbolo no local selecionado. Se ele selecionar uma opção incorreta, uma mensagem de resposta incorreta é apresentada, a alternativa correta é apresentada e o símbolo do jogador não é posicionado no tabuleiro.

No início, as questões que foram adicionadas no JoVeTest foram retiradas das provas do CTFL (*Certified Tester Foundation Level*) realizadas pelo ISTQB (*International Software Testing Qualifications Board*) nos anos de 2011 a 2015. Entretanto, novas questões podem ser adicionadas na ferramenta de forma simples, enviando um arquivo de texto para o banco de questões da ferramenta.

O JoVeTest foi testado com 20 estudantes de graduação, divididos em dois grupos, grupo A e B. Os participantes dos dois grupos tiveram experiência com o jogo, a diferença é que o conteúdo abordado no jogo foi diferente entre os grupos, sendo que o grupo A teve questões sobre qualidade de processo e o B sobre qualidade de produto. Os autores aplicaram pré e pós-teste para verificar o ganho de conhecimento nos assuntos abordados por cada grupo antes e depois da experiência com o jogo. Também foi aplicado um questionário sobre o uso do jogo, que apresentou resultados positivos na percepção dos estudantes. Apesar disso, alguns estudantes que responderam ao questionário ainda afirmaram que não houve diferença na facilidade de consolidação do conteúdo com o jogo, que o jogo não aumentou a motivação e que não tornou o estudo do conteúdo mais satisfatório, o que pode ser parcialmente explicado pelo fato do jogo ser focado em perguntas e respostas.

3.9 *BlackBox*

BlackBox é um jogo *single-player* para desktop que aborda teste funcional de software (ARAUJO, et al., 2017). O jogo se passa em uma aparente agência de espionagem do governo que está recrutando espiões. Entretanto, a agência, na verdade, é uma organização criminosa comandada por invasores de sistemas. O jogador deve então usar seus conhecimentos para impedir que os vilões alcancem seus objetivos.

Ao iniciar uma missão, o jogador recebe as informações dos objetivos da missão. Dependendo do desempenho em cada missão, o jogador visualiza uma mensagem diferente do resultado da missão, podendo ser uma mensagem de

sucesso ou de fracasso. Na primeira missão, mesmo que o jogador erre a solução, ele pode acessar a missão 2. Entretanto, para acessar as demais missões, o jogador deve solucionar sequencialmente as missões.

As missões do jogo foram desenvolvidas a fim de abranger conceitos sobre classes de equivalência, análise do valor limite e geração de casos de teste. A Figura 11-A apresenta uma das missões do jogo, na qual o jogador deve selecionar as saídas de um programa baseado nas entradas que são fornecidas pelo jogo. A Figura 11-B apresenta a tela com os resultados da missão.

Figura 11. Tela da primeira missão do BlackBox por Araujo et al. (2017).

O jogador ganha o jogo quando conseguir resolver todas as missões do jogo. Se o jogador conseguir finalizar todas as missões, é apresentada a tela final do jogo com uma animação de honra como reconhecimento ao seu trabalho.

O BlackBox foi avaliado com 39 estudantes de graduação do curso de Ciência da Computação. Após os estudantes terem experiência com o jogo, foi aplicado um questionário baseado no Modelo de Avaliação de Jogos Educacionais (MEEGA) (SAVI, et al., 2011) e no *Game Engagement Questionnaire* (GEQ) (BROCKMYER, et al., 2009) para avaliar a percepção dos jogadores com o jogo e o aprendizado percebido. Apesar da avaliação positiva, os estudantes apontaram algumas necessidades de melhorias no jogo, relacionados principalmente com a interface do jogo e da quantidade de desafios.

3.10 *Code Defenders*

Code Defenders é um jogo não digital *multi-player* para ensino de testes de mutação (CLEGG, et al., 2017). O jogo foi construído para permitir que os jogadores pratiquem teste de mutação de forma intuitiva e simples, desenvolvendo testes e verificando a equivalência entre um código mutante e o código original.

O jogo consiste em dois papéis para os jogadores: um atacante e um defensor. Cada jogador assume apenas um dos papéis durante o jogo. No início do jogo, um programa (código-fonte) que está sendo testado é apresentado para os jogadores. O atacante deve criar um programa mutante baseado no original com mudanças tão sutis que não possam ser detectadas pelos testes do defensor. Já o defensor deve criar um conjunto de testes mais completo a fim de detectar as mudanças nos mutantes.

A jogabilidade do jogo ocorre em rodadas, onde primeiro o atacante deve submeter um código mutante e depois o defensor deve submeter novos casos de teste para encontrar os defeitos no código. Se conseguir identificar o mutante, o defensor ganha um ponto na rodada. Caso contrário, o atacante ganha um ponto na rodada. Um cenário especial pode surgir se o defensor suspeitar que o código que o atacante submeteu é um mutante equivalente, ou seja, se comporta da mesma forma que o código original. Com isso, o defensor pode fazer um duelo de equivalência com o atacante, requisitando que este apresente um teste que identifique o código mutante. Se for provado que é um equivalente, o defensor pode ganhar mais pontos. Caso contrário, o atacante ganha mais pontos. A quantidade de rodadas do jogo pode ser determinada pelo aplicador, baseado em suas restrições de tempo ou do espaço do problema que está sendo testado.

O *Code Defenders* apresenta os principais aspectos do teste de mutação, como a criação de mutantes, detecção dos mutantes e verificação da equivalência do mutante.

Rojas et al. (2017) apresentam os resultados de aplicação do jogo com 41 estudantes de mestrado e doutorado. O objetivo do experimento era avaliar se testadores produzem testes melhores durante o jogo e se testadores preferem escrever testes durante o jogo. Os participantes foram divididos em 3 grupos: um grupo escreve testes manualmente, um grupo joga *Code Defenders* como um

defensor e um grupo joga *Code Defenders* como atacante. Ao final da experiência, participantes de todos os grupos responderam um questionário sobre o jogo e o experimento. Com os resultados de aplicação, foi visto que, apesar dos jogadores defensores do *Code Defenders* produzirem menos testes do que os que escreveram testes manuais, a cobertura dos testes gerados por ambos é simular. Isso significa que os jogadores do *Code Defenders* produziram melhores testes do que os que não tiveram experiência com o jogo. Participantes que jogaram o *Code Defenders* também apresentaram maior satisfação criando testes do que os participantes que produziram testes manuais.

3.11 *Bug Hide-seek*

O *Bug Hide-seek* apresenta aos estudantes o objetivo dos testes de unidade, diferenciando implementações corretas e incorretas (BUFFARDI; VALDIVIA, 2018). O objetivo do *Bug Hide-and-Seek* é fazer com que estudantes criem implementações corretas e incorretas para um problema e possam identificar quais implementação resolvem o problema, ou seja, estão corretas.

Buffardi e Valdivia (2018) apresentam duas implementações do jogo. A primeira implementação do jogo foi aplicada com os estudantes sendo divididos em dois papéis diferentes: um de *seeker*, que deveriam programar uma solução correta; e um de *hider*, que deveriam criar códigos relativamente corretos, mas com algumas falhas na implementação, ou seja, criam implementações que se comportam corretamente em alguns casos, mas em outros não. Os estudantes então deveriam enviar o código implementado por eles juntamente com um conjunto de casos de testes que mostrem se o programa implementado por eles está correto ou não. Após isso, cada estudante deve usar seus casos de teste para testar os programas dos demais estudantes e verificar o resultado da execução dos testes.

Na segunda implementação, todos os estudantes assumem o papel de *hider* e têm o objetivo de esconder pequenos erros em implementações quase corretas de um problema e criar um conjunto de casos de testes para identificar tais erros. A premissa é a mesma da primeira implementação, onde os estudantes têm que pensar em um conjunto de casos de teste que revelem se uma implementação de um

programa está correta ou não, entretanto focando mais na verificação das funções do que do programa como um todo.

Nas duas implementações os autores faziam a aplicação em pares de estudantes, aplicando os testes criados por um estudante no código de outro. Para avaliar a capacidade de identificação de erros dos testes dos estudantes, os autores calculam as taxas de verdadeiros positivos e falsos negativos dos conjuntos de testes. A taxa de verdadeiros positivos é a porcentagem de soluções positivas aprovadas. A taxa de falsos negativos é a porcentagem de implementações negativas com falha. Utilizando as taxas, os autores fazem o ranqueamento dos estudantes.

Não foram encontrados na literatura trabalhos que apresentem os resultados de aprendizado ou percepção dos estudantes com o jogo. Buffardi e Valdivia (2018) ainda apresentam os resultados dos jogadores com o jogo como uma forma de avaliar as habilidades dos jogadores em encontrar codificações corretas e incorretas. Mas, até onde foi estudado na literatura, não há resultados referentes ao aprendizado ganho com o jogo ou sobre a percepção dos jogadores.

3.12 *Testing Game*

O *Testing Game* é um jogo Web para apoiar o ensino de testes de software a estudantes de graduação (VALLE, et al., 2017). O jogador controla um avatar que pode executar vários comandos no jogo, como correr, andar, pular, desviar e atacar inimigos, dentre outros comandos. Baseado na descrição e estrutura do programa, os jogadores devem realizar os desafios que são apresentados em cada fase para terminar o jogo.

O jogo consiste em três níveis diferentes e são representados por três portas diferentes no menu do jogo. Cada nível aborda uma técnica para testes de software, que são: estrutural, funcional e baseado em defeitos. Cada nível apresenta desafios diferentes em relação ao conteúdo abordado. Todas as técnicas de teste abordadas no jogo foram projetadas para testar um programa em Java que implementa o algoritmo *BubbleSort*.

No início de cada fase, há uma descrição sobre o que os jogadores devem fazer na fase. Além disso, para cada novo conteúdo considerado no jogo, existe um módulo no qual o jogador pode acessar e ler sobre o novo conteúdo. Além disso, o código-

fonte do programa que implementa o algoritmo BubbleSort pode ser acessado pelo jogador na fase de teste estrutural.

No primeiro nível o jogador deve solucionar desafios relacionados a teste funcional, como selecionar entradas válidas de teste para o programa. No segundo nível o jogador deve realizar desafios relacionadas a teste estrutural de software, como identificar o grafo do fluxo de dados do programa e identificar nós onde ocorrem a definição e uso de variáveis do programa no grafo. A Figura 12 apresenta uma das questões sobre definição e uso de variáveis usando o grafo de fluxo de controle. No terceiro nível, o jogador deve realizar desafios relacionados a teste de mutação, como eliminar mutantes e identificar mutantes equivalentes.

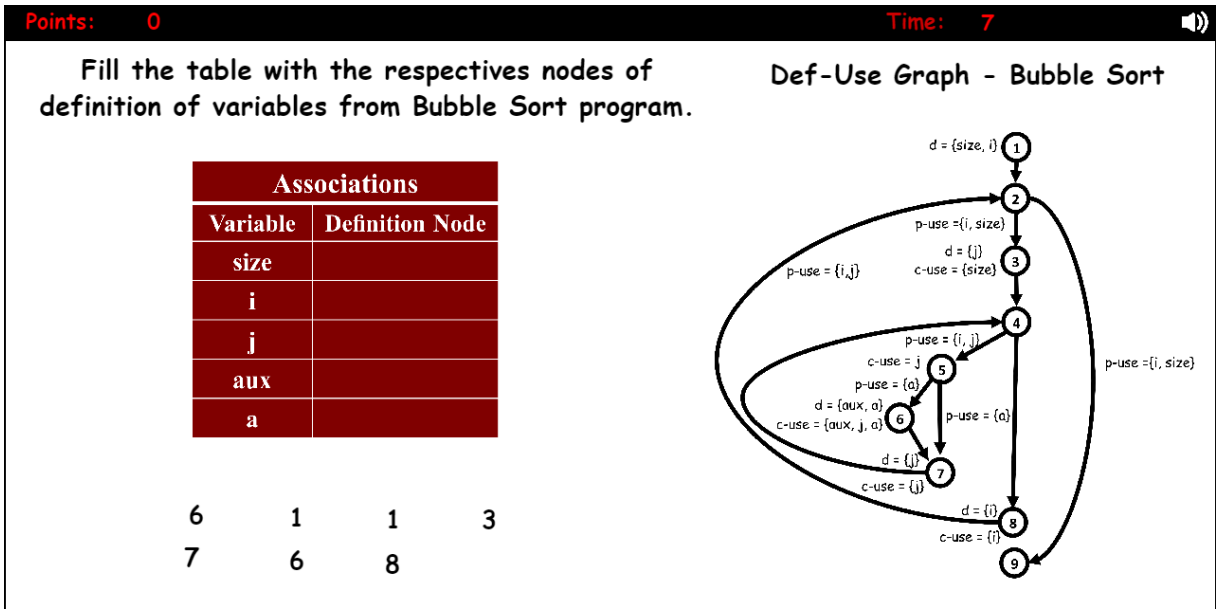


Figura 12. Questão sobre definição e uso de variáveis com o GFC no Testing Game por Valle et al. (2017).

O *Testing Game* foi aplicado com 15 estudantes de mestrado e doutorado. O experimento com o jogo tinha o objetivo de avaliar principalmente a experiência e o aprendizado dos jogadores com o jogo. Para a avaliação, após os jogadores terem a experiência com o jogo, foi aplicado o questionário do MEEGA (SAVI, et al., 2011) para avaliar a experiência e o aprendizado dos jogadores. A avaliação do jogo com o MEEGA foi bastante positiva, os jogadores apontaram apenas que sentiram dificuldade ao avançar no jogo. Após isso, foi realizada uma avaliação da usabilidade do jogo com as heurísticas de Nielsen (NIELSEN, 1994). Após a avaliação heurística,

foram encontrados problemas relacionados principalmente ao controle do jogador no jogo e em relação a ajudas (ou dicas) do jogo.

3.13 *GreaTest*

O *GreaTest* é um jogo *multi-player* de cartas criado para o ensino de testes de software, com o objetivo de ensinar teste de software de forma mais lúdica (BEPPE et al., 2018). No jogo os jogadores devem resolver problemas que surgem com as cartas que eles possuem na mão. Os problemas apresentados no jogo representam cenários de uso de uma aplicação e os jogadores têm que acertar o tipo de teste que revela a situação descrita. Ao responder corretamente o tipo de teste do cenário, o jogador ganha prêmios.

O objetivo do jogo é fortalecer o entendimento dos jogadores sobre qual teste está relacionado a uma situação específica de uso de uma aplicação. Os tipos de testes tratados no jogo são: Teste de Aceitação, Teste de Desempenho, Teste de Estresse, Teste Funcional, Teste de Segurança e Teste de Usabilidade. A escolha dos testes foi baseada nas dificuldades que os autores observaram que os alunos da disciplina de Engenharia de Software tinham em diferenciar e aprender os conceitos sobre os tipos de teste (BEPPE et al., 2018).

O jogo é composto por três decks de cartas:

- Deck de Desafios - contém 38 desafios com cenários de uso de uma aplicação, podendo representar um comportamento correto ou com problema. A Figura 13 apresenta uma carta do deck de desafios. As situações descritas dizem respeito a três aplicações, um sistema bancário, um sistema de chamada de vídeo e um sistema de matrículas;
- Deck de Jogo - contém 48 cartas com os tipos de teste previamente descritos, 8 testadores, 16 cartas extras, como, por exemplo, ferramentas ou reuniões, totalizando um deck de 72 cartas;
- Deck de Bônus - contém 36 cartas com funções especiais, que são prêmios para cada acerto de um jogador.

As cartas de desafio apresentam a descrição de uma situação de uma aplicação e, na parte de trás da carta, apresenta os tipos de teste que podem verificar

a situação descrita. Em cada teste possível há um conjunto de números representam os valores de um dado a ser jogado pelo jogador para acertar a resposta.

No início do jogo, cinco cartas do Deck de Desafio devem ser colocadas com a face para cima, com a descrição da situação à mostra para os jogadores. Em seguida, os participantes devem decidir no dado a ordem de jogada dos jogadores. Antes de iniciar o jogo, cada jogador deve receber cinco cartas do Deck de Jogo e, a cada turno, cada jogador pode puxar duas cartas desse deck. O primeiro jogador pode fazer sua jogada, baixando para a mesa uma carta de testador, caso ele tenha essa carta. Se o jogador não tiver essa carta, ele pula a vez para o próximo jogador.

Figura 13. Cartas de desafio do GreaTest por Beppe et al. (2018).

Ao baixar uma carta de testador, o jogador pode escolher até duas cartas desafio que estão na mesa para responder, considerando as cartas de teste que ele possui na mão. Para resolver o desafio escolhido por ele, ele precisa associar uma carta de teste de sua mão com a situação descrita na carta desafio. Então, após verificar se a carta de teste é compatível com a situação descrita, o jogador deve lançar o dado. Caso ele tire um dos números indicados na carta, ele ganha um ponto e pode puxar uma carta do Deck de Bônus. Ganha o jogador que primeiro conseguir sete pontos no total.

O jogo foi aplicado com 59 estudantes de graduação, mestrado e doutorado. O jogo também foi aplicado em um grupo de quatro professores especialistas em Engenharia de Software para coletar o *feedback* sobre o jogo e sua aplicabilidade. Com os estudantes, foi aplicado o questionário MEEGA (SAVI, et al., 2011) para

avaliar a experiência e aprendizado com o jogo. O questionário MEEGA também foi aplicado com os professores especialistas, com a alteração das perguntas sobre o aprendizado percebido. Apesar do *feedback* geral positivo sobre o jogo, alguns jogadores afirmaram não sentir confiança no aprendizado do conteúdo com o jogo. Uma razão para isso pode ser porque o jogo já prevê que o jogador conheça sobre todas as técnicas e conteúdos abordados e ele só os reforça, não ensinando sobre eles.

O GreaTest foi atualizado para torna-lo um jogo de cartas híbrido, com o auxílio de uma aplicação móvel chamada GreaTest Card Helper para auxiliar na interação dos jogadores com o jogo (DE SOUSA SILVA, et al., 2020). O uso de uma aplicação móvel foi escolhido para tratar de 4 problemas identificados na aplicação do card game: (I) a utilização de um dado, que nem sempre era disponível para ser utilizado; (II) a contabilização dos pontos, que na versão anterior era feita de maneira improvisada; (III) o atraso da rodada de alguns jogadores pode comprometer o progresso do jogo; e (IV) a grande quantidade de regras do jogo torna a memorização difícil para os jogadores.

Com a identificação desses problemas, a aplicação móvel do jogo tem 4 mecânicas para tratar esse problema: um contador de pontos, um temporizador, dados para serem jogados e as regras do jogo descritas no *app*. A Figura 14 apresenta algumas telas do aplicativo GreaTest Card Helper, mostrando algumas das mecânicas incluídas na aplicação, como o temporizador, os dados e as regras do jogo.

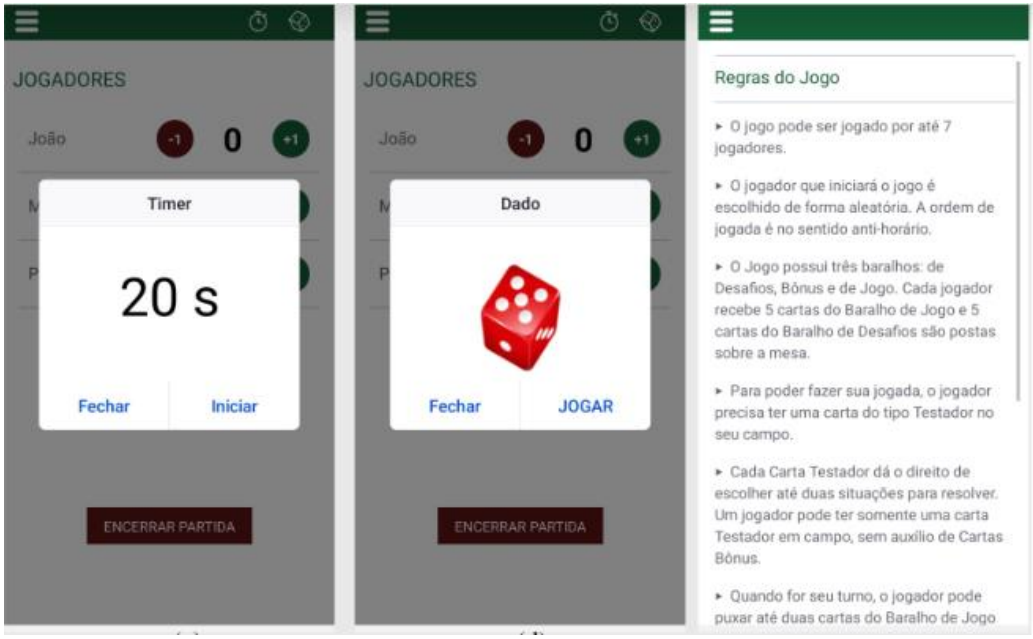


Figura 14. Telas do aplicativo GreaTest Card Helper por De Souza Silva, et al. (2020).

Com a aplicação de questionários para avaliar as mudanças no jogo, os jogadores afirmaram que o aplicativo foi útil para a aplicação do GreaTest e gerou poucas dificuldades e atrasos no jogo.

3.14 *IslandTest*

O *IslandTest* é um jogo educacional online para apoiar o processo de ensino-aprendizagem de testes de software (QUEIROZ, et al., 2019). O enredo do jogo foi inspirado na série de televisão americano Lost. A Figura 15 apresenta a tela inicial do jogo, contendo alguns personagens da série. No jogo, o jogador tem o papel de Jack, um estudante de Sistemas da Informação que decide fazer uma viagem para participar do Congresso Brasileiro de Engenharia de Software (CBES). Entretanto, o avião em que Jack viajava caiu em uma ilha. Jack e outros passageiros sobrevivem à queda do avião e, a fim de conseguir sair da ilha, Jack deve terminar de desenvolver um aplicativo para enviar mensagens instantâneas para smartphones em seu computador.

O aplicativo de Jack está quase finalizado, porém ele ainda não fez os testes no aplicativo. O jogador então deve realizar uma série de desafios no papel de Jack, para testar corretamente o aplicativo de várias formas possíveis, a fim de assegurar

que o aplicativo funciona corretamente. Há vários tipos de desafios no jogo, testando os conhecimentos do jogador sobre vários tópicos testes de software, como as atividades do processo de teste, tipos, níveis e técnicas de teste de software.

Figura 15. Tela inicial do IslandTest por Queiroz, Pinto e Silva (2019).

No jogo, a bateria do computador de Jack tem apenas 30 minutos de funcionamento até que a carga acabe. Por isso, o jogador deve responder corretamente os desafios sobre testes de software antes que a carga da bateria do computador acabe. A cada resposta correta, a carga do computador aumenta, aumentando também o tempo que o jogador tem para terminar os desafios. A cada resposta errada, a carga do computador diminui, consequentemente diminuindo também o tempo para terminar os desafios.

Há vários tipos de desafios no jogo, que testam as habilidades do jogador em tópicos de testes de software, como as fases do processo de testes, níveis de teste, como teste de unidade, integração e de sistema, técnicas de testes, como funcional e estrutural, e critérios para criação de casos de teste, como análise do valor limite.

Este jogo, assim como o que este trabalho tem como objetivo desenvolver, aborda o processo de teste de software. Entretanto, o *IslandTest* apresenta quais são as atividades do processo de teste. O jogo proposto neste trabalho não só abordará as atividades do processo, como também os papéis, artefatos do processo, a resolução de problemas e a gerência do processo de teste.

O *IslandTest* foi avaliado em quatro turmas, contabilizando um total de 85 estudantes participantes. Para avaliar a experiência e aprendizado com o jogo, foi utilizado o questionário de Savi, Wangenheim e Borgatto (2011). A avaliação com o

jogo foi bastante positiva na perspectiva dos jogadores, sendo que algumas sugestões de melhorias foram identificadas na interface do jogo, especialmente na facilidade de entendimento e facilidade para resolução dos desafios do jogo.

3.15 Comparação das propostas

Tendo em vista a variedade de propostas de jogos educacionais que tratam de vários tópicos de teste de software, é necessário fazer uma distinção entre as propostas já estabelecidas e a proposta a ser desenvolvida neste trabalho. O Quadro 3-1 apresenta as características dos jogos educacionais identificados como trabalhos relacionados.

No Quadro 2 é possível ver o tipo de jogo (digital ou não digital), o conteúdo de testes abordado seguindo o conteúdo de teste proposto pelo SWEBoK (2014) e se é um jogo de simulação ou não. A importância de diferenciar os jogos que são de simulação ou não, se deve principalmente às características do conteúdo a ser ensinado. Como apresentado no Capítulo 1 - Introdução, uma das principais requisições da indústria em relação ao ensino de teste de software está relacionada ao processo de teste. Tendo em vista isso, um jogo que apresente o processo ou traga questões sobre algumas atividades do processo ou sobre alguns dos papéis do processo não têm capacidade de ensinar sobre o processo como um todo. Um jogo de simulação que aborde o processo de teste, seus problemas e as responsabilidades dos papéis seria o mais interessante para ensinar sobre o processo de testes. Além disso, estudos como o de Efe e Efe (2011) e Ben-Zvi (2010) mostram que simulações são capazes de promover melhores resultados de conhecimento e níveis mais altos de complexidade, facilitando a tomada de decisão dos estudantes na resolução de problemas.

Quadro 2. Características das propostas para ensino de teste de software. Fonte: O Autor.

JOGO	AUTORES	TIPO DE JOGO	CONTEÚDO DE TESTES
U-TEST	(SILVA & THIRY, 2010)	Digital	- Técnica de teste baseada nas entradas do domínio (funcional)
JETS	(SILVA, et al., 2011)	Digital	- Processo de teste – considerações práticas
Jogo das 7 falhas	(DINIZ & DAZZI, 2011)	Digital	- Técnica de teste baseada nas entradas do domínio (funcional)

iTest Learning	(FARIAS, et al., 2012)	Digital	- Processo de teste – atividades de teste
TestEG	(OLIVEIRA & COSTA, 2013)	Digital	Qualquer conteúdo*
Code hunt	(TILLMANN, et al., 2014)	Digital	- Técnica de teste baseada nas entradas do domínio (funcional)
iLearnTest	(RIBEIRO & PAIVA, 2014)	Digital	- Fundamentos de teste - Técnica de teste baseada nas entradas do domínio (funcional) - Técnica de teste baseada no código (estrutural)
JoVeTest	(BARBOSA, et al., 2016)	Digital	Qualquer conteúdo*
BlackBox	(ARAUJO, et al., 2017)	Digital	- Técnica de teste baseada nas entradas do domínio (funcional)
Code Defenders	(CLEGG, et al., 2017)	Não digital	- Técnica de teste baseado em defeitos
Testing Game	(VALLE, et al., 2017)	Digital	- Técnica de teste baseada nas entradas do domínio (funcional) - Técnica de teste baseada no código (estrutural) - Técnica de teste baseado em defeitos
Bug Hide-seek	(BUFFARDI & VALDIVIA, 2018)	Não digital	- Técnica de teste baseada nas entradas do domínio (funcional) - Técnica de teste baseada no código (estrutural) - Técnica de teste baseado em defeitos
GreaTest	(BEPPE, et al., 2018)	Não digital	- Níveis de teste
IslandTest	(QUEIROZ, et al., 2019)	Digital	- Processo de teste – considerações práticas - Níveis de teste - Técnica de teste baseada nas entradas do domínio (funcional) - Técnica de teste baseada no código (estrutural)

* O jogo permite receber questões enviadas pelos instrutores para a base de questões do jogo. Portanto, qualquer conteúdo de teste de software pode ser incluído para esse jogo.

Como pode ser visto no Quadro 2, há alguns jogos que abordam o conteúdo do processo de teste, mas eles apresentam falhas em relação ao ensino completo deste tópico ou até da forma como o conteúdo é abordado. O IslandTest (QUEIROZ, et al., 2019) apresenta, em forma de perguntas, as atividades do processo de teste de software e a visão geral do fluxo das atividades. O IslandTest não aborda os papéis e os artefatos nem como realizar cada uma das atividades de teste.

O iTest Learning (FARIAS, et al., 2012), por sua vez, é um jogo de simulação das atividades da atividade de planejamento de testes. Entretanto, o jogo aborda uma visão geral do planejamento de testes, mas não aborda papéis, não apresenta as relações entre as demais atividades de teste, nem quando devem ser gerados os artefatos e nem problemas que podem surgir ao longo da execução do processo. Além

disso, o jogo foca mais nos tipos, níveis e artefatos de teste do que realmente no processo de teste.

O JETS (SILVA, et al., 2011) é um jogo de simulação dos papéis do processo de teste. Este jogo, ao contrário dos outros, diferencia as responsabilidades dos papéis da equipe de teste, mas não mostra as atividades do processo, a relação entre as atividades e os artefatos que devem ser produzidos. Além disso, a própria ordem das fases do jogo pode dificultar o entendimento sobre o processo. O jogador inicia como Testador, se torna Analista, Arquiteto e por fim se torna o Líder da equipe. Entretanto, na realidade, as atividades de planejamento (responsabilidade do Líder), infraestrutura (responsabilidade do Arquiteto), elaboração (responsabilidade do Analista) e execução dos testes (responsabilidade do Testador) são realizadas em uma ordem diferente do que o jogo apresenta.

Baseado em alguns dos problemas encontrados nos jogos educacionais que abordam o processo de teste de software, é possível notar que nenhum deles inclui totalmente o processo com suas atividades, papéis, artefatos e possíveis problemas que podem surgir. Diante disso, foi inserida no *Quadro 2* a abordagem que este trabalho propõe realizar com algumas de suas características.

3.16 Considerações finais

Este capítulo de trabalhos relacionados apresentou as propostas de jogos educacionais encontradas na revisão da literatura. As características, o conteúdo tratado e a forma de aplicação de cada jogo foram apresentados neste capítulo. Por fim, foi apresentada uma visão geral sobre as propostas, suas diferenças, destacando principalmente as diferenças entre os jogos que abordam o processo de teste de software do jogo proposto neste trabalho. O próximo capítulo apresenta a estruturação da pesquisa, mostrando o método de pesquisa que será utilizado neste trabalho.

CAPÍTULO 4 - ESTRUTURAÇÃO DA PESQUISA

“Não existem métodos fáceis para resolver problemas difíceis.”

René Descartes

Este capítulo tem como objetivo apresentar o método de pesquisa escolhido para a condução da pesquisa, bem como retoma o objetivo principal da pesquisa e define as atividades que foram realizadas para o seu cumprimento.

4.1 Seleção do método de pesquisa

Devido às características da pesquisa, cujo objetivo é a criação de um artefato que é embasado tanto no *design* do artefato, o jogo em si, quanto no problema, que é o ensino de teste, mais especificamente a motivação para aprender sobre testes, optou-se pelo método *Design Science Research* (DSR).

A *Design Science Research* constitui um processo rigoroso de projetar artefatos para resolver problemas, avaliar o que foi projetado e comunicar os resultados obtidos (ÇAĞDAŞ & STUBKJÆR, 2011). Segundo Bayazit (2004), a DSR se preocupa com:

- a incorporação física das coisas feitas pelo homem, como essas coisas realizam suas funções e como funcionam;
- a construção, como uma atividade humana, como *designers* trabalham, como pensam e como fazem a atividade de *design*;
- com o que é alcançado no final de uma atividade de *design*, como uma coisa artificial aparece e o que ela significa;
- com a incorporação de configurações;
- em ser uma busca sistemática e aquisição de conhecimento relacionado ao *design* e à atividade de *design*.

Para Simon (1996, p. 113), *Design Science* é *“uma doutrina intelectualmente robusta, analítica, parcialmente formalizável, parcialmente empírica e ensinável sobre*

o processo de design". O principal objetivo da *Design Science* é desenvolver conhecimento para a criação e desenvolvimento de artefatos (AKEN, 2004). É importante destacar que *Design Science* é relacionada com a criação sistemática de conhecimento sobre o *design* e com *design*. A *Design Science* é direcionada à compreensão e procura por componentes potenciais para construir um artefato que se destina a resolver um problema (BARKERVILLE, 2008).

A Figura 16 apresenta a abordagem utilizada nessa pesquisa seguindo o método proposto por Peffers et al. (2007). Nela são apresentadas as 6 atividades da *Design Science Research* e dentro de cada atividade são apresentas as ações realizadas no escopo desta pesquisa. As etapas serão detalhadas nas próximas seções.

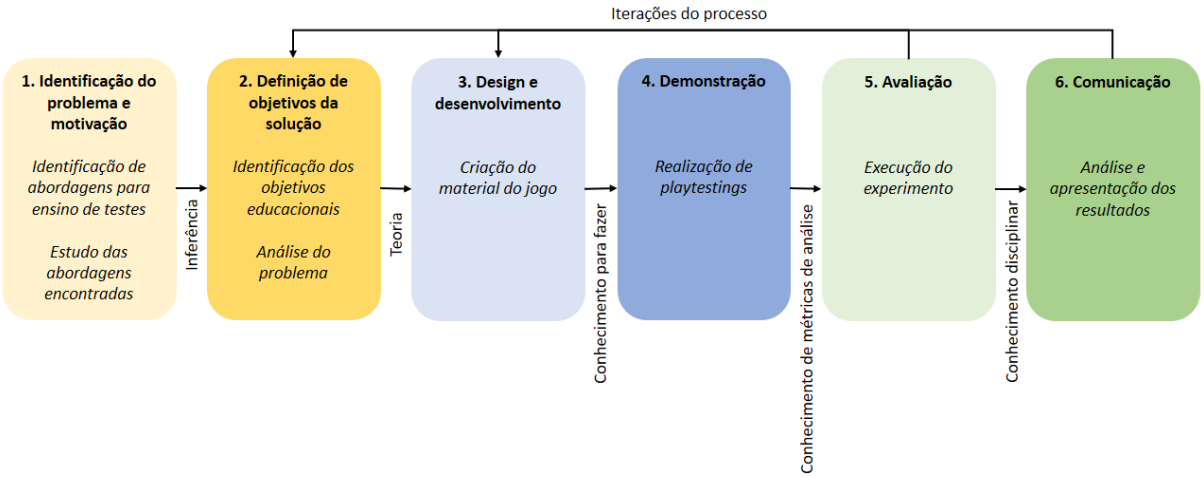


Figura 16. Método de Pesquisa usado para o desenvolvimento do jogo educacional, adaptado de Peffers et al. (2007).

4.2 Etapas da Pesquisa

4.2.1 Atividade 1 – Identificação do problema e motivação

A atividade de Identificação do problema e motivação foi realizada por 2 tarefas dentro do projeto de pesquisa: identificação de abordagens para o ensino de teste de software e estudo das abordagens encontradas, conforme relatado nos capítulos anteriores.

A partir do artigo de Paschoal e De Souza (2018) e do mapeamento sistemático de Valle et al. (2015), foram realizadas buscas do tipo *snowballing* para identificar as

principais abordagens para ensino de teste de software. Para alguns dos trabalhos encontrados no primeiro *snowballing,* foram realizadas novas buscas do tipo *snowballing* para identificar mais trabalhos relacionados.

Como resultados dessa fase, foram encontradas 51 abordagens que de alguma forma tratam do ensino de teste de software. As abordagens foram organizadas em uma planilha e para cada uma delas foram destacados: o tipo de abordagem, o nome da abordagem, a referência para o trabalho que apresenta a abordagem, a motivação, como funciona a abordagem e os resultados que foram obtidos com a aplicação da abordagem.

Ao final do estudo dessas abordagens foram encontradas sete motivações/desmotivações principais relacionadas ao ensino de testes: (M1) Desmotivação dos estudantes para aprender testes de software; (M2) Má estruturação dos cursos; (M3) Diferença entre práticas da indústria e os conteúdos ensinados em sala; (M4) Técnicas e práticas de teste de software não são totalmente empregadas pelas empresas; (M5) Assegurar que os estudantes adquiram experiência na aplicação dos conceitos por meio da prática; (M6) Estudantes não têm oportunidades para praticar técnicas para validar os programas por eles desenvolvidos na graduação; e (M7) Pouco conhecimento dos profissionais da indústria em relação a testes de software.

Grande parte das abordagens são jogos educacionais, sendo usadas principalmente para ensino das técnicas e critérios para elaboração de casos de teste. Mesmo com a importância do processo para a melhor aplicação de testes de software, em um estudo mais focado nos jogos educacionais, foi visto que nenhum deles aborda o processo de teste de software por completo. Esses jogos foram apresentados e discutidos no Capítulo 3.

4.2.2 Atividade 2 – Definição de objetivos da solução

A atividade de Definição de objetivos da solução foi realizada por 2 tarefas dentro do projeto de pesquisa: análise do problema e identificação dos objetivos educacionais.

Diante da falta de jogos educacionais que foquem no ensino do processo de teste de software e considerando os benefícios que jogos educacionais podem ter no

contexto de simulação de cenários reais (SAVI, 2011), o objetivo da pesquisa é a criação de um jogo educacional que aborda tal conteúdo de testes de software.

Tendo em vista as motivações identificadas, um jogo educacional para ensino do processo de teste deveria ter a capacidade de tratar tais problemas:

- M1 - Desmotivação dos estudantes para aprender testes de software: considerando o entretenimento que os jogos podem fornecer durante seu uso, é esperado que um jogo educacional motive os estudantes a aprender o conteúdo de teste de software;
- M3 - Diferença entre práticas da indústria e os conteúdos ensinados em sala: ao simular como o processo de teste de software ocorre, é possível aproximar mais as práticas da indústria do ensino aos estudantes;
- M5 - Assegurar que os estudantes adquiram experiência na aplicação dos conceitos por meio da prática: por ser um jogo que simula o ambiente real, é esperado que os jogadores tenham experiência prática com as atividades de teste, apesar de ainda ser uma simulação.
- M7 - Pouco conhecimento dos profissionais da indústria em relação a testes de software: com a criação de um jogo que aborda de forma prática o ensino de testes de software é possível fazer com que os futuros profissionais, hoje estudantes, tenham mais conhecimento durante suas atividades na indústria.

Como forma de guiar a criação do jogo, foi criado um plano de desenvolvimento, que apresenta o problema e o conteúdo de ensino do jogo. Para isso foi criado o modelo de planejamento feito com base no quadro de apoio fornecido por Rocha e Araujo (2014) que é apresentado no Quadro 3.

Quadro 3. Planejamento seguindo a metodologia DevJSTA (ROCHA & ARAUJO, 2014). ***Fonte: o Autor.***

Planejamento	
Protocolo	Jogo para ensino de teste de software
Treinamento	
Problema real	O ensino de teste de software, por ser um tópico que é diretamente relacionado com a prática das atividades na indústria, é difícil de ser tratado de forma prática em disciplinas de graduação. Isso faz com que os profissionais formados não saibam exatamente como realizar as atividades de teste na indústria de maneira organizada
Alvo do treinamento	Estudantes de graduação em Ciência da Computação e áreas afins

Objetivos do treinamento	O jogo deverá fazer com que os estudantes adquiram as seguintes competências: • C1. Organizar as principais atividades que compõem o processo de testes de software; • C2. Distinguir os principais artefatos de teste para as atividades em que devem ser produzidos; • C3. Organizar as responsabilidades de cada papel do processo de teste; • C4. Analisar algumas possíveis situações/problemas que comumente ocorrem durante o processo.
Método de treinamento	Um jogo educacional que permita a simulação do ambiente de teste de software, focando principalmente no ensino sobre o processo de teste de software, suas atividades, papéis, artefatos e situações que podem ocorrer durante
Simulação	
Objetivo da simulação	Facilitar o entendimento, por meio de uma prática simulada, sobre como ocorre o processo de teste
Possível cenário para simulação	Um ambiente onde um software para resolver um problema está sendo desenvolvido em várias empresas concorrentes. Os funcionários de uma empresa (jogadores) deverão realizar as atividades de teste, seguindo o processo de teste de software, de maneira com que as atividades realizadas garantam maior qualidade ao software desenvolvido. E, com isso, a empresa dos jogadores tenha vantagem no mercado pela qualidade do produto desenvolvido
Anexos	

4.2.3 Atividade 3 – *Design* e desenvolvimento

A atividade *Design* e desenvolvimento foi por iterações de criação de versões do jogo e aplicação de *playtestings* para avaliação das alterações do jogo. Inicialmente foi estabelecida em termos de conteúdo e jogabilidade uma versão base do jogo que foi acertada com 2 pesquisadores experientes, um na área de jogos educacionais e o outro em qualidade de software. A partir da versão base e de melhorias apontadas foram realizadas as iterações de *design* e *playtesting*. A cada *playtesting* eram discutidos tópicos relacionados a jogabilidade, atratividade e conteúdo do jogo. Com as discussões de cada *playtesting,* melhorias eram identificadas no jogo que eram realizadas e reavaliadas em *playtestings* seguintes. No Capítulo 5 são apresentadas as versões do jogo que foram elaboradas ao longo da atividade de *design* e desenvolvimento e no Capítulo 6 são apresentadas as discussões e os detalhes de cada *playtesting* realizados para avaliar as versões do jogo.

4.2.4 Atividade 4 – Demonstração

A atividade de Demonstração tem o objetivo de demonstrar que o jogo desenvolvido é capaz de atingir os objetivos motivacionais e educacionais propostos.

A fim de avaliar o artefato criado, foi utilizado o *framework* proposto por Pries-Heje, Baskerville & Venable (2008), que divide as avaliações em pesquisas de *Design Science* em dois tipos: *Ex ante* e *Ex post*. A avaliação *Ex ante* fornece modelos para avaliar teoricamente um *design* sem realmente implementar o sistema de materiais ou tecnologia. Em outras palavras, o artefato é avaliado apenas com base em suas especificações de *design*. Já a avaliação *Ex post* fornece modelos para avaliar um *design* já em seu ambiente de final uso.

Em relação à pesquisa, foram utilizadas as duas formas de avaliação. Isso porque um jogo educacional traz mais complicações do que o próprio *design* do problema, como a jogabilidade e aspectos relacionados ao divertimento com o jogo. Por isso, foram realizadas três rodadas de *playtesting* (avaliação *Ex ante*) para avaliação do jogo em relação à sua jogabilidade e aprendizagem percebida por jogadores de tabuleiro. No final das rodadas de *playtesting*, o artefato passou por diversas mudanças e foi feita uma avaliação *Ex post* com os jogadores-alvo do artefato, que são estudantes de graduação em cursos de computação, em que foi realizado o experimento final de avaliação da pesquisa. No Capítulo 6 serão apresentadas as avaliações feitas com os *playtestings.*

4.2.5 Atividade 5 – Avaliação

A atividade de Avaliação tem o objetivo de observar e medir o quanto o artefato é capaz de apoiar a solução do problema abordado. Neste projeto esta atividade é realizada pela tarefa de execução do experimento. Para avaliar o jogo e verificar os cumprimentos de seus objetivos educacionais e motivacionais foram aplicados questionários de pré e pós-teste. O questionário de pré-teste tinha o objetivo de obter a aceitação dos participantes no experimento, conhece-los em relação à experiência deles com jogos e com testes de software e verificar o conhecimento que os participantes tinham sobre tópicos do processo de teste de software antes da aplicação do jogo. O questionário de pré-teste é composto por três sessões: a Sessão 1 com o termo livre e esclarecido de aceitação, Sessão 2 com perguntas demográficas

adaptadas do questionário do MEEGA+ e a Sessão 3 com perguntas relacionadas ao processo de teste de software. As perguntas sobre o processo de teste na Sessão 3 foram definidas com base no conteúdo a ser abordado no jogo e foram validadas por 2 pesquisadores experientes, um na área de jogos educacionais e o outro em qualidade de software.

O questionário de pós-teste tinha o objetivo de verificar o conhecimento dos jogadores sobre os tópicos do processo de teste após o experimento do jogo e verificar a percepção dos jogadores nos aspectos de usabilidade, motivação, desafio, satisfação e outros promovidos pelo jogo. O questionário de pós-teste é composto por duas sessões: a Sessão 1 com perguntas relacionadas ao processo de teste de software, assim como no pré-teste e a Sessão 2 com perguntas dos fatores de Usabilidade e Experiência do jogador definidos pelo MEEGA+. As perguntas sobre o processo de teste na Sessão 1 são praticamente iguais às questões sobre o processo no pré-teste, com exceção de uma questão, que é a última. Isso foi feito propositalmente a fim de diminuir a influência que a dificuldade que um questionário pode ter na avaliação dos estudantes e, consequentemente, na comparação dos resultados de pré e pós-teste. Novamente as questões do pós-teste foram baseadas no conteúdo a ser abordado no jogo e foram validadas por 2 pesquisadores experientes, um na área de jogos educacionais e o outro em qualidade de software. O Quadro 4 mostra a relação entre as questões sobre o processo de teste no pré e pós-teste com as competências do processo de teste a serem obtidas com o jogo.

Quadro 4. Relação das questões sobre o processo de teste nos pré e pós-teste com as competências definidas.

TESTE	QUESTÃO	COMPETÊNCIA
- Pré-teste - Pós-teste	Há várias formas de se organizar o processo de teste de software. Entretanto todas as formas devem seguir uma ordem lógica para que seja executado corretamente. Com base nisso, selecione a opção que melhor apresenta a ordem de atividades do processo de teste	C1. Organizar as principais atividades que compõem o processo de testes de software
- Pré-teste - Pós-teste	João é um funcionário eficiente, detalhista e de certa forma antissocial. João tem muitas habilidades com programação, é muito atencioso e sabe solucionar problemas em seus programas e de seus coletas, o que o ajuda muito em seu trabalho. Dentro da empresa onde trabalha, João é responsável pela a implementação de casos de teste e com a execução dos testes para testar os produtos de software da empresa. Tendo em vista a descrição acima da persona de João, é possível afirmar	C3. Organizar as responsabilidades de cada papel do processo de teste

	que as características apresentadas mais se relacionam a qual papel do processo de teste de software?	
- Pré-teste - Pós-teste	Luiza é uma funcionária extremamente dedicada e comunicativa. Ela gosta de saber de tudo o que acontece na empresa. Luiza tem grandes habilidades de planejamento, negociação e gestão, especialmente gestão de pessoas. Por conta disso, Luiza assumiu um papel importante na empresa onde trabalha. Ela é responsável por fazer acordos sobre os objetivos e entregáveis do processo do qual é responsável. Além disso, como forma de promover melhoria continua desse processo, ela sempre busca avaliar a efetividade e produtividade e manter dados sobre a execução dos processos que coordena. Tendo em vista a descrição acima da persona de Luiza, é possível afirmar que as características apresentadas mais se relacionam a qual papel do processo de teste de software?	C3. Organizar as responsabilidades de cada papel do processo de teste;
- Pré-teste - Pós-teste	Há vários papéis responsáveis por executar o processo de teste. Qual são as principais atribuições do analista de teste?	C3. Organizar as responsabilidades de cada papel do processo de teste
- Pré-teste - Pós-teste	Para que os testes possam ser devidamente executados, é necessário se ter produzido o artefato ____1____, que é uma coleção de casos de teste utilizados para testar um produto. Esse artefato é produzido pelo Analista de teste, ou designer de teste, que, para auxiliar na aplicação desses casos de teste cria também o artefato ____2____, que ajuda os testadores na execução dos testes. Em relação ao processo de teste, marque a opção que apresenta as respectivas palavras que corretamente completam o texto.	C1. Organizar as principais atividades que compõem o processo de testes de software C2. Distinguir os principais artefatos de teste para as atividades em que devem ser produzidos C3. Organizar as responsabilidades de cada papel do processo de teste
- Pré-teste - Pós-teste	O artefato Script de teste é responsável especificamente por:	C2. Distinguir os principais artefatos de teste para as atividades em que devem ser produzidos
- Pré-teste - Pós-teste	O artefato ____1____ é produto do processo de teste e deve ser produzido logo após a execução dos testes. Este artefato é produzido pelo ____2____ tem o objetivo principal de ____3____. Em relação ao processo de teste, marque a opção que apresenta as respectivas palavras que corretamente completam o texto.	C2. Distinguir os principais artefatos de teste para as atividades em que devem ser produzidos C3. Organizar as responsabilidades de cada papel do processo de teste

- Pré-teste - Pós-teste	A execução do conjunto de testes para verificar o produto no processo de teste é realizada principalmente pelo ____1____, que também será responsável por analisar as falhas encontradas por esses testes. Como resultado da execução dos testes, é gerado um artefato ____2____, que poderá ser utilizado futuramente para as recomendações de correções. Em relação ao processo de teste, marque a opção que apresenta as respectivas palavras que corretamente completam o texto.	C2. Distinguir os principais artefatos de teste para as atividades em que devem ser produzidos C3. Organizar as responsabilidades de cada papel do processo de teste
- Pré-teste - Pós-teste	O acordo com o cliente sobre os objetivos dos testes e os entregáveis de cada interação é realizado pelo ____1____, e por isso esse papel exige muito ____2____. Como resultado deste acordo, é gerado o ____3____, que definirá os métodos e procedimentos a serem utilizados bem como será utilizado futuramente para a verificação do atingimento do nível acordado de qualidade do software testado. Em relação ao processo de teste, marque a opção que apresenta as respectivas palavras que corretamente completam o texto.	C1. Organizar as principais atividades que compõem o processo de testes de software C2. Distinguir os principais artefatos de teste para as atividades em que devem ser produzidos C3. Organizar as responsabilidades de cada papel do processo de teste
- Pré-teste	Analise a seguinte situação: Gilberto é o líder da equipe de testes de uma empresa. A empresa em que Gilberto trabalha mantém diversos sistemas e atualmente está avançando com a implementação de um novo sistema de gerenciamento de estoques. Gilberto, ao voltar de suas férias, nota que há uma grande confusão sobre as atividades que estão sendo realizadas. Ninguém da equipe de teste sabe o que está sendo feito, o que vem impedido as atividades de teste de serem realizadas. Analisando a situação, quais ações Gilberto poderia fazer para corrigir tal problema?	C4. Analisar algumas possíveis situações/problemas que comumente ocorrem durante o processo
- Pós-teste	Analise a seguinte situação: A Sys é uma empresa antiga, com funcionários fiéis e empenhados em suas atividades. Entretanto, em busca de novos mercados, a Sys iniciou um trabalho inovador dentro da empresa focado no mercado de aplicativos móveis. Só que há um problema: a equipe de testes está tendo dificuldade para testar e entender as novas tecnologias utilizadas no projeto. Analisando a situação, qual ação a empresa Sys poderia fazer para corrigir tal problema?	C4. Analisar algumas possíveis situações/problemas que comumente ocorrem durante o processo

4.2.6 Atividade 6 - Comunicação

A última atividade da abordagem seguida é a Comunicação que tem o objetivo de comunicar os resultados obtidos com o artefato, sua utilidade e teste das hipóteses. Para a comunicação dos resultados foram preparados gráficos definidos pelas ferramentas de medida utilizadas no experimento com o jogo bem como gráficos

estatísticos que apresentam os resultados obtidos com o experimento. Neste projeto esta atividade é realizada pela tarefa de análise e apresentação dos resultados, que será apresentada no Capítulo 6.

4.3 Considerações sobre o Capítulo

Este capítulo apresentou uma visão geral da estrutura da pesquisa que se baseou na *Design Science Research* (DSR), conforme definido por Peffers (2007). Foram explicadas as seis etapas do método e como formam organizadas no âmbito deste trabalho.

CAPÍTULO 5 - PROTESTERS

“Eu não falhei 10 mil vezes. Apenas encontrei 10 mil maneiras que não funcionam.”

Thomas Edinson

Neste capítulo é apresentado o jogo educacional criado para o cumprimento dos objetivos desta pesquisa que foi denominado ProTesters. O ProTesters foi construído de maneira iterativa e incremental, utilizando os feedbacks obtidos nas sessões de *playtestings*. Como forma de apresentar as alterações que o jogo sofreu e facilitar o entendimento das discussões geradas nos *playtestings* e no experimento com o jogo, optou-se, neste capítulo, por mostrar inicialmente a versão final e em seguida as versões intermediárias utilizadas em cada *playtesting*.

5.1 Visão geral do jogo

Após todas as discussões e aplicações realizadas, a versão final ficou pronta e o jogo elaborado foi chamado de ProTesters, como uma referência ao processo de teste, como também uma analogia para testadores profissionais. ProTesters é um jogo de tabuleiro jogado em 6 rodadas, no qual os jogadores trabalham a atribuição de atividades, solução de problemas e simulação da execução do processo de teste.

Para a definição do conteúdo a ser abordado, foi utilizado o processo de software, papéis, atividades e artefatos definidos pelo RUP (RATIONAL SOFTWARE CORP, 2021), considerando que esse processo é robusto e completo. Assim como apresentado no Capítulo 2, o RUP define quatro papéis principais: Gerente de teste, Analista de teste, *Designer* de teste e Testador. Cada papel do RUP possui um representante no ProTesters, mas o papel do Analista de teste, por ser considerado um papel que raramente é usado na indústria na realidade foi mesclado com o papel do *Designer*, para juntar as responsabilidades de ambos e melhor representar o

papéis do processo de teste praticado na indústria. Para o ProTesters foram definidos três papéis embasados nos quatro papéis do RUP:

- Líder de teste: representa o papel de Gerente de teste do RUP e é relacionado com atividades de planejamento e gestão das atividades de teste;
- Analista de teste: representa a junção dos papéis de Designer e Analista de teste do RUP e é responsável por atividades de definição de requisitos e abordagens de teste; e
- Testador: representa o papel Testador do RUP e é responsável pela execução dos testes e registro dos resultados.

Em relação as atividades de teste, o RUP define 21 atividades distribuídas para os quatro papéis definidos. Como o RUP é um processo muito completo e formal, para o ProTesters foram escolhidas nove principais atividades dentre as 21 que foram definidas pelo RUP. Essas atividades foram escolhidas pois seriam as mais importantes para a execução do processo de teste e que seriam imprescindíveis para qualquer processo de testes. Essas nove atividades estão relacionadas ao papel que é responsável por executá-las, sendo que cada papel no jogo está relacionado a três atividades.

O RUP também define vários artefatos que são produzidas pelas atividades de teste. Entretanto, nem toda atividade produz artefatos de teste e por isso apenas sete artefatos foram definidos no ProTesters. O Quadro 5 traz os papéis do processo de teste definidos pelo ProTesters, as atividades pelas quais eles são responsáveis e as atividades que cada atividade produz.

Quadro 5. Relação entre papéis, atividades e artefatos do processo de teste do ProTesters.

PAPÉIS	ATIVIDADE	ARTEFATO
Líder de teste	Fazer acordo sobre os objetivos e entregáveis das interações	Plano de teste
	Avaliar a efetividade e produtividade e fazer melhorias	Relatório de teste
	Identificar eventos e artefatos que influenciarão nos testes	-
Analista de teste	Estruturar a suíte de testes automatizados	Suíte de teste
	Fazer sumário dos resultados de teste para propor correções	Resultado de teste
	Escrever guia para execução da suíte de testes	Guia de teste
Testador	Executar o conjunto de testes para verificar o produto	Log de teste
	Implementar conjunto de testes significativo	Script de teste

	Analisar falhas que ocorreram durante a execução do programa	-

Como forma de medir a boa execução do processo, e consequentemente a vitória do jogo, é utilizado o marcador do bugômetro. A utilização de bugs como forma de vitória é refletida pelo processo de teste na realidade, cujo objetivo é maximizar a capacidade de se encontrar bugs no produto testado. A Figura 17 apresenta um exemplo de cartão do jogador com o bugômetro na parte de baixo e o espaço para os funcionários e atividades no centro.Após todas as discussões e aplicações realizadas, a versão final ficou pronta e o jogo elaborado foi chamado de ProTesters, como uma referência ao processo de teste, como também uma analogia para testadores profissionais. ProTesters é um jogo de tabuleiro jogado em 6 rodadas, no qual os jogadores trabalham a atribuição de atividades, solução de problemas e simulação da execução do processo de teste. O Quadro 6 apresentada a relação de material do jogo ProTesters. O material completo do jogo está presente no Apêndice B, com todas as suas cartas e materiais necessários para executá-lo. O material do jogo também está disponível no Google Drive[1].

Quadro 6. Material do jogo ProTesters. Fonte: o Autor.

Material do jogo	
Cartões do jogador	4 cartões (amarelo, marrom, verde e vermelho)
Deck	64 (16 para cada jogador)
Cartas de atividade	36 (9 para cada jogador)
Artefatos	28 (7 para cada jogador)
Ambiente	15 (divididos em 3 níveis)
Metas	15 (divididas em 3 níveis)
Funcionários	24 (8 para cada papel)
Moedas	50
Esforço	50

[1] Disponível em: https://drive.google.com/drive/folders/1NYOMmlShcrD10c8YCAv1CTk2YZFSYDOZ Acesso 09/01/2022

Marcadores de bugômetro	4 (1 para cada jogador)

Como forma de medir a boa execução do processo, e consequentemente a vitória do jogo, é utilizado o marcador do bugômetro. A utilização de bugs como forma de vitória é refletida pelo processo de teste na realidade, cujo objetivo é maximizar a capacidade de se encontrar bugs no produto testado. A Figura 17 apresenta um exemplo de cartão do jogador com o bugômetro na parte de baixo e o espaço para os funcionários e atividades no centro.

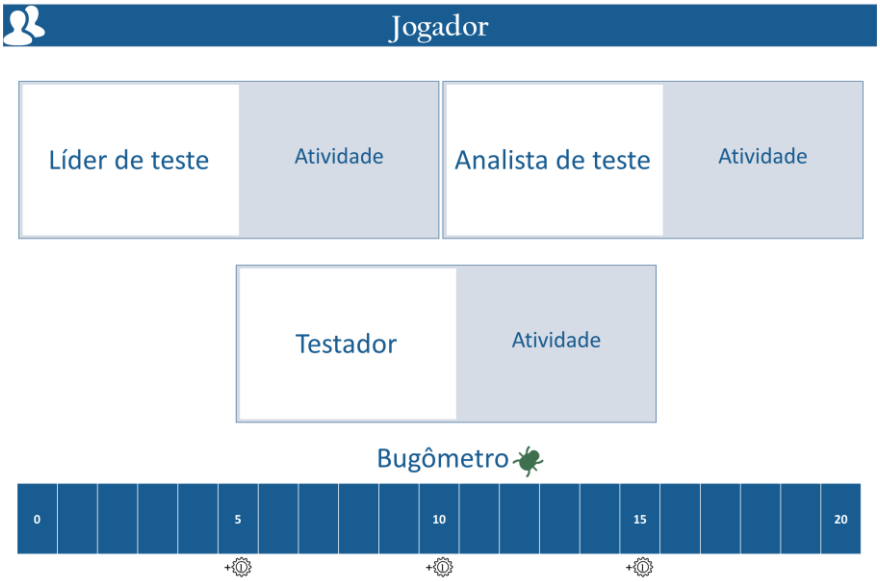


Figura 17. Exemplo de cartão de jogador do ProTesters. Fonte: o Autor.

Como benefício por executar corretamente o processo, o bugômetro pode gerar bônus para os jogadores que atingem certos *checkpoints*, como 5, 10 e 15 pontos. Ao atingir esses *checkpoints* o jogador ganha mais esforço por rodada, simulando a facilidade que existe na realidade de realizar atividades em um processo mais bem definido e sequencialmente mais lógico.

O jogo também contém cartas de *deck*, que são divididas entre cartas de sabotagem e de soluções para problemas. As cartas de sabotagem são utilizadas no final de cada rodada em um outro jogador e tem um efeito lúdico de promover a competição entre jogadores e gerar imprevisibilidade no jogo. A Figura 18 apresenta dois exemplos de cartas de sabotagem.

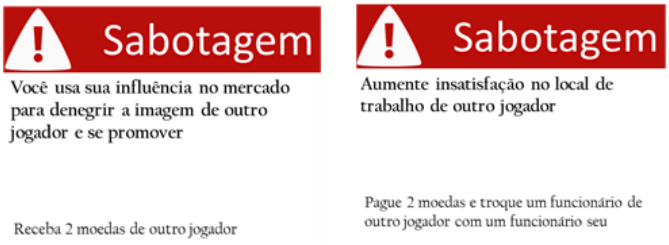


Figura 18. Exemplos de cartas de sabotagem. Fonte: o Autor.

Além das sabotagens, o *deck* de cada jogador também tem cartas de solução para problemas, que, como o próprio nome sugere, solucionam problemas que podem ocorrer nos ambientes. Ao usar uma solução contra uma carta de problema, o efeito do problema é anulado e um bônus em forma de pontos no bugômetro é dado para o jogador. A Figura 19 apresenta a relação entre solução e problema que pode ocorrer.

Figura 19. Carta de ambiente problema e sua solução. Fonte: o Autor.

Cada jogador tem um conjunto de 11 cartas de atividade que ele deve cumprir. Cada atividade tem um título, que descreve a atividade que ela representa, a esforço, as habilidades requisitadas para cumpri-la e o artefato que ela produz. Além disso, cada carta de atividade tem uma *tag*, informando o tipo da atividade, o que ajuda os jogadores a atribuir atividades para os funcionários corretos. A Figura 20 apresenta exemplos de cartas de atividade.

Figura 20. Exemplos de cartas de atividade. Fonte: o Autor.

Ao concluir uma atividade, o artefato que ela produzir é obtido pelo jogador. Cada artefato tem um nome, que reflete o artefato que está sendo representado e a descrição do artefato. Como consequência por ter sido produzido, cada artefato gera um bônus em moedas e pontos no bugômetro. A Figura 21 apresenta exemplos das cartas de artefatos do jogo.

Figura 21. Exemplos de cartas de artefato. Fonte: o Autor.

Cada atividade do jogo deve ser atribuída a algum funcionário para que ele possa executá-la. As cartas de funcionário representam um papel do processo de teste, e têm um custo de contratação e as habilidades do funcionário. Cada funcionário também tem uma esforço de teste que é somado com o restante do time para gerar o esforço do time por rodada, que a cada rodada é usado para executar atividades.

Cada funcionário também tem um bônus de especialidade, que diminui o esforço necessário para realizar um certo tipo de atividade. A Figura 22 apresenta uma carta de Analista de teste, que têm um bônus para realizar a atividade apresentada ao lado. Com isso, ao atribuir essa atividade para o Analista, ao invés de 4 de esforço para cumprir essa atividade, será necessário apenas 3.

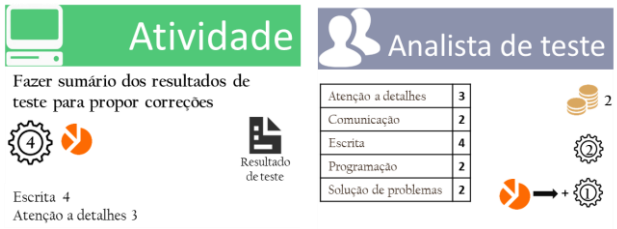


Figura 22. Suposição de atribuição de atividades para funcionário. Fonte: o Autor.

As cartas de ambiente podem ser tanto um problema como uma carta de efeito positivo ou negativo. Essas cartas têm um objetivo lúdico de trazer competitividade, adversidade entre jogadores, como também um efeito de estimular a aprendizagem do processo, apresentando situações que refletem situações reais do processo de

teste. A Figura 23 apresenta dois tipos de carta de ambiente, a primeira uma carta de ambiente de efeito negativo e a segunda, uma carta de problema do tipo Execução, que impacta as habilidades dos funcionários do jogador que não possuir uma solução para o problema.

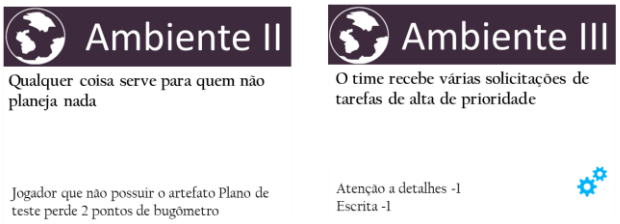


Figura 23. Exemplos de cartas de ambiente. Fonte: o Autor.

As cartas de meta são puxadas uma vez a cada duas rodadas, até o final. Com isso, cada jogador vai puxar 3 cartas de meta até o final do jogo. As cartas de meta informam qual o objetivo que cada jogador tem que cumprir para ganhar uma bonificação em pontos no bugômetro. As metas têm um efeito de aprendizagem no jogo, estimulando os jogadores a melhor ordenarem as atividades que são realizadas. A Figura 24 apresenta uma meta com o seu objetivo e a bonificação que ela gera ao cumpri-la.

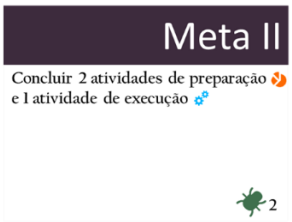


Figura 24. Exemplo de carta de meta. Fonte: o Autor.

Por fim, o jogo tem seus recursos em forma de moedas e esforço por rodada. As moedas são usadas em contratações e apostas nos eventos das cartas de ambiente. Já a esforço é atribuída nas atividades de cada rodada para que as atividades escolhidas sejam cumpridas. A Figura 25 apresenta da direita para a esquerda respectivamente o recurso de esforço, o recurso de moeda do jogo e o marcador do bugômetro.

Figura 25. Recursos do jogo: esforço, moeda e bugômetro. Fonte: o Autor.

O jogo ProTesters é um jogo de tabuleiro para ser jogado em no máximo seis rodadas. Além de terem o material do jogo, os jogadores devem ser capazes de organizar o material do jogo para jogá-lo. Por isso o jogo foi dividido em 3 fases de organização de material: Preparação, Contratações e Execução das rodadas.

5.1.1 Fase 1 - Preparação

A primeira fase de organização do jogo é a de Preparação, na qual os jogadores devem fazer a separação dos componentes e das cartas que serão usadas em seus respectivos espaços. A Figura 26 apresenta o fluxo de execução de ações de preparação que deve ser feita para preparar o ambiente de jogo. Primeiramente, cada jogador irá selecionar sua cor, que pode ser amarelo, vermelho, verde e marrom. Além disso, cada jogador vai ganhar 7 moedas logo no início do jogo, que poderão ser usadas para realizar as contratações de cartas de funcionários.

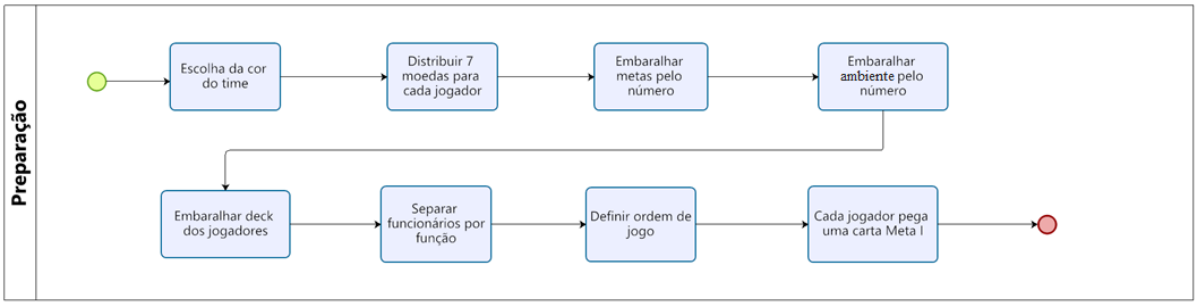


Figura 26. Ações da fase de preparação. Fonte: o Autor.

Com cada time formado, os jogadores irão embaralhar as cartas de ambiente e meta pelo número, por exemplo: cartas Ambiente I serão embaralhadas em conjunto e as cartas Ambiente II serão embaralhadas em outro conjunto. Com isso, existirão 6 conjuntos de cartas embaralhados: 3 conjuntos de cartas de meta e 3 de ambiente. Após isso, as cartas de deck de cada jogador também devem ser embaralhadas.

A fim de melhor organizar a fase de contratações, os jogadores devem separar as cartas de funcionário pelo seu papel. Os jogadores também devem definir a ordem que cada jogador jogará. Por fim, cada jogador puxará uma carta Meta I, que indicará a primeira meta de cada jogado que deverá ser cumprida até o final da segunda

rodada de jogo. A Figura 27 apresenta a organização inicial do jogo presente no manual do jogo.

A Figura 27 apresenta os cartões de 2 jogadores (marrom, ao lado esquerdo da figura, e vermelho, do lado direito da figura) com suas atividades e artefatos ao redor, as cartas de ambiente e metas no centro do tabuleiro, as cartas de funcionário na parte de baixo do tabuleiro, separadas por papel e na parte superior os recursos de moedas e esforço.

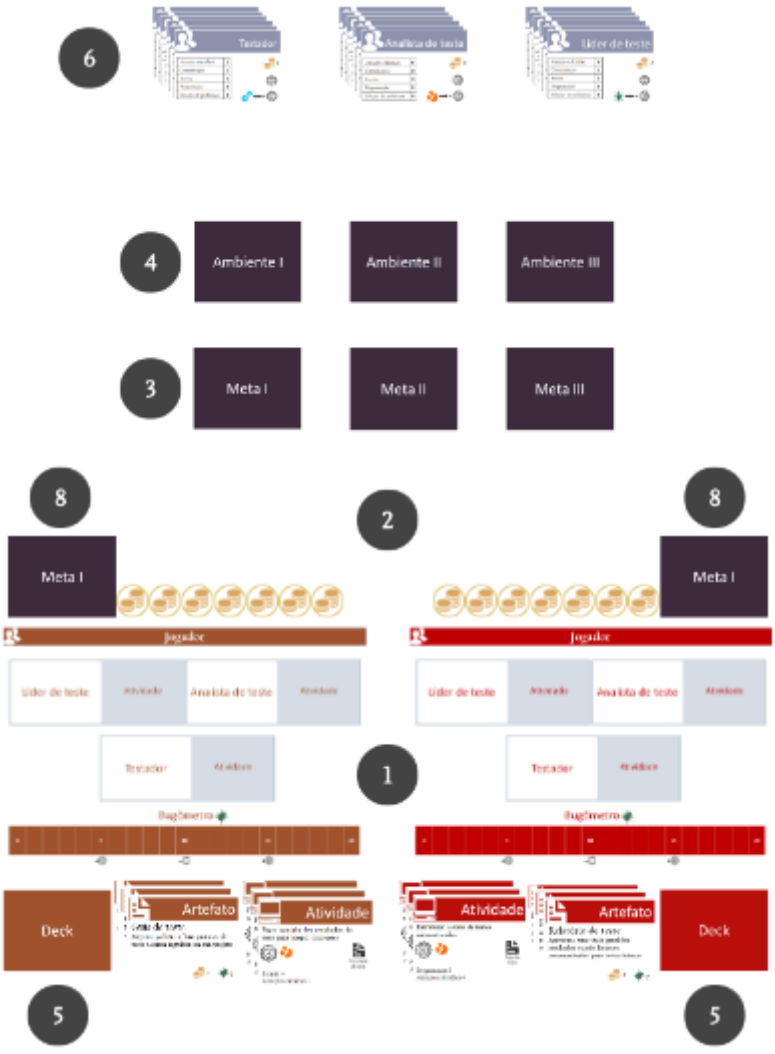


Figura 27. Ambiente inicial de jogo. Fonte: o Autor.

5.1.2 Fase 2 - Contratações

A segunda fase do jogo é a fase de contratações, na qual os jogadores irão selecionar os funcionários que participarão da sua equipe de testes. A Figura 28 apresenta a ordem de ações que devem ser feitas para realizar essa atividade. Cada jogador, na ordem definida na fase anterior, deve selecionar um funcionário por vez até que todos os espaços de funcionário do cartão de jogador estejam preenchidos com uma carta de funcionário. Desta forma, o jogador terá a sua equipe formada.

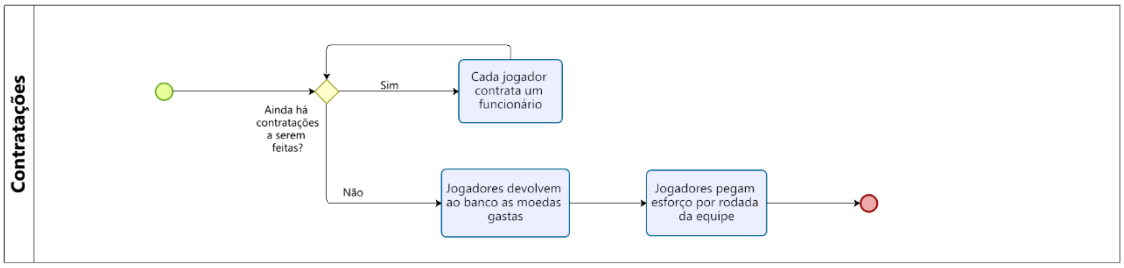


Figura 28. Ações da fase de contratações. Fonte: o Autor.

Após realizar as contratações, cada jogador deve devolver para o banco a quantidade de moedas gastas nas contratações e obter do banco a quantidade de recurso esforço por rodada que a equipe formada por ele tem. Cada carta de funcionário tem um esforço por rodada que é somada. A soma do esforço da equipe de teste informa o quanto de esforço aquela equipe tem e, consequentemente, o quanto de esforço o jogador pode gastar por rodada para executar as atividades de teste.

5.1.3 Fase 3 - Execução das rodadas

A última fase do jogo é a execução do jogo em si. A Figura 29 apresenta o fluxo de ações, que representa o passo-a-passo de cada rodada do jogo. Logo no início do jogo os jogadores devem iniciar com cinco cartas do seu *deck* na mão e puxar mais uma a cada início de rodada. Para a primeira rodada, cada jogador já possui uma carta Meta I, que foi puxada na fase de preparação. Já para a terceira e quinta rodadas, os jogadores devem descartar a meta que possuir e puxar uma nova meta: na terceira rodada puxam a Meta II e na quinta rodada é puxam a Meta III.

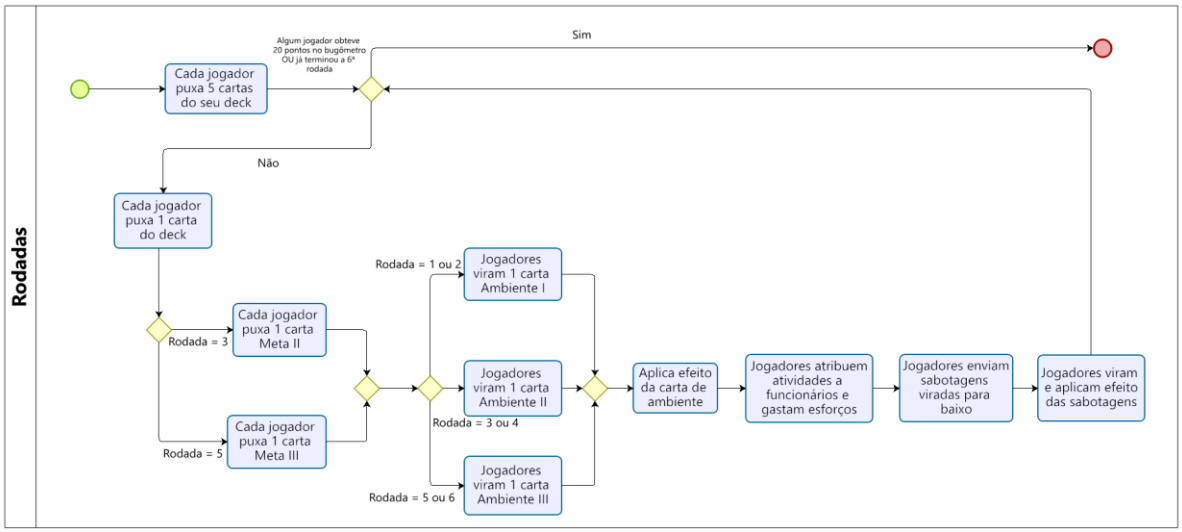


Figura 29. Ações da fase de execução das rodadas. Fonte: o Autor.

Após atualizarem as metas, os jogadores viram uma carta de ambiente, que tem influência para todos os jogadores. As cartas de ambiente dependem da rodada do jogo: na primeira e segunda rodadas são viradas cartas de Ambiente I, na terceira e quarta, são viradas cartas de Ambiente II e na quinta e sexta, as de Ambiente III. Após virar a carta de ambiente da rodada, o efeito do ambiente é aplicado.

Após a resolução das cartas de ambiente, os jogadores irão executar o processo de teste, fazendo a atribuição das atividades para os funcionários e gastando o esforço necessário para cumprir as atividades selecionadas para a rodada. Para isso, os jogadores devem ter em vista tanto o esforço que eles possuem, os bônus de esforço dos seus funcionários, os níveis de habilidade requeridos por cada atividade para serem cumpridos e os efeitos das cartas de ambiente.

Após atribuir as atividades, os jogadores ainda não terminam a rodada, pois ainda têm que enviar sabotagens para outros jogadores. Cada jogador pode enviar quantas sabotagens quiser para outros jogadores por rodada. Para isso eles devem ter uma sabotagem em sua mão e jogar a sabotagem para outro jogador, com a face voltada para baixo. Quando todos tiverem enviado sabotagens, as cartas são viradas e o efeito de cada sabotagem é aplicado no jogador que a recebeu.

Após as sabotagens, a rodada termina. As atividades executadas pelos jogadores são cumpridas e com isso o artefato da atividade é produzido pelo jogador e os bônus gerados pelo artefato são obtidos. Além disso, o esforço gasto pelo jogador é retornado para sua mão para ser usada na próxima rodada e os jogadores podem

fazer novas contratações. Para um melhor entendimento do funcionamento do jogo e do seu funcionamento, o manual do jogo é apresentado no Apêndice C, onde suas regras e a dinâmica do jogo são melhor apresentados. O manual do jogo assim como o material do jogo também estão disponíveis no Google Drive[2].

5.1.4 Cumprimento das competências educacionais

Em relação as competências educacionais que foram definidas no Capítulo 4. Em relação a competência “Organizar as principais atividades que compõem o processo de testes de software” o jogo traz a mecânica de seguir metas para guiar as atividades que devem ser realizadas pelo jogador ao longo do jogo. As cartas de metas do jogo são divididas em 3 decks: Meta I, Meta II e Meta III. Cada meta pede que o jogador atinja algum objetivo relativo ao processo de teste dentro de duas rodadas, seja obtendo artefatos ou concluindo atividades, isso permite que o jogador entenda como funciona o ordenamento das atividades do processo de testes. Os jogadores começam com a Meta I, que é a meta de início de jogo. As metas são atualizadas a cada duas rodadas para uma meta acima, indicando o andamento do processo de teste. A Meta I é a primeira que os jogadores têm, e elas pedem para que sejam feitas atividades de planejamento de testes ou obtidos artefatos iniciais do processo, como o “Plano de teste” e o “Guia de testes”. Já a Meta II é a meta do meio do processo e pedem que atividades de preparação sejam realizadas ou que artefatos como a “Suíte de teste” e “Script de teste” sejam obtidos. Já a Meta III é a meta da parte final do processo e pedem que atividades de execução sejam realizadas ou artefatos tais como “Log de teste” e “Relatório de teste” sejam obtidos.

Em relação a competência “Distinguir os principais artefatos de teste para as atividades em que devem ser produzidos” há duas mecânicas a que abordam. A primeira mecânica que trata desta competência é a produção de artefatos por determinadas atividades. Cada carta de atividade especifica qual artefato que ela produz ao ser executada, que é relativo ao que cada atividade produz em um processo real. Desta forma o material do jogo e a mecânica de produção de artefato pelas atividades do jogo ajudam que os jogadores entendam quais atividades produzem

[2] Disponível em: https://drive.google.com/drive/folders/1NYOMmlShcrD10c8YCAv1CTk2YZFSYDOZ
Acesso 09/01/2022

quais artefatos. A outra mecânica é o *design* das cartas de artefato que apresentam uma descrição do artefato que está sendo representado pela carta. Desta forma o jogo ajuda o jogador a entender o que representa cada artefato e o que ele realmente é dentro do processo de teste.

A competência “Organizar as responsabilidades de cada papel do processo de teste” também é tratada por duas mecânicas dentro do jogo. A primeira mecânica que trata das responsabilidades de cada papel do processo de teste é a *tag* das atividades. Cada carta de atividade do jogo possui uma *tag*, indicando qual é o tipo daquela atividade, seja uma atividade de planejamento, preparação ou execução. Além disso, cada carta de papel possui um bônus de esforço fornecido por realizar atividades de um certo tipo, o que indica que cada papel é responsável para realizar atividades de um certo tipo. Os funcionários do tipo “Líder de teste” são responsáveis por atividades de planejamento e por isso recebem um esforço a mais na rodada ao realizar atividades desse tipo. A mesma coisa acontece com os funcionários do tipo “Analista de teste” ao realizarem atividades de preparação e “Testador” ao realizarem atividades de execução.

Uma outra mecânica que aborda a competência “Organizar as responsabilidades de cada papel do processo de teste” é os níveis de habilidades dos funcionários. Cada carta de papel tem níveis de habilidades diferentes de “Atenção a detalhes”, “Comunicação”, “Escrita”, “Programação” e “Solução de problemas”. Dependendo do papel que está sendo representado na carta, aquela carta terá melhores habilidades em tipos específicos, por exemplo: funcionários do tipo “Testador” têm mais habilidades de programação e atenção a detalhes, já funcionários do tipo “Líder de teste” têm mais habilidades de comunicação e escrita. Cada atividade requer que o funcionário que irá realizá-la tenha certos níveis de habilidades específicos que estão relacionadas às habilidades do papel que é responsável por executá-la.

Já a competência “Analisar algumas possíveis situações/problemas que comumente ocorrem durante o processo de teste” é tratada por uma mecânica apenas que é a de carta de solução de problemas. As cartas de ambiente são divididas em 3 *decks*, cada um contendo 2 cartas de problemas, totalizando 6 problemas. Para solucionar tais problemas, o *deck* de cada jogador possui 6 cartas de solução, cada

uma sendo relacionada a um problema específico. Ao virar uma carta de problema no ambiente, os jogadores devem ter puxado para sua mão uma carta de solução do *deck* para jogá-la e solucionar o problema. Como auxílio, cada carta de problema possui uma *tag*, igual a das atividades, e a solução para aquele problema possui uma *tag* igual, que liga o problema a solução. Além disso, a descrição do problema e da solução devem estar relacionadas para que a solução usada pelo jogador possa realmente solucionar o problema levantado

5.2 Versão utilizada no *Playtesting 1*

A versão do jogo utilizada no *Playtesting 1* era bem simples e similar à versão final do jogo. O jogo apresentava, como ainda apresenta em sua versão final, o cartão de jogador, os artefatos, as atividades e os papéis. A jogabilidade também era feita em seis rodadas e com a utilização de cartas e recursos para simular o processo de teste. Além disso, havia também o papel do desenvolvedor, que futuramente foi eliminado, uma vez que não era o foco do ensino do processo de teste. Havia mais pontos que poderiam ser obtidos pelos jogadores, além dos pontos de vitória, que continuaram até a versão final.

O cartão do jogador tinha espaço para atribuir atividades para 2 funcionários, alocar *bugs* para que o desenvolvedor gerasse dinheiro e 4 medidores de pontos, como apresentados na Figura 30: pontos de vitória (primeira coluna), pontos de planejamento (segunda coluna), pontos de preparação (terceira coluna) e pontos de execução (quarta coluna). Cada coluna tem uma pontuação máxima de 20 pontos, e com isso, o jogador que chegasse a 20 pontos no marcador de vitória seria o ganhador do jogo. Caso ninguém chegasse a essa marca, o jogador com maior pontuação no marcador de vitória seria o vencedor.

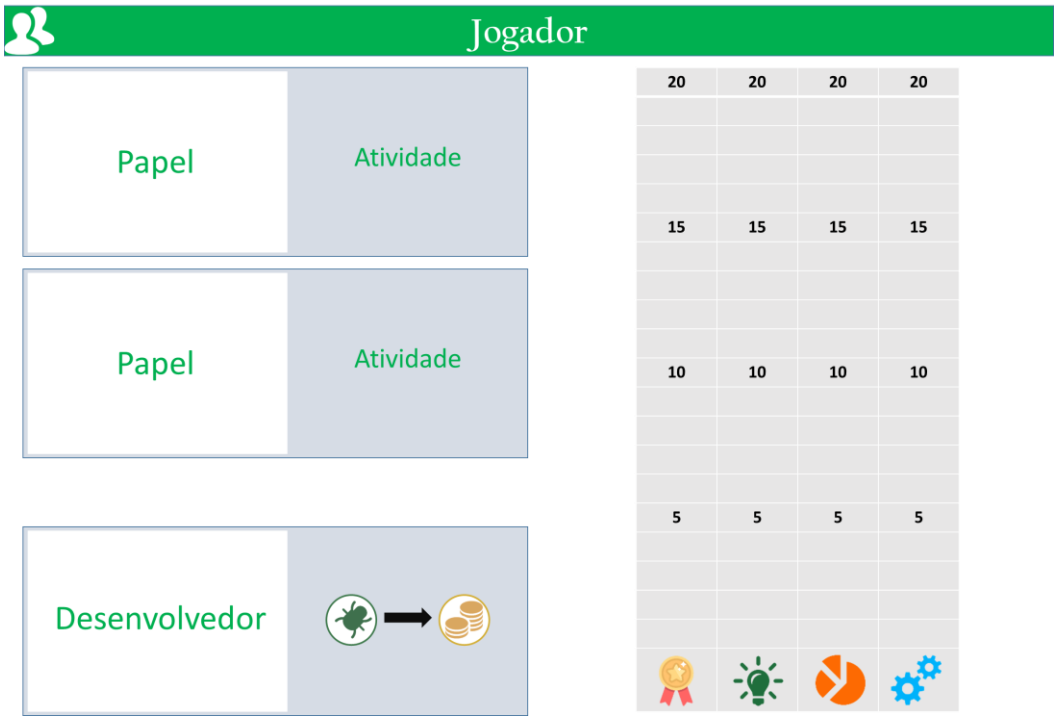


Figura 30. Cartão do jogador verde. Fonte: o Autor.

Cada atividade gera pontos de algum tipo, que foi definido por quem realiza aquela atividade. Por exemplo: atividades que geram planejamento são de responsabilidade do papel Líder de teste e atividades que geram pontos de execução são de responsabilidade do Testador. A pontuação gerada por cada atividade é incrementada nos marcadores de pontuação do cartão do jogador.

Cada atividade que era realizada gerava bugs, que era um dos recursos do jogo. Isso era utilizado para demonstrar que ao se realizar mais atividades do processo, maior era a quantidade de bugs encontrados no produto testado. Esses bugs poderiam ser posteriormente utilizados com os cartões Desenvolvedor para que fossem transformados em moedas. A Figura 31 apresenta duas cartas de atividade e seus elementos.

Figura 31. Cartas de atividade. Fonte: o Autor.

Os pontos gerados pela realização de atividades e somados nos marcadores de pontuação do cartão do jogador poderiam ser posteriormente utilizados para comprar Artefatos. Apesar de não haver um link explícito entre a atividade que produz cada artefato, há um link implícito que é visto pela quantidade de pontos que cada artefato exige para ser comprado. Por exemplo: o artefato Plano de teste exige 10 pontos de planejamento para ser comprado e as atividades que o produzem no processo real de teste, que são “Identificar eventos e artefatos que influenciarão nos testes” e “Fazer acordo sobre os objetivos e entregáveis das interações” geram respectivamente 3 e 7 pontos de planejamento, totalizando 10 pontos.

Apesar de haver um link implícito, era possível comprar artefatos sem ter realizado as atividades que realmente o geram no processo real de teste de software. Por isso, a questão dos pontos de Planejamento, Preparação e Execução que são utilizados para comprar os artefatos foram trocados e foi utilizado um link explícito entre os artefatos e atividades que são utilizadas no jogo. A Figura 32 apresenta dois exemplos de cartas de artefato, com o artefato que está sendo representado e a pontuação necessária para comprá-la e a pontuação de vitória que ela fornece.

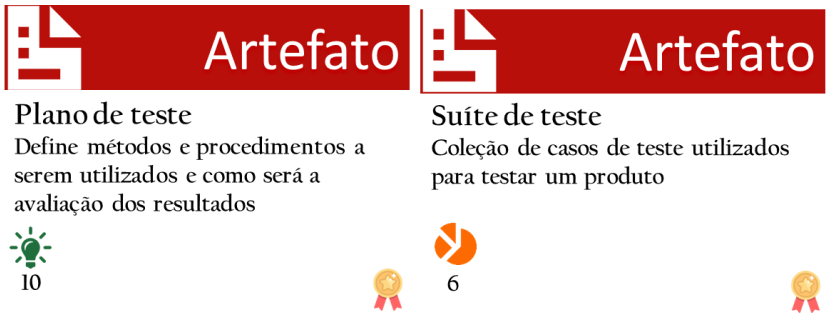


Figura 32. Cartas de artefato. Fonte: o Autor.

O principal objetivo das cartas Objetivo era o de fazer uma sequência das atividades que eram realizadas no jogo, o que deveria incentivar os jogadores a executarem corretamente o processo de teste. Dessa forma, se os jogadores seguissem a ordem de cada carta de Objetivo, eles seguiriam uma boa execução do processo de teste.

Havia cartas a serem puxadas pelos jogadores: Objetivo I, II e III. As três cartas viradas eram objetivo de todos os jogadores, ou seja, eram objetivos gerais, e

deveriam ser cumpridas em sequência: primeiro se cumpre o Objetivo I para depois poder cumprir o Objetivo II, e assim por diante. Entretanto essa forma de jogo se mostrou falha, uma vez que tornava o jogo de cada jogador muito igual, uma vez que os objetivos eram para todos e permitia que os jogadores pudessem fazer suas atividades desordenada. Por conta disso, a mecânica de puxar as 3 cartas Objetivo logo no início do jogo foi trocada nas versões futuras. A Figura 33 apresenta duas cartas de objetivo II, apresentando seu efeito e os pontos de vitória que são ganhos ao completar cada objetivo.

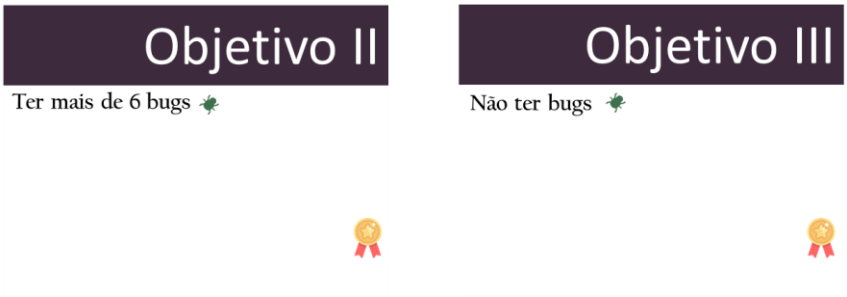


Figura 33. Cartas de artefato. Fonte: o Autor.

O jogo apresentava três papéis relacionados ao processo de teste de software: Líder de teste, Analista de teste e Testador e também um papel extra que era o Desenvolvedor. Os três primeiros papeis eram utilizados explicitamente para executar as atividades do processo de teste, ou seja, estavam relacionados às cartas de Atividade. Ao realizar uma atividade do processo de teste, era gerado uma quantidade de bugs, que poderia ser utilizada futuramente pelo Desenvolvedor para transformar bugs em dinheiro, simulando a correção dos bugs encontrados. A Figura 34 apresenta duas cartas de funcionário, um desenvolvedor, que transforma bugs em moedas, e um testador, que tem habilidades e esforço para realizar atividades.

Figura 34. Cartas de funcionários. Fonte: o Autor.

5.3 Versão utilizada no *Playtesting 2*

A versão do jogo utilizada neste *playtesting* era parecida com a versão utilizada no primeiro. O jogo apresentava o cartão de jogador, os artefatos, atividades e papéis e a jogabilidade feita em seis rodadas e com a utilização de cartas e recursos para simular o processo de teste. Uma das alterações que a segunda versão do jogo tem em relação a primeira foi a utilização de um deck de carta para cada jogador, contendo sabotagens e soluções para problemas que podem aparecer nas cartas de Ambiente, como apresentado na *Figura 35*, onde é mostrado a solução para um problema.

Cada jogador começa com 3 cartas do deck na mão e ao passar das rodadas puxa 1 carta. No total existem 8 cartas no deck dos jogadores: 6 cartas Solução que são iguais para todos os jogadores e 2 cartas Sabotagem que são únicas para cada jogador. As cartas de Sabotagem são usadas em outros jogadores antes do término da rodada e as de solução são usadas para evitar o efeito negativo de cartas de Ambiente e ganhar bônus em alguma pontuação do jogo e pontos de vitória. A Figura 35 apresenta uma carta de ambiente problema e ao lado a carta de solução que anula o problema apresentado.

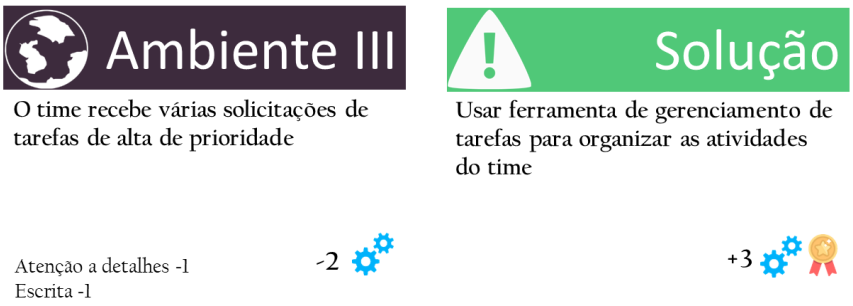


Figura 35. Cartas de ambiente problema e carta de solução relacionada. Fonte: o Autor.

O cartão do jogador continuou tendo espaço para atribuir atividades para 2 funcionários, alocar bugs para que o desenvolvedor gerasse dinheiro e 4 medidores de pontos, como na primeira versão do jogo. Cada coluna continuou tendo pontuação máxima de 20 pontos e as regras de vitória do jogo continuaram como na primeira versão.

Cada atividade continua gerando pontos de algum tipo, que foi definido por quem realiza aquela atividade. Houve uma mudança nessa versão do jogo em relação a primeira em termos de benefício para a atribuição correta de atividades para os papéis. Na segunda versão do jogo ao atribuir corretamente um tipo de atividade para o papel correto, o jogador ganharia mais pontos dos marcadores de pontuação do jogo. Por exemplo, na Figura 36, a atribuição da atividade apresentara para o Analista renderia mais 2 pontos de Preparação para o jogador, além dos 4 pontos que a atividade naturalmente produz.

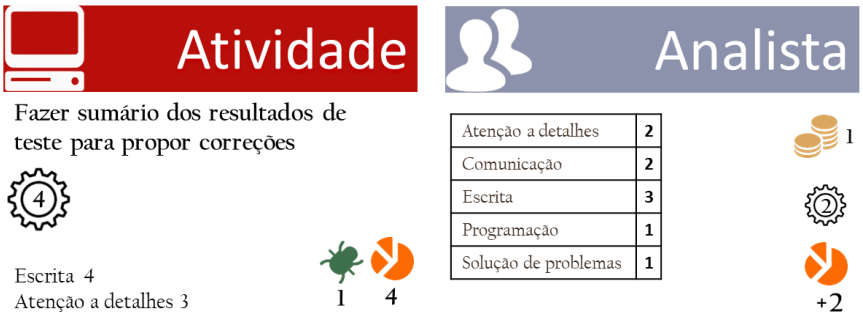


Figura 36. Cartas de ambiente problema e carta de solução relacionada. Fonte: o Autor.

Cada atividade que é realizada continua gerando bugs, mas em quantidade menor do que na primeira versão do jogo, pois foi uma recomendação dos participantes do primeiro *playtesting*. Esses bugs poderiam ser posteriormente utilizados com os cartões Desenvolvedor para que fossem transformados em moedas.

Os pontos gerados pela realização de atividades e somados nos marcadores de pontuação do cartão do jogador poderiam ser posteriormente utilizados para comprar Artefatos. Apesar de no primeiro *playtesting* os jogadores já terem recomendado criar um link explícito entre as cartas Atividade e Artefatos, o autor do trabalho não conseguiu pensar em uma mecânica ainda para esta versão do jogo que atendesse a esse pedido.

O jogo continuou tendo as cartas de Objetivo I, II e III que deveriam ser puxadas agora por cada jogador individualmente, ou seja, elas eram únicas para cada jogador, o que trouxa uma mudança na jogabilidade do jogo. Como mudança, também foi aumentado de 1 para 2 pontos de vitória fornecidos pelas cartas Objetivo. A Figura 37 apresenta as novas cartas de objetivo, fornecendo mais pontos de vitória do que na

primeira versão. Entretanto, todas as cartas de Objetivo continuaram sendo puxadas logo no início do jogo pelos jogadores, assim como na primeira versão.

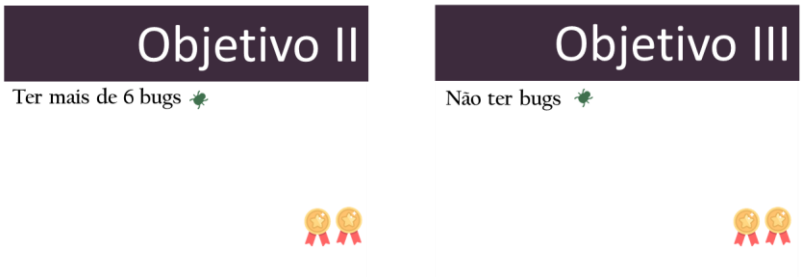


Figura 37. Cartas de ambiente problema e carta de solução relacionada. Fonte: o Autor.

O jogo continuou contemplando os papéis do Líder de teste, Analista de teste e Testador e do Desenvolvedor. Os três primeiros sendo utilizados para executar as atividades do processo de teste, e o Desenvolvedor para transformar bugs em dinheiro, simulando a correção dos bugs encontrados. Como apresentado anteriormente, foi alterado apenas um bônus em cada carta de papel para fornecer maior pontuação nos marcadores do jogo ao jogador que atribuir corretamente atividades relacionadas.

5.4 Versão utilizada no *Playtesting 3*

A versão do jogo utilizada no terceiro *playtesting* sofreu várias alterações no sentido de jogabilidade, o que facilitou o jogo e o enfoque em aspectos de aprendizagem do conteúdo. Essa versão do jogo continuou sendo feita em 6 rodadas e contou com alterações o cartão de jogador, como pode ser visto na Figura 38.

Com as discussões feitas no segundo *playtesting*, o autor considerou a remoção dos outros marcadores de jogo, como pontos Planejamento, Preparação e Execução. A influência desses marcadores ainda continua no jogo, mas de uma forma mais simples e menos importante para a jogabilidade. Com isso, o único marcador que continuou foi o de Vitória, além dos espaços para a contratação dos funcionários.

Figura 38. Cartão do jogador azul. Fonte: o Autor.

Como já afirmado, os pontos de Preparação, Planejamento e Execução ainda continuaram no jogo, mas de uma forma mais simples, apenas como uma classificação de atividades e problemas que guiarão os jogadores. Na Figura 39 é possível ver que as *tags* de atividade de execução ainda estão sendo usadas nas cartas Atividade. Elas são usadas para marcar qual o tipo da atividade que está sendo descrita.

Figura 39. Cartas de funcionário e atividade com utilização de tags. Fonte: o Autor.

Ao atribuir atividades do tipo Execução para um funcionário Testador, como na Figura acima, o jogador ganhará mais um ponto de vitória. O mesmo acontece ao atribuir atividades de Planejamento para o Líder e atividades de Preparação para o Analista. Dessa forma, essas *tags* passaram a ser usadas para guiar os jogadores e

estimular uma melhor atribuição de atividades. A mesma coisa ocorreu com as cartas de Solução e Ambiente.

Além disso, as cartas de deck foram aumentadas de 8 para 16 cartas e com isso existiam 6 cartas Solução e 10 Sabotagens. Com isso era esperado estimular a competição e a diversão entre os jogadores.

Uma das mudanças requeridas nos *playtestings* anteriores era de criar uma relação mais clara entre Atividade e Artefato. Com isso, o autor do trabalho criou uma mecânica de liberação e compra de artefatos. Ao realizar uma Atividade, a carta da atividade informa o artefato que ela libera para o jogador comprar, como na Figura 40, onde a carta Atividade informa no seu canto direito o artefato que ela libera. Ao liberar a carta, o jogador pode gastar uma quantidade de moedas para realmente obtê-la e ganhar pontos de vitória e cumprir seus objetivos.

Figura 40. Cartas de funcionário e atividade com utilização de tags. Fonte: o Autor.

Além disso, como requisições do *playtesting* anterior, houve um incremento na quantidade de esforço necessária para cumprir cada tarefa e houve uma troca de cartas Objetivo consideradas não representativas do processo de teste. Como questão de nomenclatura as cartas Objetivo tiveram seu nome alterado para Meta, mas elas continuam tendo o mesmo funcionamento das cartas Objetivo das versões anteriores.

5.5 Considerações sobre o Capítulo

Este capítulo apresentou o ProTesters, jogo elaborado no âmbito deste projeto de pesquisa. Foi apresentado todo o material do jogo, suas cartas, recursos e explicado como cada componente funciona e interage com o jogo. Também foi explicado como as mecânicas do ProTesters abordam as competências de

aprendizagem propostas pelo jogo. Como o jogo foi feito com várias avaliações nos *playtestings* e correções, este capítulo também apresentou uma visão geral de como era a versão do jogo utilizada em cada *playtesting.* Essa visão geral facilita as discussões de cada *playtesting* que serão apresentadas no Capítulo 6, além de mostrar como ocorreu a evolução do jogo ao longo das iterações de avaliação e correção no seu desenvolvimento.

CAPÍTULO 6 - AVALIAÇÃO DA PROPOSTA

“Quando você quer alguma coisa, todo o universo conspira para que você realize o seu desejo.”

Paulo Coelho, O alquimista

Neste capítulo serão apresentadas as avaliações realizadas no jogo. Como afirmado anteriormente, os *playtestings* são avaliações *Ex ante* e tiveram o objetivo de avaliar o jogo em relação à sua jogabilidade e aprendizagem percebida por jogadores de tabuleiro. Com essas avaliações foi possível observar as discussões ocorridas entre os jogadores e fazer melhorias antes de aplicar o jogo com seu público-alvo, que são estudantes de graduação em Computação. Depois destas rodadas, já com o jogo ajustado, foi realizado o experimento *Ex post* como avaliação final, cujo objetivo foi verificar os efeitos motivacionais e educacionais promovidos pelo jogo. Nas seções seguintes serão apresentados os *playtestings* e o experimento, na ordem em que foram realizados.

6.1 *Playtesting* 1

O primeiro *playtesting* realizado com a primeira versão do jogo foi realizado no dia 24/06/2021. O *playtesting* teve duração de aproximadamente uma hora e meia sendo contados desde o momento inicial de apresentação do jogo e suas regras até a finalização das discussões sobre o jogo. Este *playtesting* contou com três participantes que foram escolhidos por frequentemente jogarem jogos de tabuleiro ou jogos de cartas e por serem estudantes ou ex-estudantes do curso de Engenharia de Software. O objetivo da escolha desses participantes com experiência em jogos e estudantes da área de Computação era para permitir melhores discussões e conselhos sobre o jogo, tanto do ponto de vista de suas regras, design e mecânicas, como também do ponto de vista educacional em relação a testes de software.

Todos os *playtestings* foram gravados com autorização dos participantes. O autor apresentou a visão geral do que se tratava o *playtesting* e como ele ocorreria, explicando que o objetivo final era a discussão e aprimoramento do jogo.

O jogo foi apresentado aos três jogadores pelo autor do trabalho, que também fez algumas simulações de situações de jogo para que os jogadores entendessem melhor as mecânicas de jogo. Após a apresentação, os jogadores tiveram a experiência com o jogo. O autor do trabalho, como moderador, ficou o tempo inteiro do jogo fazendo observações sobre comentários e interações dos jogadores com o jogo, além de auxiliar os jogadores.

Por restrições de tempo de alguns dos participantes, o jogo não foi jogado até o final, mas apenas até a terceira rodada. Apesar disso, foi possível ver os diversos mecanismos e artefatos do jogo em ação. No final, o autor do trabalho preparou um guia de discussão com 6 questões, que é apresentado no Apêndice A, que foram repetidas em todos os *playtestings*, o que permitiu diversas discussões, que serão apresentadas a seguir.

6.1.1 Principais discussões

Respondendo a primeira pergunta do guia de discussão, o Participante 3 afirmou que o jogo promove a competividade entre os jogadores, mas no sentido de competição por melhores recursos humanos, no caso, cartas de funcionários. Já o Participante 1, apesar de ter perdido uma aposta que permitia com que outro trocasse um de seus funcionários com ele, ele não sentiu o efeito dessa perda. Isso porque a troca não foi tão danosa para ele porque o jogador vencedor da aposta fez uma troca ruim, que não prejudicou tanto o Participante 1.

> Participante 3: ‘Ele promove a competitividade, especialmente em relação aos “Papeizinhos”, os atores (funcionários)’
>
> Participante 1: ‘As apostas eu não senti muito, não ... Por exemplo, não sei se no futuro isso iria me prejudicar mais, mas eu perdi a aposta e não senti que isso me prejudicou tanto’

Durante as discussões sobre um dos objetivos a serem cumpridos, os jogadores entenderam que no formato em que o jogo estava jamais seria possível atingir o objetivo proposto. O autor, entendendo o problema apresentado, tomou como mudança a ser realizada a diminuição da quantidade de bugs gerados por atividade e aumentar premiação das cartas Objetivo.

> Participante 1: ‘Toda atividade sua vai gerar bug, ou seja, você sempre vai estocar bug’
>
> Participante 2: ‘Esse objetivo não dá para atingir. E só dá um (ponto de vitória)? Era para dar mais. Aumenta a premiação’
>
> Autor: ‘Então tenho que pensar talvez nas atividades que gerem bugs, né? Porque nem toda atividade realmente gera bug*’*

O Participante 1 reclamou que era difícil saber o valor da habilidade de cada funcionário quando suas habilidades eram exibidas em uma tabela e marcadas com célula dourada no valor atual (ver Figura 4-6).

> Participante 1: ‘Eu acho que precisa colocar só o valor do que o cara tem (habilidades).

Uma das reclamações dos jogadores em relação ao *design* do jogo na versão utilizada nesse *playtesting* foi que os fundos de todas as cartas eram iguais, o que dificultava saber o que era carta de ‘Ambiente’ e ‘Objetivo’. Nas versões futuras isso foi alterado para dizer especificamente qual era o tipo da carta.

> *Participante 3: ‘Tem que mudar o fundo das cartas, porque eu não tava sabendo diferenciar nenhum’*

O Participante 1 também achou que na versão utilizada no primeiro *playtesting* era difícil saber qual era o tipo de artefato que estava sendo representado nas cartas de Artefato. A solução proposta foi a de aumentar a fonte do título do artefato e diminuir a explicação do artefato.

> *Participante 1: 'Outra coisa: quando tem por exemplo Artefato – relatório de teste. Aí, tipo assim, tem a descrição do relatório de teste, mas eu queria que o nome relatório de teste fosse mais destacado. Tipo: em negrito ou maior'*

Quando falado que havia funcionários com esforço 3, o Participante 3 que havia contratado um desses funcionários reclamou que esse esforço a mais não havia ajudado muito, pois todas as atividades requisitavam um número par de esforço. Como mudança identificada pelo autor foi a de aumentar o esforço dos funcionários mais caros de 3 para 4.

> *Participante 3: 'Ei, mas não serviu para muita coisa, não, viu? Porque só tem atividade com (esforço necessário) 4 e 2'*

Quando perguntados sobre se o jogo contribuiu para o ensino do processo de teste, os jogadores apontaram que não perceberam melhora no ensino de teste. Alguns até explicaram que o jogo poderia se tratar do processo de desenvolvimento, uma vez que trazia o papel do Desenvolvedor. Uma das recomendações, como apresentadas pelo Participante 3 foi a de criar *links* mais fortes entre atividades e artefatos que são gerados.

> *Participante 3: 'Eu acho que o jogo reforça principalmente os papéis e a importância deles para cumprir as atividades. Mas em relação ao processo de teste eu não sei'*

> *Participante 3: 'Eu acho que para reforçar o teste poderia não ter mais atividade solta, entendeu? Poderia ser atividades que levem a outras atividades. E eu acho que os artefatos, por exemplo, o cara faz uma atividade de JUnit e poderia ter ... um artefato específico. Então poderia ser: você faz um conjunto de atividades e ganha um artefato'*

Nessa versão do jogo, as cartas Objetivo eram gerais. E isso foi visto como algo ruim pelos jogadores, pois todos eles jogaram de uma forma parecida. Como

recomendação, o Participante 2 propôs que cada jogador tivesse seus próprios objetivos uma vez que cada um seria uma empresa diferente.

> *Participante 2: 'Eu acho mais legal separar os objetivos para cada um. Porque é como se cada um fosse uma empresa mesmo'*

6.1.2 Visão geral

O Quadro 7 apresenta a visão geral do primeiro *playtesting*, apresentando a descrição dos participantes, data, duração da avaliação e as principais melhorias (*insights*) obtidas com as discussões.

Quadro 7. Visão geral das informações do *Playtesting 1*. Fonte: o Autor.

Informações gerais	
Data	24/06/2021
Duração	1 hora e 28 minutos
Participantes	- Participante 1: Ex-estudante de graduação em Engenharia de Software com experiência em jogos de cartas e de tabuleiro - Participante 2: Estudante de graduação em Engenharia de Software com experiência em jogos de cartas e de tabuleiro - Participante 3: Estudante de graduação em Engenharia de Software com experiência em jogos de cartas e de tabuleiro
Insights	- Balancear a quantidade de bugs que são gerados por atividade - Aumentar premiação das cartas Objetivo - Mudar o padrão de apresentação dos níveis de habilidade das cartas de funcionários - Mudar o fundo das cartas de Objetivo e Ambiente para que os jogadores saibam qual é o tipo da carta sem virá-la - Aumentar o destaque para cada tipo de artefato, para que os jogadores tenham mais facilidade em descobrir o artefato que está sendo representado - Cartas de funcionário que adicionam 3 de esforço deveriam adicionar na verdade um número par de esforço - Deve haver um link mais forte entre atividades e artefatos - Cada jogador deve receber seus próprios objetivos que devem ser únicos

6.2 *Playtesting* 2

O *Playtesting 2* foi realizado com a segunda versão do jogo, após as modificações feitas com os aprendizados do *Playtesting 1* e foi realizado no dia 08/07/2021. O *playtesting* teve duração de aproximadamente duas horas e quinze minutos sendo contados desde o momento inicial de apresentação do jogo e suas regras até a finalização das discussões sobre o jogo. Este *playtesting* contou com três participantes, dois participantes que já haviam participado do primeiro *playtesting* e um terceiro participante que não participou do primeiro *playtesting*, não jogava com frequência jogos de tabuleiro e não era estudante da área de computação. A escolha por convidar dois participantes do primeiro *playtesting* era para novamente permitir discussões sobre o jogo e verificar se as melhorias identificadas nas discussões do primeiro *playtesting* tinham sido tratadas. A escolha por convidar um participante novo, sem experiência com jogos de tabuleiro ou de cartas e nem da área de computação foi para verificar a apreensibilidade do jogo com um jogador sem muita experiência com jogos e também a capacidade de um jogador sem conhecimento sobre computação teria para aprender sobre o conteúdo tratado no jogo.

Inicialmente o autor e moderador do *playtesting* pediu a permissão dos jogadores para fazer a gravação. O autor apresentou a visão geral do que se tratava o *playtesting* e como ele ocorreria, explicando que o objetivo final era a discussão e aprimoramento do jogo.

O jogo foi apresentado aos três jogadores pelo autor do trabalho, que também fez algumas simulações de situações de jogo para que os jogadores entendessem melhor as mecânicas de jogo. Após a apresentação, os jogadores tiveram a experiência com o jogo. O autor do trabalho, como moderador, ficou o tempo inteiro do jogo fazendo observações sobre comentários e interações dos jogadores com o jogo além de auxiliar os jogadores.

A versão do jogo jogada nesse *playtesting* foi considerada muito fácil pelos participantes e por isso o jogo terminou ainda na quarta rodada do jogo, quando um dos participantes atingiu os 20 pontos de vitória necessários. Após o término do jogo, o autor do trabalho utilizou o guia de discussão, que é apresentado no Apêndice A, para realizar as discussões, que serão apresentadas a seguir.

6.2.1 Principais discussões

Uma das reclamações que os jogadores tiveram em relação às novas mecânicas inseridas nessa versão do jogo foi sobre a quantidade de cartas Solução que havia no *deck*. Nessa versão havia 8 cartas em cada *deck* sendo 6 cartas de solução, ou seja, uma quantidade muito grande de solução em relação ao total. A solução proposta pelos jogadores foi a de inserir mais cartas de sabotagem, pois isso tornaria o jogo mais competitivo e com maior variedade de cartas e eventos.

> *Participante 2: 'Parece que toda vez que você puxa um ambiente ... sempre alguém vai ter uma solução'*
>
> *Participante 2: 'Coloca sabotagem pra perder pontos de medalha (ponto de vitória) ... Porque senão, fica fácil demais assim'*
>
> *Participante 2: 'Eu tô achando a sabotagem mó daora. Eu botaria bem mais''*

Um dos participantes reclamou que as cartas de Desenvolvedor estavam muito mais fortes do que as dos demais papéis. Isso porque sempre são gerados *bugs* nas atividades e sempre os desenvolvedores poderiam usar aqueles *bugs* para gerar moedas e com isso seria possível fazer melhores contratações e ganhar vantagens. A solução proposta pelo Participante 1 para enfraquecer as cartas de Desenvolvedor foi a de requerer uma quantidade de esforço por rodada para a carta Desenvolvedor pudesse ser usada.

> *Participante 1: 'Outra coisa: eu acho que o desenvolvedor deveria gastar 'produtividade' ... para dar uma balanceada na 'produtividade'. Porque ele não custa nada, então eu posso só ficar "farmando", porque bug não custa nada'*

O Participante 1 também deu a ideia de estabelecer os espaços de cada funcionário do time logo nos cartões de jogador. Nessa versão o cartão jogador tem apenas 2 espaços disponíveis para atribuição de atividades, apesar de cada jogador poder contratar 3 funcionários de teste e o Desenvolvedor.

> *Participante 1: 'Uma outra coisa que tu podia fazer era "liberar" 3 lugares aqui. Tipo: já botava o lugar do líder, do desenvolvedor'*

Quando perguntados sobre se aprenderam algo sobre o processo de teste, o Participante 3 não sentiu que aprendeu algo com o jogo. Nessa versão havia alguns objetivos não representativos do processo de teste, como ganhar moedas e ter um time mais caro. Curiosamente, o Participante 3 puxou duas cartas Objetivo que não eram relativas ao processo de teste, o que o autor considerou que afetou o aprendizado desse participante com o jogo. Como mudança a ser feita, foi pensado em remover os Objetivo genéricos e inserir novos, mais representativos do processo de testes em si.

> *Participante 3: 'Sinceramente, não aprendi'*
>
> *Autor: 'É porque tuas metas também ... tinha que ter umas metas mais representativas para ela"*

No segundo *playtesting,* o Participante 2, que também participou do primeiro *playtesting,* venceu e comentou que achou a versão anterior do jogo melhor e mais difícil. O Participante 1 concordou e comentou que o esforço (chamado pelo participante de "produtividade") por rodada aumentou muito da primeira versão para a que foi usada neste *playtesting*. Anteriormente, uma carta de funcionário tinha no máximo 3 de esforço e nessa tinha vários funcionários com 4, o que facilitou demais o jogo. Como solução, o autor identificou que seria necessário, ao invés de diminuir o esforço por funcionário, aumentar a quantidade de esforço exigida pelas atividades dos jogadores, o que tornaria o jogo mais longo e atacaria o problema da grande disponibilidade de esforço.

> *Participante 2: 'acho que da primeira vez que a gente jogou tava melhor. Eu não acho que da primeira vez não tava tão fácil quanto hoje'*
>
> *Participante 1: 'Eu acho que a 'produtividade' também aumentou muito'*

Nas discussões, os jogadores apontaram que ainda não aprenderam sobre o processo e sobre as atividades. O autor refletiu se os marcadores de pontuação de Preparação, Planejamento e Execução realmente estavam ajudando ou não. Uma

recomendação no *playtesting* anterior já havia sido feita em relação aos *links* entre Atividades e Artefatos e com isso o autor concluiu que poderia ser interessante retirar esses pontos para incluir um *link* explícito.

> *Autor: 'Eu acho que o foco agora tá muito grande nesses pontos: de preparação, planejamento e execução. Talvez eu tenha que mudar o foco dos pontos para as atividades.'*

6.2.2 Visão geral

O Quadro 8 apresenta a visão geral do segundo *playtesting*, apresentando a descrição dos participantes, data, duração da avaliação e as principais melhorias (*insights*) obtidas com as discussões.

Quadro 8. Visão geral das informações do *Playtesting 2.* Fonte: o Autor.

Informações gerais	
Data	08/07/2021
Duração	2 horas e 15 minutos
Participantes	- Participante 1 (participou do *Playtesting* 1): Ex-estudante de graduação na área de computação com experiência em jogos de tabuleiro e de cartas - Participante 2 (participou do *Playtesting* 1): Estudante de graduação na área de computação com experiência em jogos de tabuleiro e de cartas - Participante 3: Estudante de graduação de área fora da computação sem experiência em jogos de tabuleiro e de cartas
Insights	- Criar mais cartas de Sabotagem - Enfraquecer as cartas Desenvolvedor, fazendo com que eles percam esforço para gerar moedas - Estabelecer os espaços de cada carta Papel no cartão dos jogadores - Trocar objetivos genéricos por objetivos mais representativos do processo de teste - Aumentar a quantidade necessária de esforço para cumprir as atividades - Remover os marcadores de pontuação de Planejamento, Preparação e Execução - Incluir links claros e explícitos entre atividades e artefatos

6.3 *Playtesting* 3

O terceiro *playtesting* foi realizado com a terceira versão do jogo, após as modificações feitas com as recomendações do *Playtesting* 2 e foi realizado no dia

10/07/2021. O *playtesting* teve duração de aproximadamente duas horas e dez minutos sendo contados desde o momento inicial de apresentação do jogo e suas regras até a finalização das discussões sobre o jogo. Este *playtesting* contou com três participantes, dois participantes que já haviam participado de outros *playtestings* e um terceiro participante, que jogava com frequência jogos de tabuleiro e era estudante de Engenharia de Software. A escolha por convidar dois participantes do primeiro *playtesting* era para novamente permitir discussões sobre o jogo e verificar se as melhorias identificadas nas discussões de *playtestings* passados tinham sido tratadas. A escolha por convidar um participante novo, com experiência com jogos de tabuleiro ou de cartas e da área de computação foi para trazer novas discussões, novas visões sobre mecânicas, regras e ideias que poderiam ser inseridas no jogo.

Inicialmente o autor e moderador do *playtesting* pediu a permissão dos jogadores para fazer a gravação. O autor apresentou a visão geral do que se tratava o *playtesting* e como ele ocorreria, explicando que o objetivo final era a discussão e aprimoramento do jogo.

O jogo foi apresentado aos três jogadores pelo autor do trabalho, que também fez algumas simulações de situações de jogo para que eles entendessem melhor as mecânicas de jogo. Após a apresentação, os jogadores tiveram a experiência com o jogo. O autor do trabalho, como moderador, ficou o tempo inteiro do jogo fazendo observações sobre comentários e interações dos jogadores com o jogo além de auxiliar os jogadores.

A versão do jogo jogada nesse *playtesting* teve avaliações muito positivas do ponto de vista de jogabilidade como também da perspectiva de aprendizagem do conteúdo, como será apresentado. Após o término do jogo o autor do trabalho utilizou o guia de discussão, que é apresentado no Apêndice A, para realizar as discussões, que serão apresentadas a seguir.

6.3.1 Principais discussões

O Participante 1 avaliou positivamente as mudanças no jogo, mas ele sofreu logo no início do jogo uma sabotagem com impacto negativo, o que afetou toda sua participação no jogo. Uma das recomendações fornecidas foi a de diminuir o impacto de algumas das Sabotagens criadas.

> *Participante 1: 'Esse daqui foi muito mais divertido do que os outros ... vamo só combinar o balanceamento dessa sabotagem do [Participante 3]'*
>
> *Participante 2: 'Essa sabotagem foi muito pesada, tem que diminuir o efeito'*
>
> *Participante 1: 'Eu acho que devia deixar essa sabotagem, mas diminui o efeito. Ou só troca o funcionário ou deixa eu pegar logo um dos que tá livre'*

Esse jogador contratou funcionários muito bons logo de início e concluiu muito rápido suas atividades, restando a ele só os *bugs* e os objetivos para serem cumpridos. Uma conclusão final tirada das discussões em grupo foi a de aumentar o tempo para realizar cada atividade, para que o jogo dure mais, ou seja, aumentar ainda mais a esforço que cada atividade requer para ser cumprida.

> *Participante 2: 'Eu acho que o desenvolvedor é muito mais importante do que o analista, o líder e o testador quando as atividades acabam'*

Nessa versão, o autor estabeleceu uma quantidade inicial de moedas como sendo 12. Nas versões anteriores eram apenas 10 moedas recebidas no início do jogo para contratação. A recomendação feita pelos participantes foi a de diminuir a quantidade de moedas para dificultar mais o jogo.

> *Participante 3: 'Coloca só 10 moedas no início. 12 moedas é muito dinheiro'*

Até esta versão, os jogadores recebem todas as suas metas de uma vez. Eles acharam melhor receber uma meta a cada duas rodadas. Se não cumprirem a meta em até duas rodadas, eles perdem o benefício da fase. Além disso, foi discutido fazer um ranking de metas. Quem terminar primeiro ganha um bônus por ter feito mais rapidamente as atividades de teste. Apesar de ser uma ideia interessante, o autor considerou a priorização de metas como não interessante de ser implementado, pois assim quem jogar primeiro na rodada terá vantagem sobre os demais.

> *Participante 2: 'Eu acho que dividir as metas por fase é melhor'*
>
> *Participante 1: 'Poderia dar um benefício para ele no máximo. Tipo assim, por exemplo: ele completou primeiro as três [metas], tipo ele ganhava um ponto a mais'*

Nessa versão quando um funcionário fazia uma atividade correta, que era de sua competência, o jogador ganhava um ponto de vitória. Os participantes discutiram que seria melhor que o jogo desse uma maior esforço ao invés de dar pontos de vitória, como uma forma de simular uma atividade em que aquele funcionário tem maior facilidade em realizar por ser de sua competência.

> *Participante 1: 'Eu acho que ao invés de colocar uma medalha aqui (na realização da atividade por funcionário), tu colocava uma 'produtividade' ... é uma mecânica melhor'*

O Participante 2 apontou que seria interessante que o jogo fornecesse alguma prioridade para certos tipos de artefatos que são considerados mais importantes para o processo, dando mais pontos de vitória para artefatos mais importantes.

> *Participante 2: 'Seria interessante se... para ensinar a pessoa a escolher o melhor teste primeiro ... tipo, para ensinar melhor ... devia aumentar os pontos de vitória dos artefatos mais importantes'*

6.3.2 Visão geral

O Quadro 9 apresenta a visão geral do terceiro *playtesting*, apresentando a descrição dos participantes, data, duração da avaliação e as principais melhorias (*insights*) obtidas com as discussões

Quadro 9. Visão geral das informações do *Playtesting 3*. Fonte: o Autor.

Informações gerais	
Data	10/07/2021
Duração	2 horas e 7 minutos
Participantes	- Participante 1 (participou dos *Playtesting* 1 e 2): Ex-estudante de graduação em Engenharia de Software com experiência em jogos de cartas e de tabuleiro

	- Participante 2: Estudante de graduação em Engenharia de Software com experiência em jogos de cartas e de tabuleiro - Participante 3 (participou do *Playtesting* 1): Estudante de graduação em Engenharia de Software com experiência em jogos de cartas e de tabuleiro
Insights	- Balancear algumas cartas de sabotagem - Aumentar o esforço requisitado pelas atividades - Diminuir a quantidade inicial de moedas dos jogadores - Mudar mecânica de puxar metas para que uma nova meta seja puxada a cada duas rodadas - Aumentar pontos de vitória de artefatos mais importantes e diminuir pontos de atividades menos importantes

6.4 Avaliação com o grupo de pesquisa

Após a realização dos *playtestings*, o jogo foi modificado usando as recomendações feitas e, como forma de avaliação final do jogo antes do experimento com os estudantes, o jogo foi aplicado com membros do GPES (Grupo de Pesquisa em Engenharia de Software) da PUCPR. Cinco membros estudantes de doutorado do grupo de pesquisa com experiência conceitual e prática na indústria participaram da avaliação, sendo 3 deles professores de computação para estudantes de graduação. A intenção dessa última aplicação no grupo era o de apresentar o jogo para os membros do grupo e obter a percepção deles em relação ao aprendizado de teste com o jogo, se o conteúdo e a forma como ele é abordado no jogo está correto e representando o processo de testes na realidade.

A aplicação no GPES teve uma duração de aproximadamente 3 horas, nas quais o jogo ProTesters foi apresentado, jogado e discutido entre os participantes. A partir desta última aplicação, os participantes do grupo de pesquisa mostraram contentamento com o jogo em relação tanto ao aspecto lúdico quanto do aspecto de aprendizagem.

Durante e ao final da aplicação, diversos pontos foram debatidos entre os participantes, dentre eles algumas recomendações foram levadas em conta para a elaboração da versão final do jogo, que seria utilizada no experimento. Os principais pontos a mudar serão apresentados a seguir.

A primeira recomendação foi a criação de uma apresentação do jogo para padronizar a facilitar a explicação do jogo, seus componentes e objetivos. Também foi pedido para que fosse feita uma melhor separação do ambiente online do jogo no Miro, para que os jogadores tivessem os espaços de cada componente do jogo bem definidos, o que facilitaria na organização do jogo.

Uma das principais mudanças discutidas no grupo foi a eliminação do papel do Desenvolvedor do jogo. Os participantes entenderam que o desenvolvedor é um papel que não faz parte diretamente do processo de teste e que não deveria ser contemplado pelo jogo. Por isso, na versão final utilizada, o papel do Desenvolvedor foi retirado para focar apenas nos papeis do processo de teste. Além disso, foi feita a troca de pontos de vitória por pontos em forma de bug. Encontrar mais bugs é o objetivo do processo de teste e por isso os participantes recomendaram que a pontuação do jogo não fosse apenas um ponto de vitória genérico, mas fossem os próprios bugs, que posteriormente se transformaram no “bugômetro”.

Após as recomendações apresentadas e alteradas no jogo, a versão final do ProTesters ficou pronta para ser aplicada no experimento com os estudantes de graduação no experimento.

6.5 Experimento

A avalição do artefato (jogo) foi feita por meio de um experimento com o objetivo de avaliar a proposta em relação à motivação e ganho de aprendizado dos jogadores. O experimento com o jogo foi realizado a fim de verificar o aprendizado dos estudantes nas competências que a proposta de jogo educacional pretende abordar. No Quadro 10 é apresentado o objetivo do experimento seguindo a abordagem *Goal-Question-Metric* (GQM) (BASILI & ROMBACH, 1988).

Quadro 10. Objetivo do trabalho seguindo a abordagem GQM. Fonte: o Autor.

Analisar	**A eficácia da proposta de jogo educacional a ser desenvolvida**
Com o propósito de	Verificar o cumprimento dos objetivos da proposta
Em relação ao	Aprendizado sobre o processo de teste e motivação percebido nos jogadores

Do ponto de vista de	Estudantes de graduação em Ciência da Computação e áreas afins que ainda não tiveram disciplinas sobre testes de software
No contexto de	Avaliação da proposta de jogo educacional

6.5.1 Execução do experimento

O experimento com o jogo ProTesters foi realizado no dia 16/09/2021 com 16 estudantes dos cursos de Ciência da Computação e Engenharia de Software da PUCPR. O jogo foi aplicado para estudantes que ainda estão no início de seus respectivos cursos e que ainda não tiveram disciplinas que abordem os testes de software. Por conta do pouco conhecimento que os estudantes possuíam sobre testes de software até a data de realização do experimento, eles foram considerados uma população válida para a aplicação de um experimento com jogo educacional cujo objetivo era medir o ganho de aprendizado com o jogo.

Antes de iniciar a aplicação, o autor do trabalho preparou 6 ambientes de jogo para quatro jogadores cada no ambiente Miro. O autor fez toda a fase de preparação do ambiente de jogo, embaralhando o baralho, separando o material de cada jogador e definindo as áreas do jogo para a mão do jogador, área do banco, área das atividades e artefatos dos jogadores e espaços de cada função de funcionário. Além disso, como o autor não teria condição de atender simultaneamente todos os ambientes de jogo no Miro e tirar dúvidas em tempo real de todos, foi deixado em cada ambiente de jogo uma nota com o passo-a-passo de ações para cada rodada.

Antes da aplicação do jogo, os estudantes foram apresentados sobre os objetivos do experimento e do jogo educacional. O autor do trabalho esclareceu que o desempenho dos jogadores no jogo e nos pré e pós-teste não teriam influência na avaliação deles em qualquer disciplina. Após isso, antes de explicar a regra do ProTesters, o autor pediu para que os estudantes respondessem ao questionário pré-teste que mediria o conhecimento dos estudantes antes da experiência com o jogo. O questionário de pré-teste do jogo é apresentado no Apêndice D e tem, além das perguntas sobre o processo de teste, um Termo de Consentimento Livre e Esclarecido (TCLE) para participar do experimento e perguntas demográficas para mapeamento da amostra. Após 15 minutos cedidos para que respondessem ao pré-teste, 18

estudantes responderam ao pré-teste. Como 2 desses estudantes não participaram do experimento com o jogo, as respostas deles foram eliminadas.

Com o pré-teste aplicado, o autor do trabalho apresentou o jogo, seus objetivos, componentes, regras e jogabilidade. A fim de demonstrar algumas situações que podem ocorrer no jogo, também simulou uma rodada de jogo para que os estudantes entendessem seu funcionamento. Após isso, os estudantes tiraram suas dúvidas e os 16 participantes foram divididos em 4 grupos de quatro jogadores cada.

A fim de ajudar os jogadores e assegurar que o jogo fosse jogado de maneira correta, o autor do trabalho passou de grupo em grupo durante as 2 horas de jogo. No início era possível notar que os jogadores ainda estavam confusos com o funcionamento do jogo e execução das atividades. Apesar da nota de explicação das rodadas estar destacada, poucos jogadores conseguiram seguir o fluxo das rodadas logo de início. Entretanto, com o passar do tempo e as visitas feitas pelo autor, foi possível notar que os jogadores já estavam mais familiarizados com o jogo e estavam seguindo sozinhos o fluxo do jogo.

O experimento com o ProTesters teve duração de 2 horas. Após as 2 horas de jogo, por conta da limitação de tempo de alguns participantes e como alguns grupos já haviam terminado o jogo, o autor decidiu terminar o experimento com o jogo. Dois grupos conseguiram terminar o jogo, e outros dois chegaram nas duas rodadas finais de jogo. Um dos grupos, entretanto, apresentou mais dificuldade em aprender o jogo e terminou o experimento apenas na terceira rodada.

Como forma de avaliar o ganho de desempenho, o autor do trabalho aplicou o questionário de pós-teste com os participantes no final do experimento. O questionário de pós-teste tem 9 questões iguais ao questionário de pré-teste e apenas uma diferente, que é a última. A razão disso é para minimizar o efeito de mudar as perguntas de um teste para o outro na inferência de ganho de aprendizado antes e depois do experimento com o jogo. Além disso, como forma de avaliar a qualidade do jogo e a motivação que ele promove, o pós-teste também apresenta um questionário customizado do modelo MEEGA+ (PETRI, et al., 2019), que avalia o jogo do ponto de vista de usabilidade e experiência do jogador. O Apêndice E apresenta o questionário de pós-teste que foi respondido pelos 16 jogadores no final do experimento.

6.5.2 Análise e apresentação dos resultados

Em relação à caracterização dos participantes, o questionário de pré-teste apresenta algumas questões para melhor conhecer a amostra de jogadores que participaram do experimento. A partir das respostas dos jogadores, foi visto que a amostra era jovem, com todos os participantes tendo entre 18 e 28 anos. Também foram feitas perguntas em relação à experiência dos participantes com jogos não-digitais. Quando perguntados sobre a frequência com que jogam jogos de tabuleiro, os participantes informaram que não têm muito costume de jogar esse tipo de jogo. A Figura 41 apresenta o gráfico das respostas de frequência dos participantes, mostrando que 13% informaram nunca terem jogado jogos de tabuleiro, quase metade (44%) informaram que raramente jogam esse tipo de jogo e 31% afirmaram jogar semanalmente.

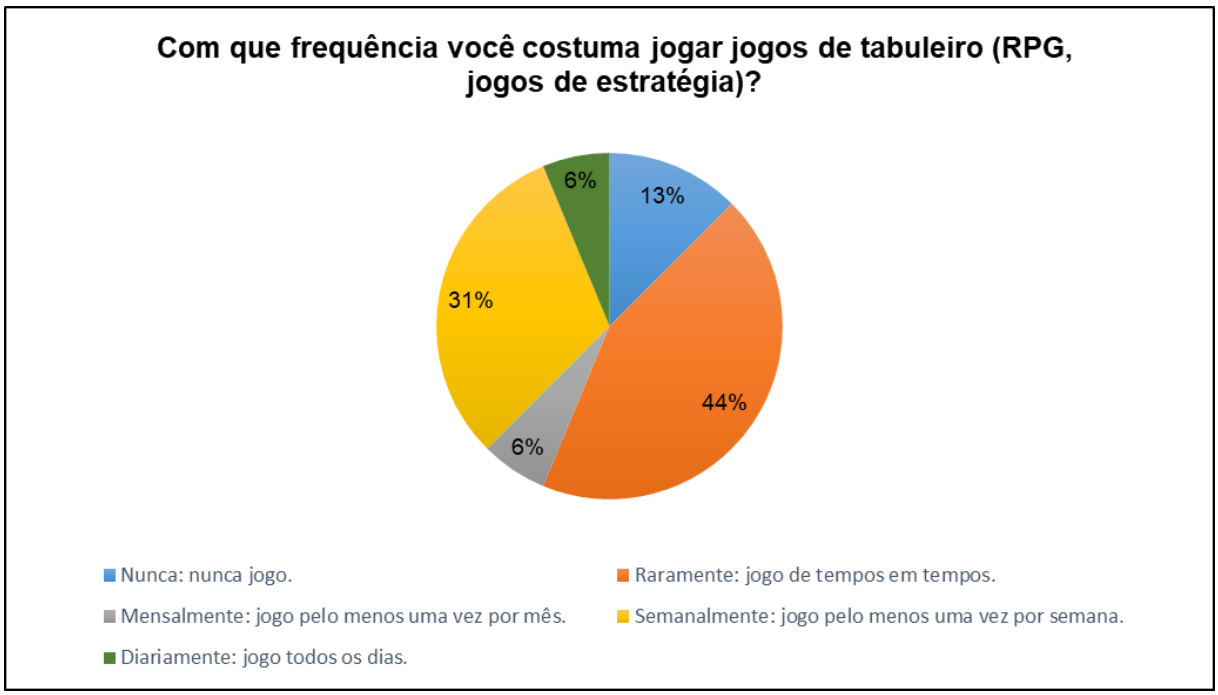


Figura 41. Gráfico de frequência dos participantes com jogos de tabuleiro. Fonte: o Autor.

Quando perguntados sobre a frequência com que jogam jogos de cartas, os participantes novamente revelarem ter pouca experiência em jogar jogos desse tipo. A Figura 42 apresenta o gráfico das respostas de frequência dos participantes, revelando que mais da metade dos participantes nunca jogaram ou raramente jogam

jogos de cartas. Além disso, 31% informaram jogar apenas mensalmente jogos de cartas.

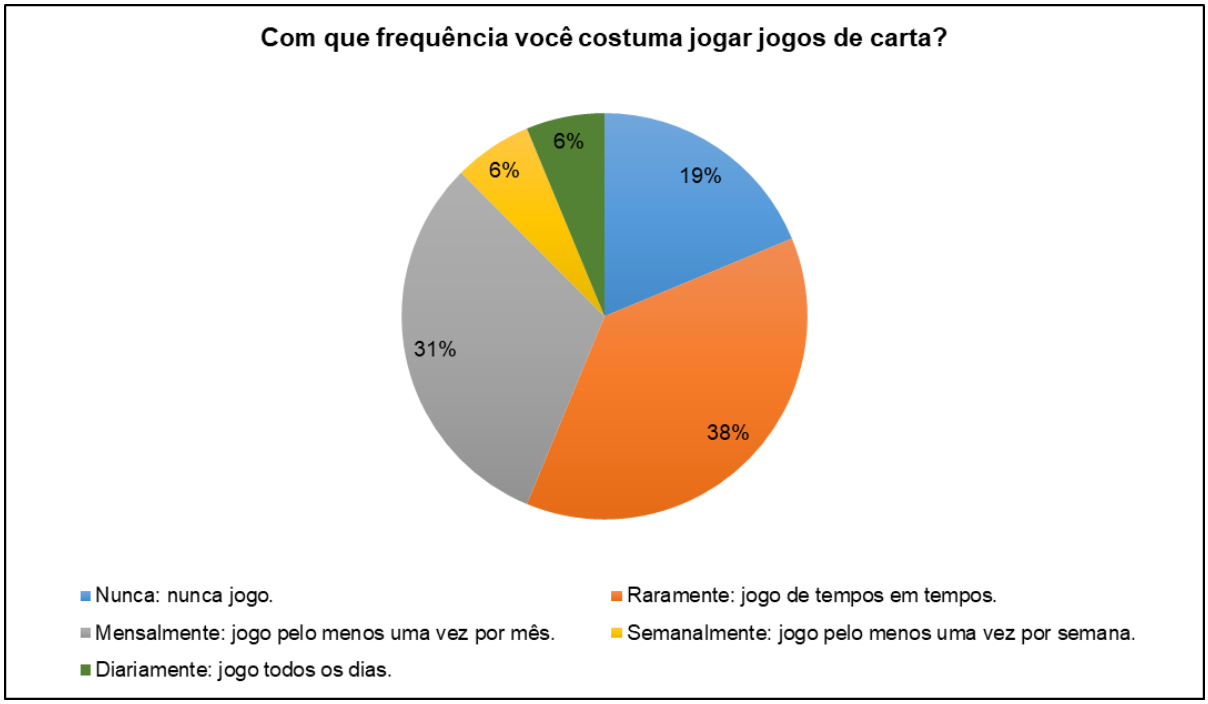


Figura 42. Gráfico de frequência dos participantes com jogos de cartas. Fonte: o Autor.

Quando perguntados sobre a frequência com utilizam testes de software no trabalho ou em atividades da universidade, os jogadores informaram utilizar testes com pouca frequência. A Figura 43 apresenta o gráfico das respostas de frequência dos jogadores, revelando que mais da metade dos participantes nunca utilizou testes de software e que 38% utilizam raramente testes em suas atividades da universidade ou trabalho. Apenas 1 participante (6%) informou sempre usar testes para testar suas soluções.

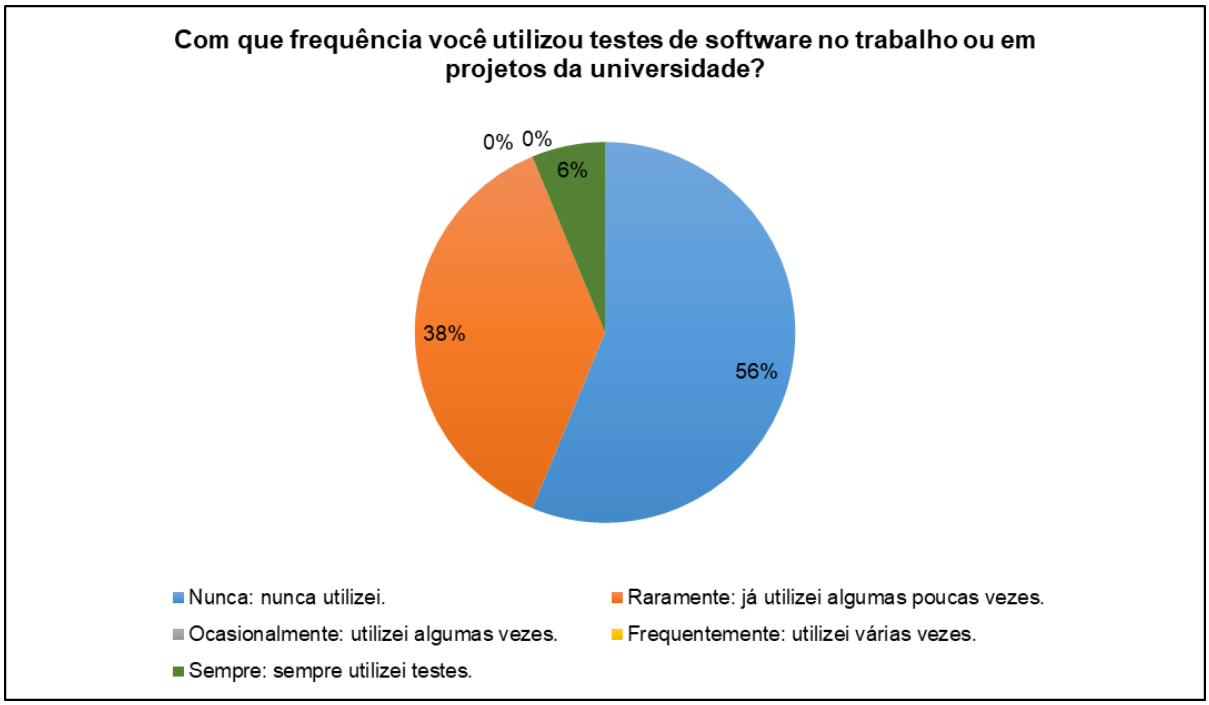


Figura 43. Gráfico de frequência dos participantes com a utilização de testes. Fonte: o Autor.

Quando perguntados sobre se já tiveram algum curso sobre testes de software, quase todos os jogadores informaram ainda não ter aulas sobre esse conteúdo. A Figura 44 apresenta o gráfico das respostas dos jogadores que participaram de cursos sobre testes de software, revelando que quase todos os jogadores (87%) não tiveram aulas sobre esse conteúdo. Apenas 2 participantes (13%) informaram ter feito pelo menos um curso sobre testes.

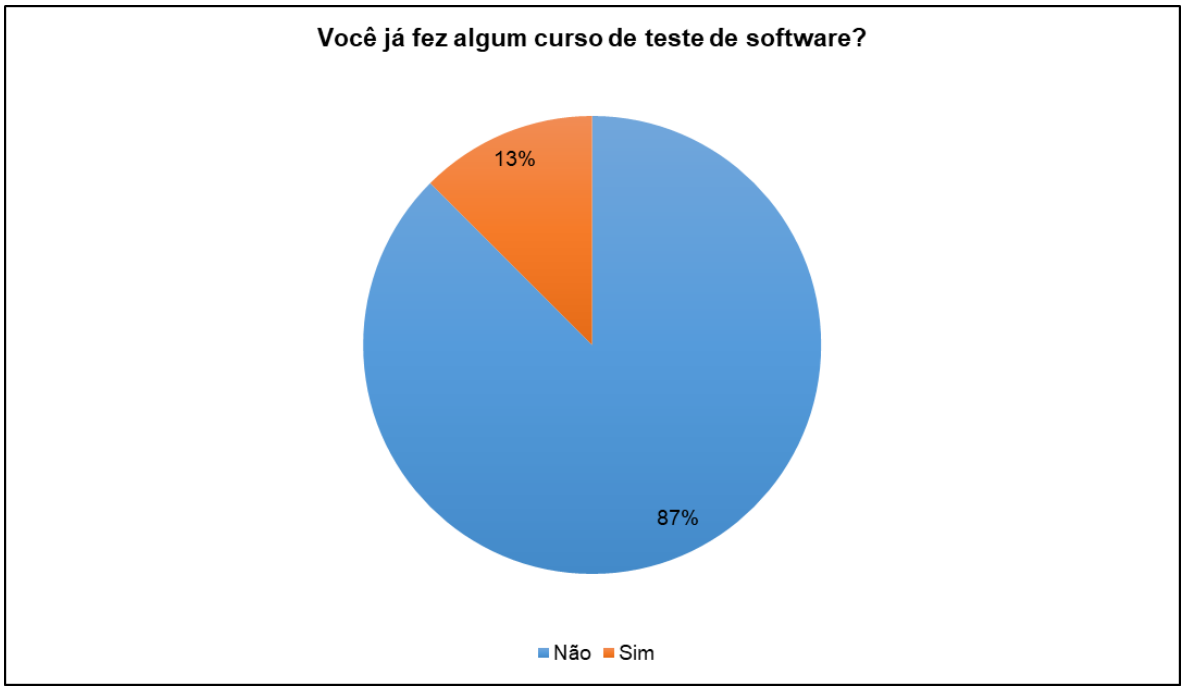


Figura 44. Gráfico de participantes que já tiveram curso sobre testes de software. Fonte: o Autor.

De modo geral, com as respostas aos itens de informações demográficas dos participantes, foi possível ver que em geral os participantes tinham pouca ou nenhuma experiência com testes de software. Além disso, também foi visto que a maioria dos participantes jogavam com pouca frequência jogos de tabuleiro e de cartas, o que pode influenciar na avaliação de um jogo tabuleiro educacional, como é o ProTesters. Isso porquê, como os participantes têm pouca experiência com jogos, é provável que eles não compreendam facilmente muitas das mecânicas utilizadas no ProTesters e não consigam relacionar as mecânicas com experiências com jogos passados. Por conta disso, alguns jogadores podem sentir mais dificuldade em entender e conseguir aplicar as mecânicas do jogo com facilidade, o que pode afetar a avaliação de aspectos de usabilidade e jogabilidade do jogo.

Como forma de avaliar a motivação proporcionada pelo jogo, além de aspectos de qualidade, como a usabilidade e experiência, foram analisadas as respostas ao questionário do MEEGA+. A Figura 45 e a Figura 46 apresentam os resultados obtidos na avaliação do ProTesters, utilizando as planilhas fornecidas para o modelo MEEGA+. Com relação ao fator de qualidade e usabilidade, foram avaliados os

aspectos estéticos, apreensibilidade, operabilidade e acessibilidade. A *Figura 45* mostra a avaliação dos participantes em cada um desses aspectos.

A avaliação da usabilidade do jogo foi de maneira geral positiva. Apenas alguns aspectos de apreensibilidade e estética obtiveram discordâncias entre os participantes.

Alguns participantes discordaram ou indicaram neutralidade em relação à afirmativa sobre os aspectos de estética: “O *design* do jogo é atraente (tabuleiro, cartas, interfaces, gráficos etc.)”. Esse resultado pode ter sido influenciado principalmente pelo ambiente de jogo, que não era o ideal para a aplicação do jogo. O ProTesters é um jogo de tabuleiro, entretanto, por conta das limitações causadas pela pandemia do Covid-19, o jogo não pode ser aplicado presencialmente, mas de forma remota, utilizando o ambiente Miro. Ao carregar o material do jogo para o ambiente, houve uma perda da qualidade visual das cartas.

Também foram observadas discordâncias e indicações de neutralidade em relação às afirmativas do aspecto apreensibilidade. A complexidade de jogo e a quantidade de cartas e regras pode ser um dos grandes influenciadores desse resultado. ProTesters tem pouco mais de 180 cartas, divididas em artefatos, atividades, cartas de *deck* e funcionários, além de diversas regras que influenciam o jogo. Por conta disso, vários jogadores começaram o jogo ainda sem saber de todas as suas regras e como jogar, mesmo após a apresentação do jogo e a simulação de uma rodada. Entretanto, com o acompanhamento constante do moderador, os jogadores logo compreenderam o fluxo de jogo e praticamente todos os grupos avançaram para as fases finais de jogo.

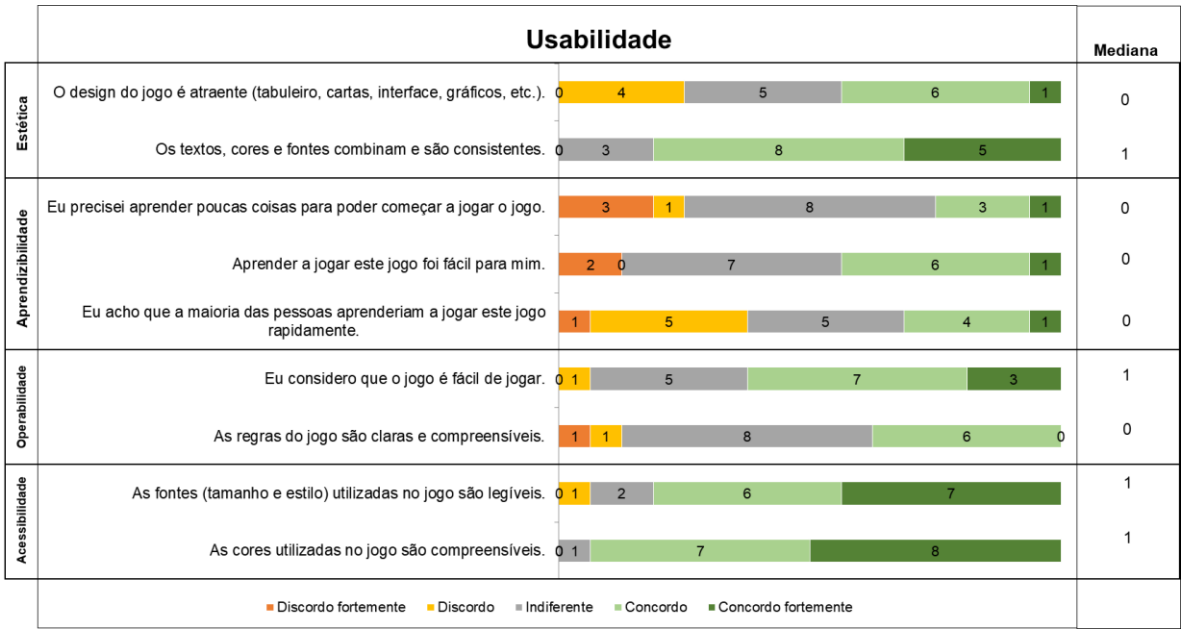


Figura 45. Gráfico de avaliação do ProTesters, fator usabilidade. Fonte: o Autor.

Em relação às discordâncias e neutralidade sobre o aspecto de operabilidade, as discordâncias desse aspecto podem ter sido resultantes do fato tanto do ambiente onde o jogo foi aplicado, que não era o ideal, como também aspectos relacionados à complexidade do jogo, que pode ter dificultado sua execução pelos jogadores.

Já em relação ao aspecto de Acessibilidade, o jogo foi novamente muito bem avaliado pelos jogadores e só uma de suas afirmações obteve discordância: “As fontes (tamanho e estilo) utilizadas no jogo são legíveis”. O tamanho das fontes já havia sido debatido em alguns *playtestings*, e o autor tratou o máximo possível esse problema tentando manter também o máximo de limpeza e legibilidade possível nas cartas. Apesar disso, essa questão ainda é uma melhoria que pode ser feita no jogo.

Com relação aos resultados no fator de qualidade da Experiência do jogador sobre o ProTesters, foram avaliados os aspectos confiança, desafio, satisfação, interação social, diversão, atenção focada, relevância e percepção de aprendizagem. A avaliação dos diversos aspectos do fator de qualidade do aprendizado do jogador foi muito positiva, obtendo poucas discordâncias em suas afirmações. A Figura 46 apresenta a avaliação dos estudantes em relação a esse fator de qualidade.

Considerando as discordâncias no fator de confiança: “A organização do conteúdo me ajudou a estar confiante de que eu iria aprender com este jogo”. Esse resultado pode ter sido influenciado principalmente pela organização do ambiente de

jogo no Miro, além da quantidade de cartas e material de jogo que pode dificultar sua organização pelos jogadores. Como forma de minimizar isso, o autor e moderador do experimento preparou o ambiente, separando as áreas de jogo de cada jogador e organizou o material de jogo em cada campo.

Em relação às discordâncias no aspecto de satisfação, foram analisadas as seguintes afirmações: “Completar as tarefas do jogo me deu um sentimento de realização”, “É devido ao meu esforço pessoal que eu consigo avançar no jogo” e “Eu recomendaria este jogo para meus colegas”. Esse resultado pode ter sido influenciado principalmente pela complexidade e não entendimento do jogo, o que fez com que, no início, os jogadores tivessem dificuldade em avançar no jogo.

O ProTesters teve avaliação positiva. Observando as afirmações dos aspectos interação social e diversão, o jogo não obteve nenhuma discordância na avaliação dos jogadores, o que mostra um dos pontos fortes do jogo, que é promover a interação entre os jogadores e diverti-los.

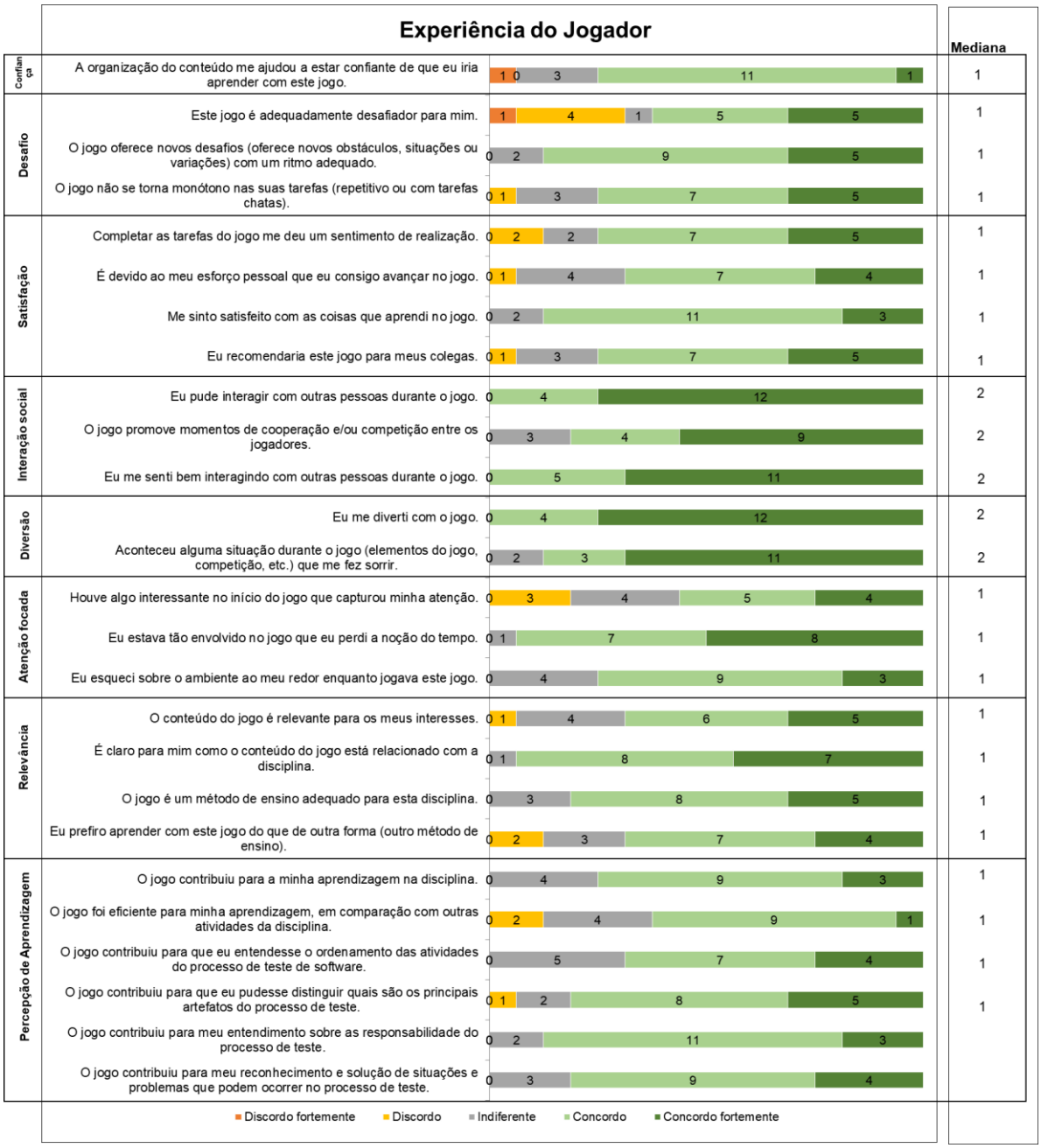


Figura 46. Gráfico de avaliação do ProTesters, fator experiência do jogador. Fonte: o Autor.

No aspecto atenção focada, apenas uma das afirmações obteve discordâncias: “Houve algo interessante no início do jogo que capturou minha atenção”. Os jogadores do ProTesters têm ainda no início do jogo que realizar suas contratações e preparação do *deck* de cartas e diversas ações não relacionadas ao jogo em si. Após essa preparação inicial é que o jogo começa apresentando seus eventos, problemas e a execução das atividades de teste. Por conta disso, o início do jogo do ProTesters não apresenta nenhuma questão de jogo que requisite a atenção dos jogadores.

No aspecto relevância, o jogo foi bem avaliado pelos jogadores e obteve apenas discordâncias nos itens: “O conteúdo do jogo é relevante para meus interesses” e “Eu prefiro aprender com este jogo do que de outra forma (outro método de ensino)”. Novamente a complexidade do jogo, o ambiente utilizado para aplica-lo e a dificuldade de alguns jogadores em compreender a jogabilidade pode ter influenciado o resultado desse aspecto, fazendo com que alguns estudantes preferissem outra forma de aprender o conteúdo. Apesar disso, de modo geral, o jogo foi bem avaliado nesse aspecto, sendo que duas afirmativas desse aspecto não apresentaram discordâncias e todos os itens tiveram uma mediana no nível de concordância. Além disso, o item “Eu prefiro aprender com este jogo do que de outra forma (outro método de ensino)”, apresentou poucas discordâncias, apenas 2, sendo que a maioria dos jogadores concordaram ou concordaram fortemente com essa afirmação, o que revela a aceitação da nova abordagem pelos participantes.

No aspecto aprendizagem percebida, o jogo foi muito bem avaliado pelos jogadores, especialmente nas afirmativas: “O jogo contribuiu para a minha aprendizagem na disciplina”, “O jogo contribuiu para que eu entendesse o ordenamento das atividades do processo de teste de software”, “O jogo contribuiu para meu entendimento sobre as responsabilidade do processo de teste” e “O jogo contribuiu para meu reconhecimento e solução de situações e problemas que podem ocorrer no processo de teste”, que não apresentaram discordâncias. Esse resultado indica que na percepção de grande parte dos jogadores o jogo cumpriu seus objetivos educacionais para abordar os assuntos propostos.

Como forma de verificar o que foi aprendido pelos estudantes, o pós-teste também questionou os estudantes sobre o que os jogadores aprenderam com o jogo. Para entender melhor os aspectos do jogo que impactam de modo positivo no aprendizado dos estudantes, foram analisadas as respostas dadas à questão aberta “Nos conte sobre o que você aprendeu com o jogo ProTesters”. Todos os 16 jogadores fizeram comentários sobre o que aprenderam com o jogo.

A questão aberta foi analisada de forma qualitativa, por meio da codificação aberta. Como resultado dessa codificação, foram identificados 8 códigos. A Figura 47 apresenta as categorias identificadas sobre o que os jogadores afirmaram ter aprendido com o jogo.

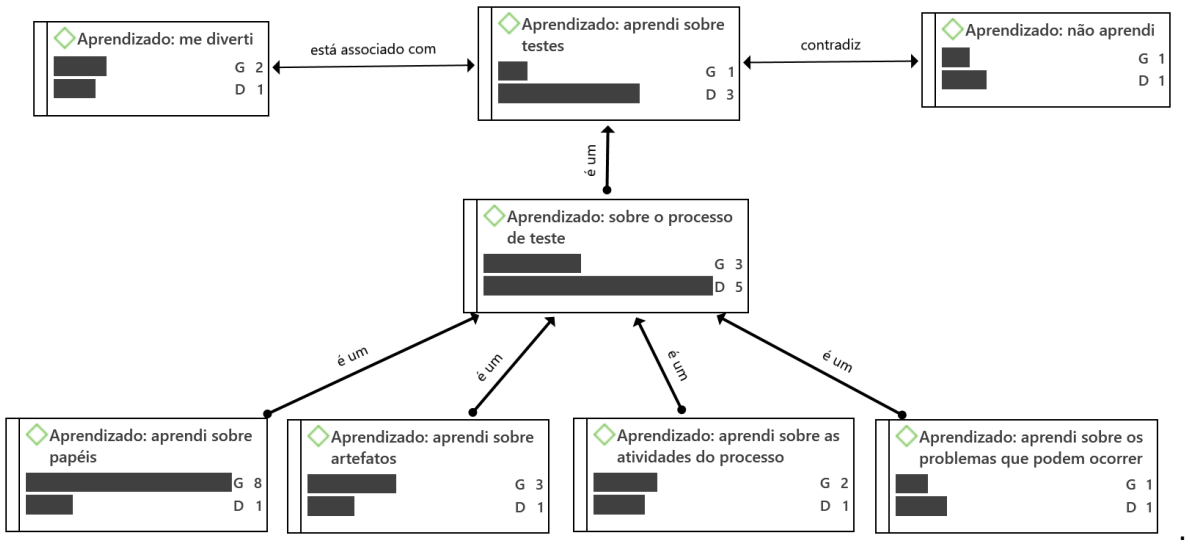


Figura 47. Códigos da análise de aprendizado com o ProTesters. Fonte: o Autor.

Analisando a Figura 47, é possível ver que os jogadores afirmaram aprender diversos tópicos relacionados ao processo de testes de software. Oito jogadores afirmaram ter aprendido sobre os papéis do processo de teste, três afirmaram ter aprendido sobre os artefatos, dois afirmaram ter aprendido algo sobre as atividades do processo de teste e um afirmou ter aprendido sobre os problemas que podem ocorrer no processo. Todos esses conteúdos estão relacionados ao processo de teste de software e, consequentemente, com o aprendizado sobre testes. Como pode ser visto, o aprendizado mais frequente entre os estudantes foi relacionado aos papéis do processo.

Três estudantes afirmaram ter aprendido sobre o processo de teste, sem especificar qual tópico. Um estudante afirmou ter aprendido sobre testes de modo geral. Um estudante também afirmou não ter aprendido nada e dois afirmaram ter se divertido com o jogo. O Quadro 11 apresenta alguns exemplos de citações e os códigos relacionados.

Quadro 11. Códigos sobre os conteúdos aprendidos pelos jogadores com o ProTesters. Fonte: o Autor.

Código {Fundamentação Teórica}	**Citações dos jogadores**
Aprendizado: aprendi sobre papéis {8}	“Aprendi um pouco sobre a função de cada funcionário ...”, “consegui compreender o papel de cada pessoa na área de trabalho ...” e “Eu entendi

Código {Fundamentação Teórica}	Citações dos jogadores
	melhor sobre o papel que cada pessoa desenvolve dentro de um processo de testes”
Aprendizado: sobre o processo de teste {3}	“Aprendi sobre as etapas e processos do teste de software” e “Eu aprendi sobre os processos de testes de software.”
Aprendizado: aprendi sobre artefatos {3}	“Aprendi sobre ...e quais artefatos são obtidos” e “a diferença entre script, log e suite de testes”
Aprendizado: me diverti {2}	“Me divertido bastante com o jogo ...”
Aprendizado: aprendi sobre as atividades do processo {2}	“Aprendi sobre as etapas ...”
Aprendizado: aprendi sobre testes {1}	“Aprendi mais sobre a área de testes sobre a qual às vezes ouço ...”
Aprendizado: aprendi sobre os problemas que podem ocorrer {1}	“... e os problemas que podem acontecer”
Aprendizado: não aprendi {1}	“Não aprendi muita coisa, porém me diverti muito jogando”

Como pode ser visto, a análise das respostas à questão aberta sobre a aprendizagem com o jogo reforça a ideia de cumprimento dos objetivos de aprendizagem do jogo. Entretanto, o MEEGA+ e a análise das respostas à questão aberta apenas mostram o aprendizado percebido pelos jogadores, o que pode não significar que eles realmente aprenderam o conteúdo abordado. Para verificar o ganho de aprendizado que os jogadores tiveram com o jogo, foi feita uma análise comparativa do pré e do pós-teste.

Para a análise de pré e pós-teste, o autor do trabalho fez as correções das respostas dos estudantes que aceitaram participar do estudo. Para isso, as respostas do questionário online foram passadas para uma folha de cálculo e as questões foram corrigidas, sendo que cada questão vale um ponto. Como cada teste tem 10 questões, a pontuação do pré e pós-teste pode variar de 0-10 pontos. Para questões de múltipla escolha, o autor dividiu o total de itens por 1, que é o valor máximo da questão. Cada

item marcado diferente da resposta correta é descontado do valor da correção da questão.

Ao realizar a correção dos resultados do pré-teste, foi possível obter o nível de conhecimento dos estudantes sobre o conteúdo de teste. Em média, os estudantes tiveram nota de 5,08 pontos, sendo a maior nota 7,5 de um dos estudantes e a menor nota 2.

Já em relação aos resultados do pós-teste, a média de pontos dos estudantes foi um pouco maior, atingindo 5,80. As pontuações do pós-teste foram mais equilibradas, com a maior nota sendo 8,33 pontos e a menor nota 1,5 pontos.

A Figura 48 apresenta um gráfico Boxplot comparativo entre os resultados obtidos no pré-teste e no pós-teste. Com ele é possível ver que os resultados do pós-teste foram mais centralizados do que os resultados do pré-teste. Além disso, é possível ver que os jogadores aumentaram de nota na avaliação de pós-teste, se concentrando em resultados melhores. Além da mediana, também é possível ver que os resultados entre o primeiro e terceiro quartil e a maior nota no pós-teste foi maior do que os resultados no pré-teste, demonstrando que no pós-teste os jogadores se concentraram em resultados melhores. Com isso, é possível verificar que além de melhorar a avaliação dos jogadores em relação ao conteúdo, a maioria dos jogadores conseguiram absorver o conteúdo de uma maneira melhor e mais homogênea, com menos variação.

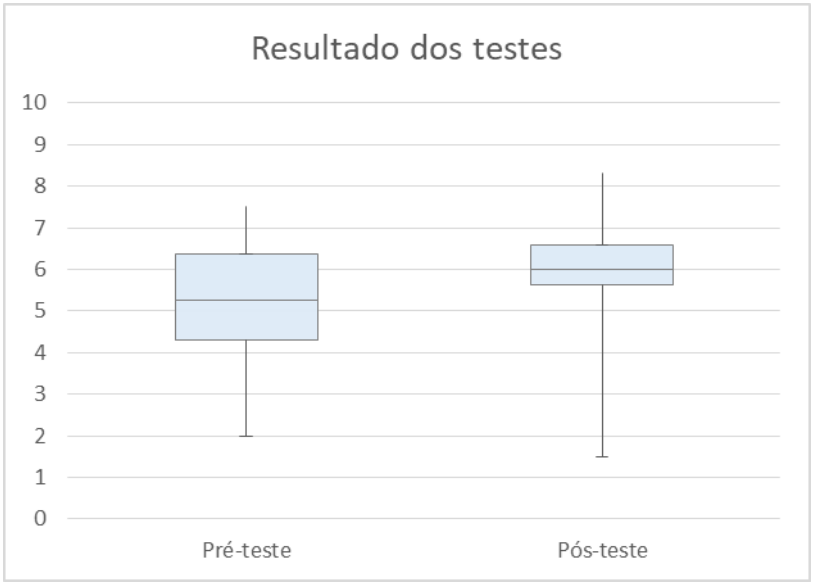


Figura 48. Boxplot comparativo de pré e pós-teste. Fonte: o Autor.

Apesar do gráfico apresentar uma melhora que os estudantes tiveram de um teste para o outro e a avalição geral dos estudantes sobre o aprendizado percebido indicar um ganho de conhecimento, é preciso saber se essa diferença é representativa, ou seja, se houve um ganho significativo de aprendizagem com o jogo entre os testes. Para isso, foi realizado um teste de hipótese para testar as seguintes hipóteses:

- **H0**: Os estudantes **não apresentam** um conhecimento superior sobre o processo de teste no pós-teste em relação aos resultados do pré-teste.
- **H1**: Os estudantes **apresentam** um conhecimento superior sobre o processo de teste no pós-teste controle em relação aos resultados do pré-teste.

Para testar as seguintes hipóteses foi aplicado um Teste-T com amostras pareadas, aplicando o valor da pontuação como comparador de médias e o tipo de teste (pré ou pós-teste) como agrupamento. Tendo em vista a pequena quantidade de participantes respondentes, o teste foi realizado com 90% de confiança para testar as hipóteses. A Figura 49 apresenta o resultado do teste de hipótese, mostrando um nível de significância de 0,07.

Paired Samples T-Test

Paired Samples T-Test

Measure 1		Measure 2	t	df	p	Mean Difference	SE Difference
Pre teste	-	Pos teste	−1.555	15	0.070	−0.718	0.462

Note. For all tests, the alternative hypothesis specifies that Pre teste is less than Pos teste.
Note. Student's t-test.

Figura 49. Resultado do Teste-T com amostras pareadas. Fonte: o Autor.

Como o nível de segurança utilizado é de 0,1 (10%), é possível rejeitar a hipótese nula (H0) com 90% de confiança. Com isso, é possível afirmar que os estudantes no pós-teste tinham conhecimento superior sobre o processo de teste do que no pré-teste. Como a única influência entre os pós e pré-teste foi a experiência com o jogo, é possível também afirmar que o ProTesters contribuiu para que os jogadores obtivessem conhecimento sobre o processo de teste em relação às competências que ele propõe abordar.

6.5.3 Discussões

A partir dos resultados apresentados com as avaliações do jogo ProTesters, é possível destacar algumas observações sobre o jogo e seus resultados.

A avaliação de aprendizagem se mostrou superior no pós-teste, o que permitiu com que a hipótese nula fosse negada com confiança. Como o fator jogo foi o único influenciador no aprendizado dos jogadores entre o período de pré e pós-teste, é possível afirmar que o jogo influenciou de forma positiva no aprendizado dos estudantes, para esta amostra estudada, que não possuía conhecimento anterior em teste de software.

A partir da resposta ao questionário do MEEGA+, foi possível observar que o jogo apresentou uma avaliação positiva na opinião dos jogadores nos aspectos de qualidade de experiência dos jogadores. Apesar disso, alguns aspectos de usabilidade apresentaram discordâncias entre os jogadores, como a apreensibilidade e a operabilidade. Alguns dos possíveis causadores das discordâncias entre os jogadores podem ter sido o ambiente utilizado para aplicar o jogo, que não era o ideal, e a complexidade das regras do jogo, o que pode ter feito com que alguns jogadores menos experientes tivessem mais dificuldade.

Em relação à análise qualitativa realizada sobre as respostas do aprendizado dos jogadores com o jogo, foi possível entender que os jogadores perceberam o que estavam aprendendo com o jogo, em relação às competências que ele aborda, e de forma lúdica. Apenas um jogador afirmou não ter conseguido aprender com o jogo, embora tenha se divertido com o jogo.

O tamanho da amostra é um grande limitador dos resultados da pesquisa. A pesquisa contou com 16 jogadores. Esses números são pequenos tendo em vista a população total de estudantes de Computação e áreas afins, o que dificulta a extrapolação dos resultados para o restante da população. Entretanto, a amostra escolhida é relevante para o estudo, pois são estudantes de graduação e ainda não tiveram disciplinas sobre testes de software, o que permitiu avaliar o ganho de aprendizagem com o jogo.

Por conta de limitações de tempo e por conta do próprio *design* do experimento, não foram realizadas discussões com os estudantes. Esta seria uma etapa recomendada para o pós-jogo, ou seja, professores em um ambiente real de ensino,

devem discutir, ao final do jogo, os temas de estudo envolvidos a partir das proposições do jogo e as suas mecânicas.

6.6 Comparação com trabalhos relacionados

A avaliação do ProTesters foi feita tanto pela análise de hipóteses feitas pelo pré e pós-teste como também pela percepção de aprendizagem, usabilidade e motivação promovida pelo jogo, o que permite comparar a nova abordagem com outras relacionadas. O JETS (SILVA & MÜLLER, 2012), foi utilizado um questionário próprio de 11 questões para a avaliação da proposta. Nesse questionário é interessante observar as respostas ao item “10 - Você gostou de jogar o JETS?”, no qual 2 estudantes dos 7 que responderam afirmaram não gostar da proposta. Tal resultado pode estar relacionado ao formato do jogo não abstrair a complexidade do mundo real para trabalhar os testes. As fases do jogo são feitas de perguntas e respostas e em uma delas o jogador tem que inclusive criar testes unitários para testar um produto. Já o ProTesters aborda o conteúdo de uma forma mais lúdica, abstraindo a complexidade do mundo real para abordar o processo de testes. Com essa nova abordagem o ProTesters obteve resultados muito positivos nos aspectos de satisfação, interação social e diversão, sendo que nesses dois últimos a mediana das respostas ficou no nível de Concordo totalmente, como mostra a Figura 46. Além disso, o JETS se concentra mais nos papéis do processo de teste, mas não aborda as atividades, artefatos nem situações que ocorrem no processo de teste.

O *iTest Learning* (FARIAS, et al., 2012), da mesma forma como o JETS, utilizou um questionário próprio com 5 questões para avaliar o jogo. A avaliação do jogo com o questionário foi positiva na visão dos jogadores, entretanto a forma de abordagem do conteúdo do *iTest Learning* é feito com várias perguntas onde o jogador deve selecionar tipos, níveis, ferramentas de teste e itens dos softwares a serem testados no jogo. Por ser feito de perguntas, isso pode não motivar os estudantes a aprender realmente sobre o conteúdo. Além disso, o jogo presume que o jogador já deve conhecer sobre o conteúdo antes de jogá-lo, servindo mais como um validador do conhecimento do jogador sobre o conteúdo. O ProTesters, por outro lado, não aborda o conteúdo em forma de perguntas, mas sim através de mecânicas de jogos que se relacionam com o conteúdo. Além disso, como já afirmado anteriormente, o *iTest*

Learning se concentra no planejamento dos testes, não abordando as demais atividades, papéis nem situações do processo de teste.

O *IslandTest* (QUEIROZ, et al., 2019) aplicou um questionário baseado no MEEGA (SAVI, et al., 2011) para avaliar o jogo quanto a satisfação, confiança, relevância e a atenção do jogador durante e depois do jogo. A avaliação dos jogadores foi positiva, apesar de o jogo apresentar os mesmos problemas que os outros na forma de abordar o conteúdo. O *IslandTest* aborda vários conteúdos de teste em forma de pergunta e resposta. Mais especificamente, em relação ao processo de teste, o jogo traz apenas uma pergunta em que o jogador tem que lembrar a ordem das atividades do processo. O ProTesters, aborda as atividades de forma mais lúdica e dinâmica, através das metas para ordenar as atividades, além de abordar os papéis, artefatos e situações do processo de teste de software.

6.7 Considerações sobre o Capítulo

Este capítulo apresentou as várias avaliações realizadas com o ProTesters. Com os *playtestings* e discussões com o grupo de pesquisa foi possível identificar melhorias na jogabilidade e em aspectos do conteúdo do jogo antes de aplicá-lo com seu público final, os estudantes. Isso permitiu que correções pudessem ser feitas ao longo das iterações, ainda durante a criação do artefato (jogo). Com o experimento e seus resultados, foi possível observar a percepção dos estudantes sobre o jogo em relação à experiência dos jogadores com o ProTesters. Também foi possível verificar que os estudantes entenderam como o jogo abordou o conteúdo proposto e o conhecimento que eles tiveram com o jogo. Em relação às avaliações de conhecimento, foi verificada uma diferença entre os resultados dos estudantes no pré e pós-teste, sendo a nota média 5,08 no primeiro e média 5,8 no segundo. Com a aplicação de um teste de hipótese sobre os resultados das avaliações, a diferença se mostrou significativa com 90% de confiança e por isso foi possível negar a hipótese nula e afirmar que o ProTesters contribuiu para o ganho de conhecimento dos jogadores sobre o processo de teste de software.

CAPÍTULO 7 - CONSIDERAÇÕES FINAIS

“O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis.”

José de Alencar

Este capítulo apresenta as considerações finais sobre a pesquisa, levando em consideração sua relevância, contribuições, limitações, ameaças a validade e trabalhos futuros.

7.1 Relevância do estudo

Este trabalho apresentou um estudo sobre a aplicação de um jogo educacional focado no ensino do processo de teste. Nele foram relatados o planejamento, preparação, execução e análise dos resultados obtidos no experimento exploratório do jogo ProTesters. Para analisar a experiência do jogo e usabilidade, foi utilizado o questionário do MEEGA+, que verifica diversos aspectos de qualidade de jogos educacionais. Já para verificar o ganho de aprendizagem dos participantes, foram aplicados pré e pós-testes para avaliar o conhecimento dos estudantes e os ganhos de aprendizagem. Foi também aplicado um teste de hipóteses para verificar se houve ganho significativo de aprendizado antes e depois do experimento com o jogo.

Considerando a importância dos testes para o desenvolvimento de software (DELAMARO, et al., 2016); (SILVA, et al., 2012) e as deficiências no ensino de teste de software apontadas anteriormente, este trabalho tem relevância por focar em um dos principais tópicos de teste de software, que é o processo de teste. Considerando que as principais deficiências no ensino de teste foram a má estruturação dos cursos de graduação (VALLE, et al., 2015); (CLEGG, et al., 2017), a falta de motivação dos

estudantes para o aprendizado do conteúdo de teste (JIA & YANG, 2013); (PASCHOAL & DE SOUZA, 2018) e as diferenças entre teoria e prática (JIA; YANG, 2013; (SCATALON, et al., 2018), pode-se considerar que a criação de um jogo educacional pode contribuir de várias formas para o ensino.

Considerando os resultados encontrados neste trabalho, foi possível verificar uma experiência positiva dos jogadores com o jogo, tanto do aspecto motivacional como educacional, tendo em vista os resultados significativos de melhora na aprendizagem do conteúdo. Além disso, o experimento com o jogo foi projetado de maneira a avaliar o aprendizado só com o jogo. Utilizando as recomendações apresentadas nas discussões, é possível utilizar o jogo de outras formas e obter melhores resultados em relação aos aspectos de aprendizagem com o jogo.

7.2 Contribuições da pesquisa

A principal contribuição deste trabalho é a criação e avaliação de uma nova proposta de jogo educacional, que é o ProTester. Esse jogo aborda de forma mais completa do que os jogos existentes, o processo de teste de software, apresentado os papéis, artefatos, atividades, situações e problemas que podem ocorrer no ambiente de teste. O jogo mostrou influenciar de maneira positiva o aprendizado dos estudantes, observando principalmente a influência significativa do jogo na avaliação de pós-teste, como também no aprendizado percebido pelos jogadores. O jogo também traz melhoras em outros aspectos além do aprendizado, como a motivação e a experiência dos jogadores, que foi medido pelo questionário MEEGA+, e a simulação do ambiente de testes de software.

Por fim, a validação do jogo ProTesters foi feita de forma exploratória, a fim de apontar os resultados do jogo em relação ao ganho de aprendizagem. Com a validação, foi possível conhecer o nível de conhecimento dos jogadores antes e depois da experiência com o jogo e com esses resultados foi verificado o ganho de aprendizado dos jogadores.

7.3 Ameaças à validade

É possível citar algumas ameaças à validade desta pesquisa e seus tratamentos baseados em Wohlin et al. (2012). Como os participantes do estudo são estudantes

de graduação com pouco experiência em testes de software e sem contato prévio com o jogo, podemos considerá-los relevantes para a avaliação de aprendizagem com o jogo. Entretanto, o experimento foi conduzido em um único contexto, e por isso é necessária a condução de mais experimentos para corroborar os resultados encontrados. Por exemplo, nas respostas ao questionário foi verificado que os participantes do experimento tinham pouca ou nenhuma experiência com testes, além de jogarem jogos de tabuleiro e de cartas com pouca frequência. É esperado que jogadores com mais experiência sobre testes, mas sem muita vivência no processo de testes, tenham mais facilidade para relacionar os elementos do jogo com seus conhecimentos. Além disso, a experiência prévia dos jogadores com jogos de tabuleiro e jogos de cartas pode facilitar a jogabilidade, permitindo com que eles foquem mais em aspectos de aprendizado e tenham uma melhor experiência com o jogo.

O material do jogo está todo em português e contendo informações suficientes para o entendimento e diferenciação de cada carta do material do jogo. Uma das ameaças não tratadas no estudo é em relação a validade ecológica do estudo. O ambiente utilizado para a aplicação do jogo foi um ambiente digital Miro. Devido às restrições da pandemia da COVID-19 não possível fazer este experimento de forma presencial, com o jogo físico e por isso ele foi aplicado no ambiente online. Tal ambiente não é o ideal para a aplicação de um jogo de tabuleiro, como o ProTesters. Como consequência disso, estudantes podem ter sentido mais dificuldade para jogar o jogo e executar as ações com o material, o que pode ter influenciado tanto no aspecto de jogabilidade do jogo como também na aprendizagem do conteúdo que estava sendo tratado. Além disso, o material do jogo, ao ser trazido para o ambiente virtual, perdeu parte de sua qualidade gráfica, embora a leitura e entendimento das cartas ainda fosse possível.

O instrumento de coleta utilizado foi utilizado conforme propõe o modelo MEEGA+, que é específico para jogos educacionais na área de computação. Apenas algumas adaptações foram realizadas no questionário em relação aos itens que avaliam os objetivos específicos de aprendizagem do jogo. O MEEGA+ possui métricas para avaliar tanto a aprendizagem como a experiência dos jogadores. Os

objetivos de aprendizagem focam na percepção de aprendizagem com o jogo, porém podem não revelar o real aprendizado com o jogo.

Como forma de obter o real ganho de aprendizagem foram realizados pré e pós-teste para verificar o conhecimento das competências do processo de teste abordados no jogo antes e depois do experimento. Com os resultados de pré e pós-teste foi feito um teste de hipótese com 90% de confiança e verificado um aumento significativo de aprendizado dos estudantes no pós-teste. Como forma de diminuir o nível de dificuldade que um teste pode ter na avaliação dos estudantes, as questões do pós-teste são iguais às do pré-teste, com exceção de uma. Apesar de isso equalizar o nível dos testes, isso também pode gerar dúvidas sobre as reflexões e discussões prévias que os estudantes tiveram com as questões no pré-teste. Como forma de minimizar isso, o único tratamento e discussões que os estudantes tiveram no período entre os testes foi o jogo e com os jogadores.

7.4 Trabalhos futuros

Com os resultados sobre a avaliação do ProTesters, foram identificadas algumas melhorias na forma de aplicação, o que deve servir de motivação para que melhorias sejam aplicadas tanto no jogo, quanto na forma de aplicação dele. Uma dos principais pontos fracos do ProTesters é a quantidade de regras do jogo, o que pode torna-lo complexo para ser explicado e aplicado pelos jogadores. Por isso novas versões simplificadas do jogo com a remoção de algumas mecânicas poderiam ser aplicadas em novos estudos, a fim de verificar a jogabilidade e a aprendizibilidade dos jogadores com diferentes versões do jogo. Além disso, o próprio *design* do jogo também pode ser futuramente alterado para a inserção de novos conteúdos ou de novas formas de abordagem do processo de teste para que melhores resultados sejam alcançados.

Considerando que o experimento ocorreu em um contexto acadêmico específico de aplicação e com poucos participantes, é interessante a condução de novos estudos em outros ambientes para corroborar os resultados apresentados e verificar a aceitação e aprendizado de estudantes com a nova proposta de jogo educacional.

REFERÊNCIAS BIBLIOGRÁFICAS

ABNT, 2001. *NBR ISO 10015:2001 - Gestão da Qualidade: diretrizes para treinamento,* Rio de Janeiro: ABNT.

ACUÑA, S. T. & FERRÉ, X., 2001. *Software Process Modelling.* s.l., s.n., pp. 237-242.

AKEN, J. E. V., 2004. Management research based on the paradigm of the design sciences: the quest for field-tested and grounded technological rules. *Journal of management studies,* 41(2), pp. 219-246.

ALHAMMAD, M. & MORENO, A., 2018. Gamification in software engineering education: A systematic mapping. *Journal of Systems and Software,* Volume 141, pp. 131-150.

AMARAL, H., BRAGA, J. L. & GALVÃO, A., 2013. *Game Architecture for teaching-learning process: An application on an undergraduate course.* s.l., IEEE, pp. 1-6.

ARAUJO, N., MACHADO, R., VIANA, D. & RIVERO, L., 2017. Avaliando a Viabilidade do BlackBox em Sala de Aula: Um Jogo Sério para Ensino de Teste Funcional de Software. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, p. 817.

BAKKER, M., VAN DEN HEUVEL-PANHUIZEN, M. & ROBITZSCH, A., 2015. Effects of playing mathematics computer games on primary school students' multiplicative reasoning ability. *Contemporary Educational Psychology,* Volume 40, pp. 55-71.

BARBOSA, A. K. T., NEVES, L. L. E. & NETO, A. C. D., 2016. *Jovetest-jogo da velha para auxiliar no ensino e estudo de teste de software.* s.l., s.n.

BASILI, V. R. & ROMBACH, H. D., 1988. The TAME project: Towards improvement-oriented software environments. *IEEE Transactions on software engineering*, pp. 758-773.

BATTISTELLA, P. & VON WANGENHEIM, C. G., 2016. Games for teaching computing in higher education–a systematic review. *IEEE Technology and Engineering Education*, pp. 8-30.

BAYAZIT, N., 2004. Investigating design: A review of forty years of design research. *Design issues,* 20(1), pp. 16-29.

BEN-ZVI, T., 2010. The efficacy of business simulation games in creating Decision Support Systems: An experimental investigation. *Decision support systems,* pp. 61-69.

BEPPE, T. A. et al., 2018. GreaTest: a card game to motivate the software testing learning. *Proceedings of the XXXII Brazilian Symposium on Software Engineering*, pp. 298-307.

BEZERRA, C. I. et al., 2014. *Evolução do jogo itest learning para o ensino de testes de software: Do planejamento ao projeto.* Fortaleza, s.n.

BHAGAT, K. K., LIOU, W. K. & CHANG, C. Y., 2016. A cost-effective interactive 3D virtual reality system applied to military live firing training. *Virtual Reality,* 20(2), pp. 127-140.

BLOOM, B. S. et al., 1956. *Taxonomy of educational objectives: The classification of educational goals. Handbook 1: Cognitive domain.* New York: David McKay.

BONWELL, C. C. & EISON, J. A., 1991. *Active Learning: Creating Excitement in the Classroom.* The George Washington University, One Dupont Circle, Suite 630, Washington, DC 20036-1183: ERIC Clearinghouse on Higher Education.

BOURGONJON, J., VALCKE, M., SOETAERT, R. & SCHELLENS, T., 2010. Students' perceptions about the use of video games in the classroom.. *Computers & Education,* pp. 1145-1156.

BRIAND, L. C., WÜST, J., DALY, J. W. & PORTER, D. V., 2000. Exploring the relationships between design measures and software quality in object-oriented systems. *Journal of systems and software,* pp. 245-273.

BROCKMYER, J. H. et al., 2009. The development of the Game Engagement Questionnaire: A measure of engagement in video game-playing. *Journal of Experimental Social Psychology,* pp. 624-634.

BUFFARDI, K. & VALDIVIA, P., 2018. Bug Hide-and-Seek: An Educational Game for Investigating Verification Accuracy in Software Tests. *IEEE Frontiers in Education Conference (FIE)*, pp. 1-8.

ÇAĞDAŞ, V. & STUBKJÆR, E., 2011. Design research for cadastral systems. *Computers, Environment and Urban Systems,* 35(1), pp. 77-87.

CLARKE, P. J., PAVA, J., WU, Y. & KING, T. M., 2011. Collaborative web-based learning of testing tools in se courses. *Proceedings of the 42nd ACM technical symposium on Computer science education*, pp. 147-152.

CLEGG, B. S., ROJAS, J. M. & FRASER, G., 2017. Teaching software testing concepts using a mutation testing game. *017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, pp. 33-36.

DAHL, O., DIJKSTRA, E. W. & HOARE, C. A. R., 1972. *Structured programming.* s.l.:Academic Press Ltd.

DE SOUSA SILVA, R. A. et al., 2020. Design and Evaluation of a Mobile Application for an Educational Card. *Journal on Interactive Systems,* pp. 110-124.

DELAMARO, M., JINO, M. & MALDONADO, J., 2016. *Introdução ao teste de software.* 2ª ed. Rio de Janeiro: ELSEVIER.

DEMILLO, R. A., LIPTON, R. J. & SAYWARD, F. G., 1978. Hints on test data selection: Help for the practicing programmer. *Computer,* pp. 34-41.

DINIZ, L. L. & DAZZI, R. L. S., 2011. Jogo para o apoio ao ensino do teste de caixa-preta. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*.

EFE, H. A. & EFE, R., 2011. Evaluating the effect of computer simulations on secondary biology instruction: An application of Bloom's taxonomy. *Scientific Research and Essays,* 6(10), pp. 2137-2146.

EKWOGE, O. M., FONTÃO, A. & DIAS-NETO, A. C., 2017. Tester Experience: Concept, Issues and Definition. *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, pp. 208-213.

ELBAUM, S., PERSON, S., DOKULIL, J. & JORDE, M., 2007. Bug hunt: Making early software testing lessons engaging and affordable. *Proceedings of the 29th international conference on Software Engineering*, pp. 688-697.

FARIAS, V., MOREIRA, C., COUTINHO, E. & SANTOS, I. S., 2012. itest learning: Um jogo para o ensino do planejamento de testes de software. *Fórum de Educação em Engenharia de Software. Simpósio Brasileiro de Engenharia de Software*.

FREZZA, S., 2002. Integrating testing and design methods for undergraduates: teaching software testing in the context of software design. *32nd Annual Frontiers in Education*.

GAROUSI, V., FELDERER, M., KUHRMANN, M. & HERKILOĞLU, K., 2017. What industry wants from academia in software testing? Hearing practitioners' opinions. *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, pp. 65-69.

GAROUSI, V., G, G., Tüzün, E. & FELDERER, M., 2019. Closing the gap between software engineering education and industrial needs. *IEEE Software 37.2,* pp. 68-77.

GAROUSI, V., RAINER, A., LAUVAS JR, P. & ARCURI, A., 2020. Software-testing education: A systematic literature mapping. *Journal of Systems and Software.*

GLASS, R. L. et al., 2006. Software Testing and Industry Needs. *IEEE Software*, pp. 55-57.

GREEN III, W. G., 2000. *Exercise alternatives for training emergency management command center staffs.* s.l.:Universal-Publishers.

GROS, B., 2007. Digital games in education: The design of games-based. *Journal of Research on Technology in Education,* Volume 40, pp. 1-23.

HATTON, L., 1997. Reexamining the fault density component size connection. *IEEE software,* pp. 89-97.

HIDI, S., 2000. An interest researcher's perspective: The effects of extrinsic and intrinsic factors on motivation. *Intrinsic and extrinsic motivation. Academic Press*, pp. 309-339.

HSIAO, H., 2007. A brief review of digital games and learning. *2007 First IEEE International Workshop on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL'07)*, pp. 124-129.

HSU, Y., LIN, C. & SHIH, J., 2013. Developing multi-player digital adventure education game with motion sensing technologies. *2013 IEEE 13th International Conference on Advanced Learning Technologies*, pp. 207-209.

HWANG, G. J., SUNG, H. Y. & HUNG, C. M., 2013. Hwang, G. J., Sung, H. Y., Hung, C. M., Yang, L. H., & Huang, I. (2013). A knowledge engineering approach to developing educational computer games for improving students' differentiating knowledge. *British journal of educational technology,* 44(2), pp. 183-196.

IEEE Computer Society, 1991. *IEEE Standard Glossary of Software Engineering Terminology (IEEE-STD-610.12-1990).* New York: IEEE.

IEEE Computer Society, 1998. *IEEE Standard for Software Test Documentation (IEEE Std 829).* New York: IEEE.

IEEE Computer Society, 2014. *SWEBOK, version 3.0: Guide to the Software Engineering Body of Knowledge.* s.l.:IEEE Computer Engineering Society.

JIA, S. & YANG, C., 2013. Teaching software testing based on cdio. *World Transactions on Engineering and Technology Education,* pp. 476-479.

JURISTO, N., MORENO, A. M. & STRIGEL, W., 2006. Software Testing Practices in Industry. *IEEE Software,* pp. 19-21.

KAPP, K. M., 2012. *The gamification of learning and instruction.* San Francisco: Wiley.

KRATHWOHL, D. R., 2002. A revision of Bloom's taxonomy: An overview. *Theory into practice,* pp. 212-218.

KRUCHTEN, P., 2004. *The rational unified process: an introduction.* s.l.:Addison-Wesley Professional.

LAPORTE, C. Y., APRIL, A. & BENCHERIF, K., 2007. Teaching software quality assurance in an undergraduate software engineering program. *Software Quality Professional,* Volume 4.

LUCCHESE, F. & RIBEIRO, B., 2009. *Conceituação de jogos digitais.* Sao Paulo: s.n.

MALDONADO, J. & BARBOSA, E., 2006. Establishing a mutation testing educational module based on IMA-CID. *Second Workshop on Mutation Analysis (Mutation 2006-ISSRE Workshops 2006)*, p. 14.

MARCHETTO, A., RICCA, F. & TONELLA, P., 2007. Empirical validation of a web fault taxonomy and its usage for fault seeding. *2007 9th IEEE International Workshop on Web Site Evolution*, pp. 31-38.

MARTINEZ, A., 2018. Use of JiTT in a graduate software testing course: an experience report. *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training*, pp. 108-115.

MATTAR, J., 2010. *Games em educação: como os nativos digitais aprendem.* s.l.:Pearson Prentice Hall.

MAYER, R. F., 2011. Multimedia learning and games. *S. Tobias, & J. D. Fletcher (Eds.), Computer games and instruction*, p. 281–305.

MEDEIROS, M. O. & SCHIMIGUEL, J., 2012. *Uma abordagem para avaliação de jogos educativos: ênfase no ensino fundamental,* s.l.: Caderno ENIC (Encontro de Iniciação Científica).

MYERS, G. J., SANDLER, C. & BADGETT, T., 2004. *The art of software testing.* s.l.:Chichester: John Wiley & Sons.

MYERS, G. J., SANDLER, C. & BADGETT, T., 2011. *The art of software testing.* s.l.:John Wiley & Sons.

NIELSEN, J., 1994. *Heuristic evaluation. In Nielsen, J., and Mack, R.L. (Eds.), Usability Inspection Methods.* New York, NY: John Wiley & Sons.

OGUZ, D. & OGUZ, K., 2019. Perspectives on the gap between the software industry and the software engineering education. *IEEE Access,* Volume 7, pp. 117527-117543.

OLIVEIRA, B. C. & COSTA, H. A., 2013. *Testeg-um software educacional para o ensino de teste de software.* s.l.:Universidade Federal de Lavras.

PASCHOAL, L. N. & DE SOUZA, S. R. S., 2018. A Survey on Software Testing Education in Brazil. *Proceedings of the 17th Brazilian Symposium on Software Quality*, pp. 334-343.

PASCHOAL, L. N. & SOUZA, S. R. S., 2018. Planejamento e aplicação de flipped classroom para o ensino de teste de software. *RENOTE,* 16(2), pp. 606-614.

PEDRÓ, F., 2006. Aprender en el nuevo milenio: Un desafío a nuestra visión de las tecnologías y la enseñanza. *París: OCDE-CERI.*

PEFFERS, K., TUUNANEN, T., ROTHENBERGER, M. A. & CHATTERJEE, S. A., 2007. A design science research methodology for information systems research. *Journal of management information systems,* pp. 45-77.

PETRI, G., VON WANGENHEIM, C. G., WANGENHEIM & BORGATTO, A., 2019. MEEGA+: Um Modelo para a Avaliação de Jogos Educacionais para o ensino de Computação. *Revista Brasileira de Informática na Educação,* 27(3), pp. 52-81.

PFLEEGER, S. L., 2004. *Engenharia de software: teoria e prática.* s.l.:Prentice Hall.

PRENKSY, M. R., 2012. *From digital natives to digital wisdom: Hopeful essays for 21st century learning.* s.l.:Corwin Press.

PRIES-HEJE, J., BASKERVILLE, R. & VENABLE, J. R., 2008. Strategies for design science research evaluation. *European Conference on Information Systems*.

QUEIROZ, R., PINTO, F. & SILVA, P., 2019. IslandTest: jogo educativo para apoiar o processo ensino-aprendizagem de testes de software. *Anais do XXVII Workshop sobre Educação em Computação*, p. Anais do XXVII Workshop sobre Educação em Computação.

RAPPS, S. & WEYUKER, E. J., 1982. Data flow analysis techniques for test data selection. *Proceedings of the 6th international conference on Software engineering. IEEE Computer Society Press*, pp. 272-278.

RATIONAL SOFTWARE CORP, 2021. *Test: Overview.* [Online]
Available at: https://sceweb.uhcl.edu/helm/RationalUnifiedProcess/
[Acesso em 30 Outubro 2021].

RIBEIRO, T. P. B. & PAIVA, A. C., 2014. *ilearntest: Jogo educativo para aprendizagem de testes de software.* s.l.:s.n.

RIOS, E. & MOREIRA, T., 2013. *Teste de software.* 3ª ed. s.l.:Alta Books Editora.

ROCHA, R. V. & ARAUJO, R. B., 2014. *Metodologia iterativa e modelos integradores para desenvolvimento de jogos sérios de treinamento e avaliação de desempenho humano.* s.l.:s.n.

ROJAS, J. M., WHITE, T. D., CLEGG, B. S. & FRASER, G., 2017. Code defenders: crowdsourcing effective tests and subtle mutants with a mutation testing game. *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, pp. 677-688.

ROSENTHAL, R. et al., 2011. Video gaming in children improves performance on a virtual reality trainer but does not yet make a laparoscopic surgeon. *Surgical innovation,* 18(2), pp. 160-170.

SÁNCHEZ, A. V., 2014. Tecnologias para a modalidade EAD: um estudo no cenário educacional atual. *Revista E-Tech: Tecnologias para Competitividade Industrial-ISSN-1983-1838,* pp. 71-104.

SAVI, R., 2011. *Avaliação de jogos voltados para a disseminação do conhecimento.* s.l.:s.n.

SAVI, R., VON WANGENHEIM, C. G. & BORGATTO, A., 2011. Um modelo de avaliação de jogos educacionais na engenharia de software. *Anais do XXV Simpósio Brasileiro de Engenharia de Software (SBES 2011)*.

SCATALON, L. P. et al., 2018. A survey on graduates' curriculum-based knowledge gaps in software testing. *2018 IEEE Frontiers in Education Conference (FIE)*, pp. 1-8.

SCHWARTZ, D. L. & BRANSFORD, J. D., 1998. A time for telling. *Cognition and instruction,* pp. 475-522.

SILVA, A. C. & THIRY, M., 2010. *Jogo educacional para apoiar o ensino de técnicas para elaboração de testes de unidade.* s.l.:s.n.

SILVA, R. A., GOMES, E. W. C. & MATOS, S. N., 2012. Plano de Teste para Validação do Subframework de Análise Semântica de Fórmulas. *CONTECSI-International Conference on Information Systems and Technology Management*, pp. 4182-4208.

SILVA, T. G. & MÜLLER, F. M., 2012. *Jogos sérios em mundos virtuais: uma abordagem para o ensino-aprendizagem de teste de software.* s.l.:s.n.

SILVA, T. G., MÜLLER, F. M. & BERNARDI, G., 2011. Panorama do ensino de engenharia de software em cursos de graduação focado em teste de software: uma proposta de aprendizagem baseada em jogos. *RENOTE.*

SIMON, H., 1996. *The science of design: creating the artificial.* s.l.:MIT Press.

SMITH, J., TESSLER, J. & KRAMER, E., 2012. Using Peer Review to Teach Software Testing. *Using Peer Review to Teach Software Testing*, pp. 93-98.

TAIPALE, O., SMOLANDER, K. & KÄLVIÄINEN, H., 2005. Finding and ranking research directions for software testing. *European Conference on Software Process Improvement*, p. 2005.

TAYLOR, L. & PARSONS, J., 2011. Improving Student Engagement. *Current issues in education,* 14(1).

TILLMANN, N. et al., 2014. Code hunt: searching for secret code for fun. *Proceedings of the 7th International Workshop on Search-Based Software Testing*, pp. 23-26.

VALLE, P. H. D., BARBOSA, E. F. & MALDONADO, J. C., 2015. CS curricula of the most relevant universities in Brazil and abroad: Perspective of software testing education. *2015 International Symposium on Computers in Education (SIIE)*, pp. 62-68.

VALLE, P. H. D., BARBOSA, E. F. & MALDONADO, J. C., 2015. Um mapeamento sistemático sobre ensino de teste de software. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, p. 71.

VALLE, P. H. D., TODA, A. M., BARBOSA, E. F. & MALDONADO, J. C., 2017. Educational games: A contribution to software testing education. *2017 IEEE Frontiers in Education Conference (FIE)*, pp. 1-8.

VERGILIO, S. R., MALDONADO, J. C. & JINO, M., 1993. Uma estratégia para a geração de dados de teste. *VII Simpósio Brasileiro de Engenharia de Software*, pp. 307-319.

VYGOTSKY, L., 1991. *A formação social da mente. Trad. José Cipolla Neto..* São Paulo: s.n.

WALSH, P. J., 1985. *A measure of test case completeness (software, engineering).* s.l.:s.n.

WERBACH, K. & HUNTER, D., 2012. *For the win: How game thinking can revolutionize your business.* s.l.:Wharton Digital Press.

WESTERMANN, K. & RUMMEL, N., 2012. Delaying instruction: Evidence from a study in a university relearning setting. *Instructional Science,* pp. 673-689.

WOHLIN, C. et al., 2012. *Experimentation in software engineering.* s.l.:Springer Science & Business Media.

WRIGHT, F. D., WHITE, D., HIRST, T. & CANN, A., 2014. Visitors and Residents. *Learning, Media,* 39(1), p. 126–141.

XIE, T., TILLMANN, N. & DE HALLEUX, J., 2013. Educational software engineering: Where software engineering, education, and gaming meet. *2013 3rd International Workshop on Games and Software Engineering: Engineering Computer Games to Enable Positive, Progressive Change (GAS)*, pp. 36-39.

ZENG, J., PARKS, S. & SHANG, J., 2020. To learn scientifically, effectively, and enjoyably: A review of educational games. *Human Behavior and Emerging Technologies,* 2(2), pp. 186-195.

ZENG, J., PARKS, S. & SHANG, J., 2020. To learn scientifically, effectively, and enjoyably: A review of educational games. *Human Behavior and Emerging Technologies,* 2(2), pp. 186-195.

ZHU, H., HALL, P. A. V. & MAY, J. H. R., 1997. Software unit test coverage and adequacy. *Acm computing surveys (csur),* pp. 366-427.

APÊNDICE A – Guia de discussões do playtesting

1) **Vocês acham que o jogo diverte?**
2) **Vocês acham que o jogo promove a competitividade?**
3) **O que vocês acharam do design do jogo?**
4) **Vocês acham que o jogo é fácil de ser aprendido e aplicado?**
5) **Vocês acham que o jogo contribui de alguma forma para o entendimento do processo de teste?**

APÊNDICE B – Material do jogo ProTesters


Atividade
Identificar eventos e artefatos que influenciarão nos testes
3
Comunicação 4
Solução de problemas 3
Artefato
Suíte de teste
Coleção de casos de teste utilizados para testar um produto
2
2
Artefato
Guia de teste
Registra padrões e boas práticas de teste a serem seguidos em um projeto
1
1
Artefato
Script de teste
Detalham o passo-a-passo e os dados de entrada necessários para executar um teste
2
2
Artefato
Log de teste
Apresenta o resultado obtido após a execução dos casos de teste
2
2
Artefato
Resultado de teste
Apresenta uma análise dos resultados de um ou mais logs de teste
2
2
Artefato
Relatório de teste
Apresenta uma visão geral dos resultados e pode fornecer recomendações para testes futuros
2
2
Artefato
Plano de teste
Define métodos e procedimentos a serem utilizados e como será a avaliação dos resultados
3
3

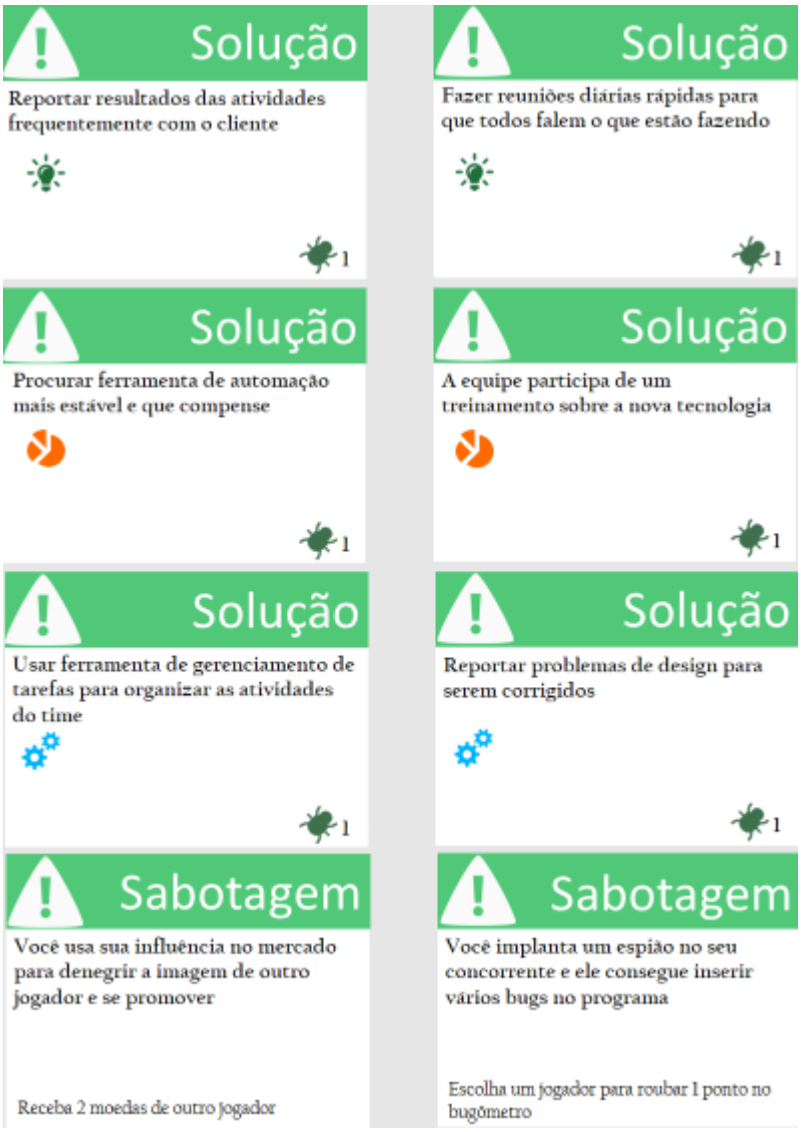

Solução
Reportar resultados das atividades frequentemente com o cliente
1
Solução
Fazer reuniões diárias rápidas para que todos falem o que estão fazendo
1
Solução
Procurar ferramenta de automação mais estável e que compense
1
Solução
A equipe participa de um treinamento sobre a nova tecnologia
1
Solução
Usar ferramenta de gerenciamento de tarefas para organizar as atividades do time
1
Solução
Reportar problemas de design para serem corrigidos
1
Sabotagem
Você usa sua influência no mercado para denegrir a imagem de outro jogador e se promover
Receba 2 moedas de outro jogador
Sabotagem
Você implanta um espião no seu concorrente e ele consegue inserir vários bugs no programa
Escolha um jogador para roubar 1 ponto no bugômetro

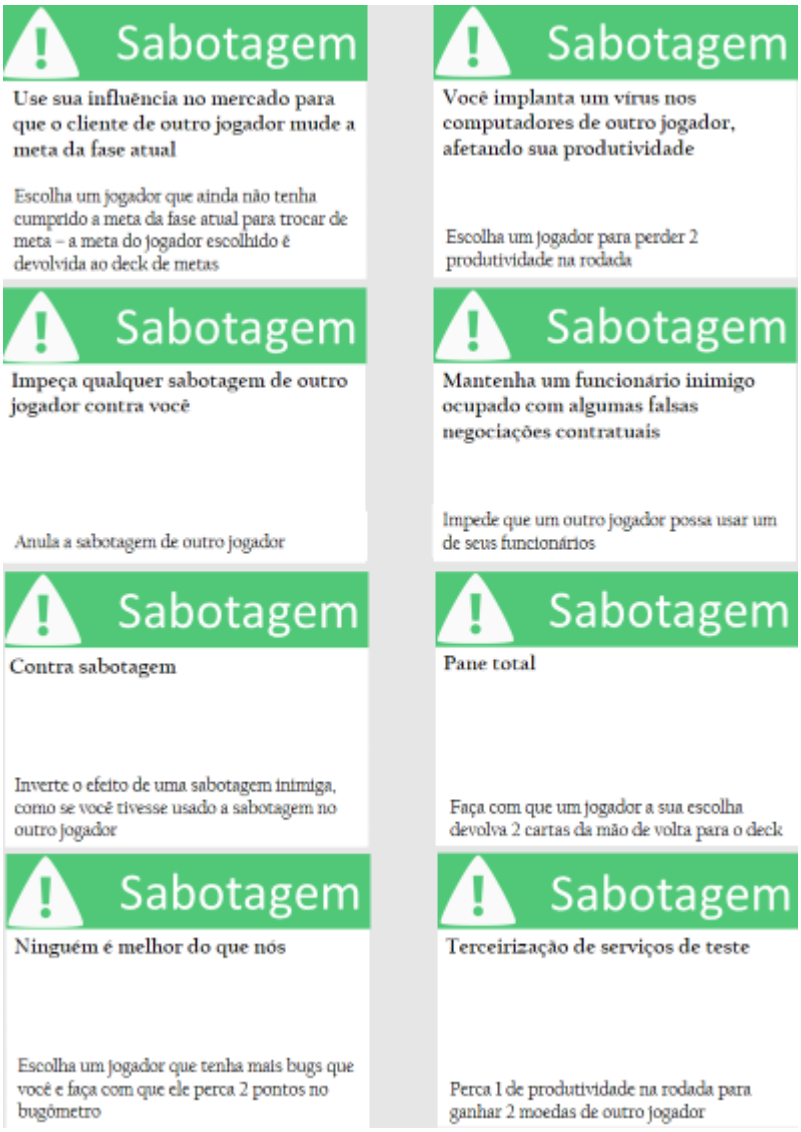

Sabotagem
Use sua influência no mercado para que o cliente de outro jogador mude a meta da fase atual
Escolha um jogador que ainda não tenha cumprido a meta da fase atual para trocar de meta – a meta do jogador escolhido é devolvida ao deck de metas
Sabotagem
Você implanta um vírus nos computadores de outro jogador, afetando sua produtividade
Escolha um jogador para perder 2 produtividade na rodada
Sabotagem
Impeça qualquer sabotagem de outro jogador contra você
Anula a sabotagem de outro jogador
Sabotagem
Mantenha um funcionário inimigo ocupado com algumas falsas negociações contratuais
Impede que um outro jogador possa usar um de seus funcionários
Sabotagem
Contra sabotagem
Inverte o efeito de uma sabotagem inimiga, como se você tivesse usado a sabotagem no outro jogador
Sabotagem
Pane total
Faça com que um jogador a sua escolha devolva 2 cartas da mão de volta para o deck
Sabotagem
Ninguém é melhor do que nós
Escolha um jogador que tenha mais bugs que você e faça com que ele perca 2 pontos no bugômetro
Sabotagem
Terceirização de serviços de teste
Perca 1 de produtividade na rodada para ganhar 2 moedas de outro jogador

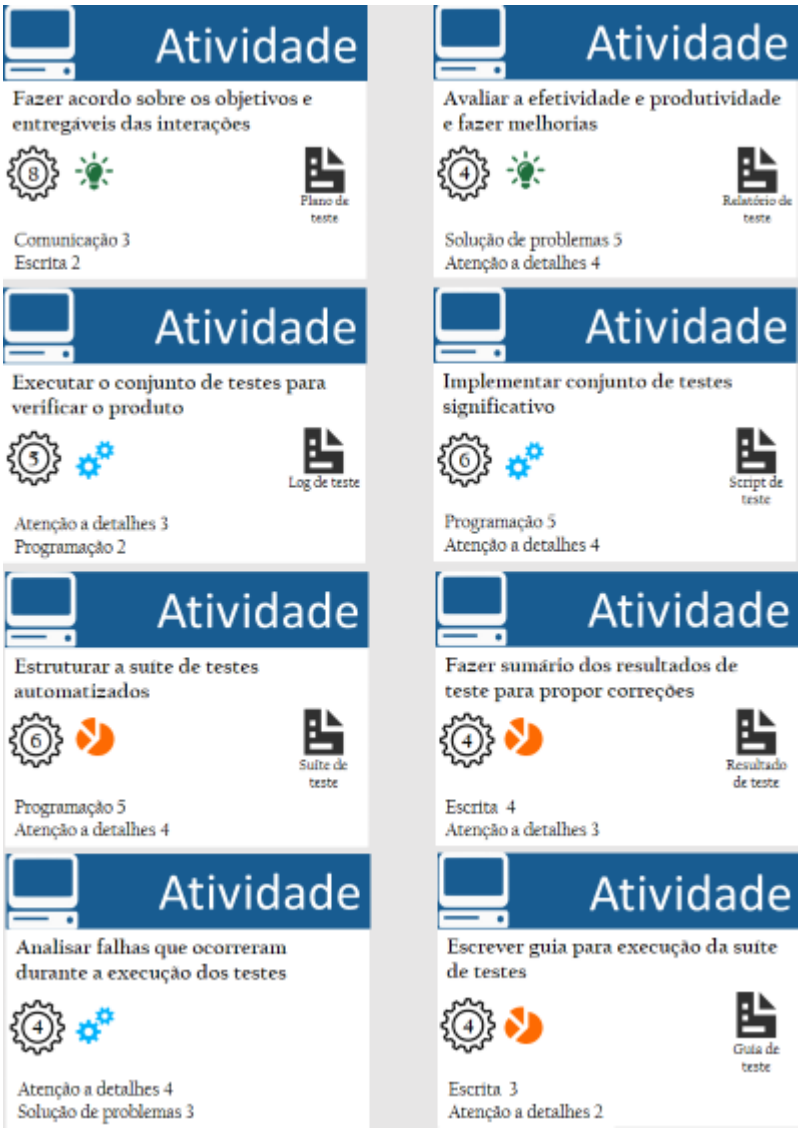

Sabotagem
Use sua influência no mercado para que o cliente de outro jogador mude a meta da fase atual
Escolha um jogador que ainda não tenha cumprido a meta da fase atual para trocar de meta – a meta do jogador escolhido é devolvida ao deck de metas
Sabotagem
Você implanta um vírus nos computadores de outro jogador, afetando sua produtividade
Escolha um jogador para perder 2 produtividade na rodada
Sabotagem
Impeça qualquer sabotagem de outro jogador contra você
Anula a sabotagem de outro jogador
Sabotagem
Mantenha um funcionário inimigo ocupado com algumas falsas negociações contratuais
Impede que um outro jogador possa usar um de seus funcionários
Sabotagem
Contra sabotagem
Inverte o efeito de uma sabotagem inimiga, como se você tivesse usado a sabotagem no outro jogador
Sabotagem
Pane total
Faça com que um jogador a sua escolha devolva 2 cartas da mão de volta para o deck
Sabotagem
Ninguém é melhor do que nós
Escolha um jogador que tenha mais bugs que você e faça com que ele perca 2 pontos no bugômetro
Sabotagem
Terceirização de serviços de teste
Perca 1 de produtividade na rodada para ganhar 2 moedas de outro jogador


Atividade
Fazer acordo sobre os objetivos e entregáveis das interações
8
Plano de teste
Comunicação 3
Escrita 2
Atividade
Avaliar a efetividade e produtividade e fazer melhorias
4
Relatório de teste
Solução de problemas 5
Atenção a detalhes 4
Atividade
Executar o conjunto de testes para verificar o produto
5
Log de teste
Atenção a detalhes 3
Programação 2
Atividade
Implementar conjunto de testes significativo
6
Script de teste
Programação 5
Atenção a detalhes 4
Atividade
Estruturar a suíte de testes automatizados
6
Suíte de teste
Programação 5
Atenção a detalhes 4
Atividade
Fazer sumário dos resultados de teste para propor correções
4
Resultado de teste
Escrita 4
Atenção a detalhes 3
Atividade
Analisar falhas que ocorreram durante a execução dos testes
4
Atenção a detalhes 4
Solução de problemas 3
Atividade
Escrever guia para execução da suíte de testes
4
Guia de teste
Escrita 3
Atenção a detalhes 2

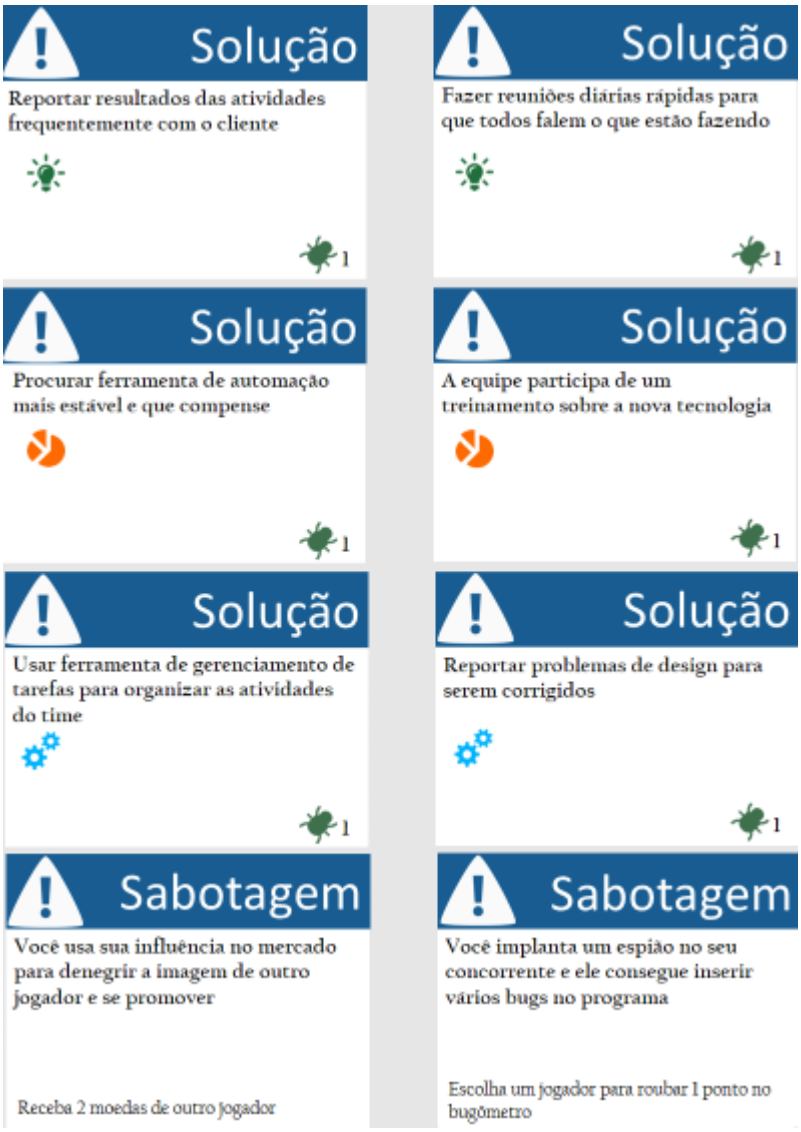

Atividade
Identificar eventos e artefatos que influenciarão nos testes
3
Comunicação 4
Solução de problemas 3
Artefato
Suíte de teste
Coleção de casos de teste utilizados para testar um produto
2
2
Artefato
Guia de teste
Registra padrões e boas práticas de teste a serem seguidos em um projeto
1
1
Artefato
Script de teste
Detalham o passo-a-passo e os dados de entrada necessários para executar um teste
2
2
Artefato
Log de teste
Apresenta o resultado obtido após a execução dos casos de teste
2
2
Artefato
Resultado de teste
Apresenta uma análise dos resultados de um ou mais logs de teste
2
2
Artefato
Relatório de teste
Apresenta uma visão geral dos resultados e pode fornecer recomendações para testes futuros
2
2
Artefato
Plano de teste
Define métodos e procedimentos a serem utilizados e como será a avaliação dos resultados
3
3


Solução
Reportar resultados das atividades frequentemente com o cliente
1
Solução
Fazer reuniões diárias rápidas para que todos falem o que estão fazendo
1
Solução
Procurar ferramenta de automação mais estável e que compense
1
Solução
A equipe participa de um treinamento sobre a nova tecnologia
1
Solução
Usar ferramenta de gerenciamento de tarefas para organizar as atividades do time
1
Solução
Reportar problemas de design para serem corrigidos
1
Sabotagem
Você usa sua influência no mercado para denegrir a imagem de outro jogador e se promover
Receba 2 moedas de outro jogador
Sabotagem
Você implanta um espião no seu concorrente e ele consegue inserir vários bugs no programa
Escolha um jogador para roubar 1 ponto no bugômetro


Sabotagem
Use sua influência no mercado para que o cliente de outro jogador mude a meta da fase atual
Escolha um jogador que ainda não tenha cumprido a meta da fase atual para trocar de meta – a meta do jogador escolhido é devolvida ao deck de metas
Sabotagem
Você implanta um vírus nos computadores de outro jogador, afetando sua produtividade
Escolha um jogador para perder 2 produtividade na rodada
Sabotagem
Impeça qualquer sabotagem de outro jogador contra você
Anula a sabotagem de outro jogador
Sabotagem
Mantenha um funcionário inimigo ocupado com algumas falsas negociações contratuais
Impede que um outro jogador possa usar um de seus funcionários
Sabotagem
Contra sabotagem
Inverte o efeito de uma sabotagem inimiga, como se você tivesse usado a sabotagem no outro jogador
Sabotagem
Pane total
Faça com que um jogador a sua escolha devolva 2 cartas da mão de volta para o deck
Sabotagem
Ninguém é melhor do que nós
Escolha um jogador que tenha mais bugs que você e faça com que ele perca 2 pontos no bugômetro
Sabotagem
Terceirização de serviços de teste
Perca 1 de produtividade na rodada para ganhar 2 moedas de outro jogador


Atividade
Fazer acordo sobre os objetivos e entregáveis das interações
8
Plano de teste
Comunicação 3
Escrita 2
Atividade
Avaliar a efetividade e produtividade e fazer melhorias
4
Relatório de teste
Solução de problemas 5
Atenção a detalhes 4
Atividade
Executar o conjunto de testes para verificar o produto
5
Log de teste
Atenção a detalhes 3
Programação 2
Atividade
Implementar conjunto de testes significativo
6
Script de teste
Programação 5
Atenção a detalhes 4
Atividade
Estruturar a suíte de testes automatizados
6
Suíte de teste
Programação 5
Atenção a detalhes 4
Atividade
Fazer sumário dos resultados de teste para propor correções
4
Resultado de teste
Escrita 4
Atenção a detalhes 3
Atividade
Analisar falhas que ocorreram durante a execução dos testes
4
Atenção a detalhes 4
Solução de problemas 3
Atividade
Escrever guia para execução da suíte de testes
4
Guia de teste
Escrita 3
Atenção a detalhes 2

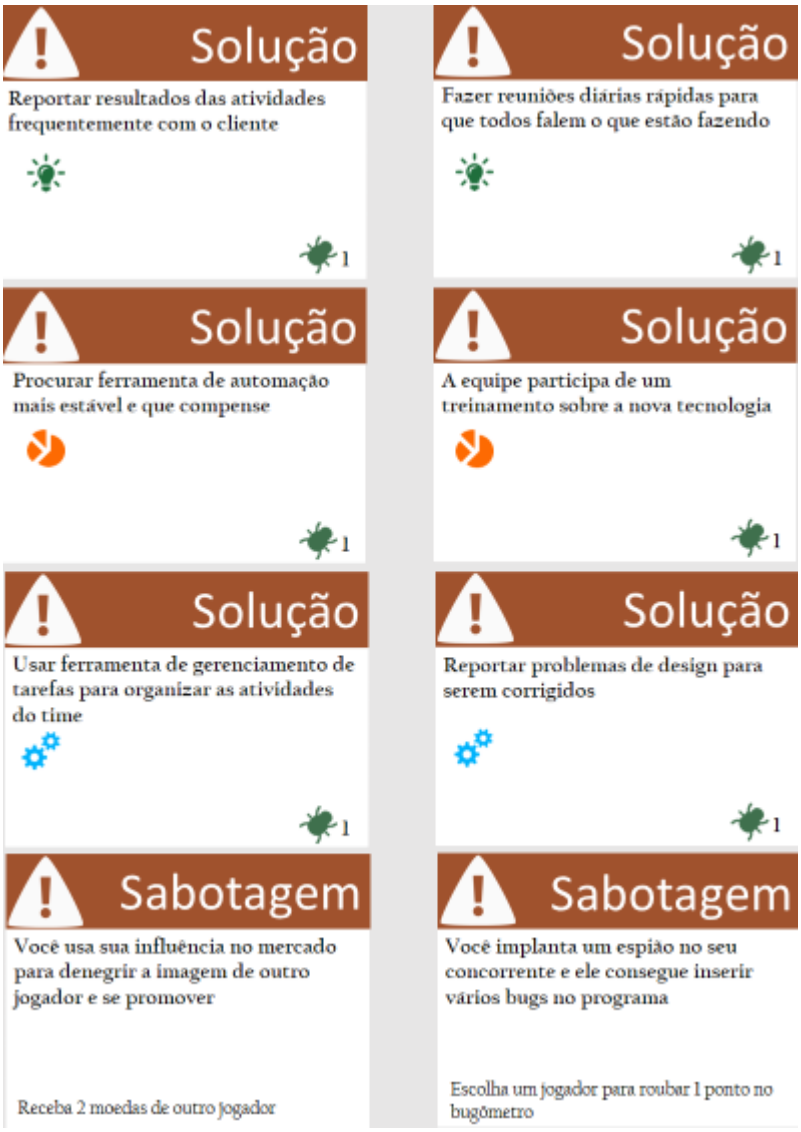

Atividade
Identificar eventos e artefatos que influenciarão nos testes
3
Comunicação 4
Solução de problemas 3
Artefato
Suíte de teste
Coleção de casos de teste utilizados para testar um produto
2
2
Artefato
Guia de teste
Registra padrões e boas práticas de teste a serem seguidos em um projeto
1
1
Artefato
Script de teste
Detalham o passo-a-passo e os dados de entrada necessários para executar um teste
2
2
Artefato
Log de teste
Apresenta o resultado obtido após a execução dos casos de teste
2
2
Artefato
Resultado de teste
Apresenta uma análise dos resultados de um ou mais logs de teste
2
2
Artefato
Relatório de teste
Apresenta uma visão geral dos resultados e pode fornecer recomendações para testes futuros
2
2
Artefato
Plano de teste
Define métodos e procedimentos a serem utilizados e como será a avaliação dos resultados
3
3

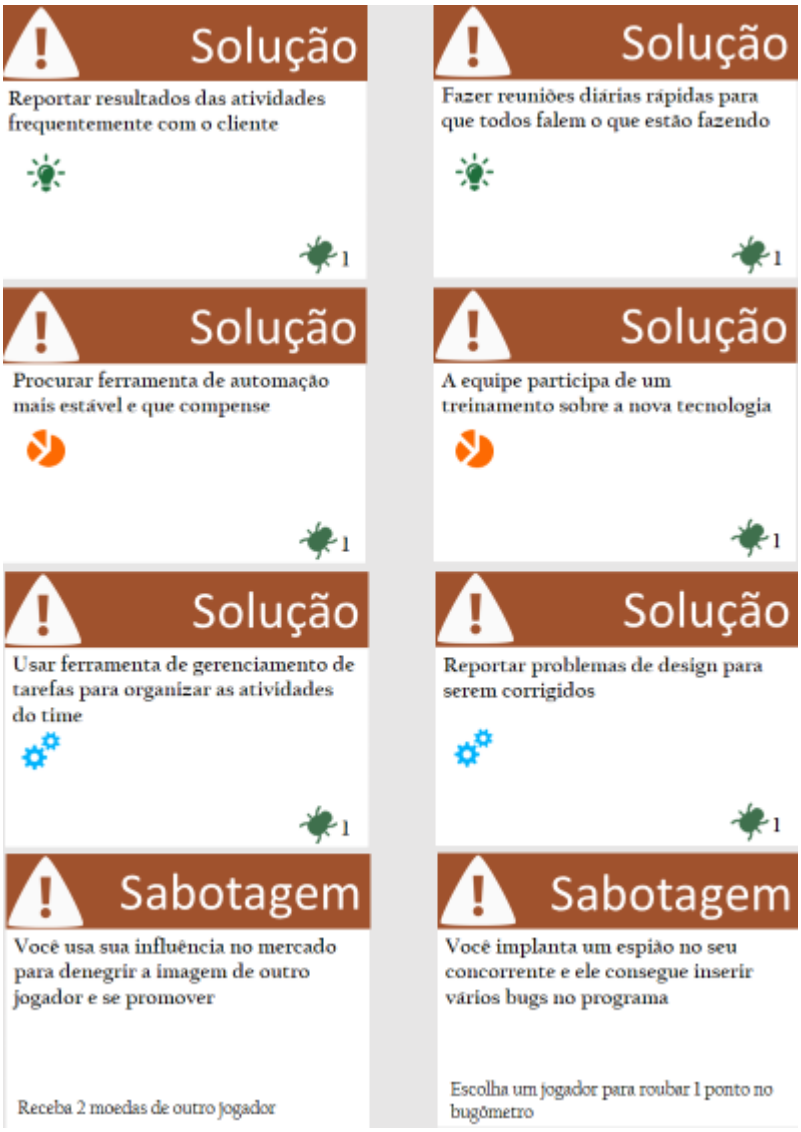

Solução
Reportar resultados das atividades frequentemente com o cliente
1
Solução
Fazer reuniões diárias rápidas para que todos falem o que estão fazendo
1
Solução
Procurar ferramenta de automação mais estável e que compense
1
Solução
A equipe participa de um treinamento sobre a nova tecnologia
1
Solução
Usar ferramenta de gerenciamento de tarefas para organizar as atividades do time
1
Solução
Reportar problemas de design para serem corrigidos
1
Sabotagem
Você usa sua influência no mercado para denegrir a imagem de outro jogador e se promover
Receba 2 moedas de outro jogador
Sabotagem
Você implanta um espião no seu concorrente e ele consegue inserir vários bugs no programa
Escolha um jogador para roubar 1 ponto no bugômetro


Sabotagem
Use sua influência no mercado para que o cliente de outro jogador mude a meta da fase atual
Escolha um jogador que ainda não tenha cumprido a meta da fase atual para trocar de meta – a meta do jogador escolhido é devolvida ao deck de metas
Sabotagem
Você implanta um vírus nos computadores de outro jogador, afetando sua produtividade
Escolha um jogador para perder 2 produtividade na rodada
Sabotagem
Impeça qualquer sabotagem de outro jogador contra você
Anula a sabotagem de outro jogador
Sabotagem
Mantenha um funcionário inimigo ocupado com algumas falsas negociações contratuais
Impede que um outro jogador possa usar um de seus funcionários
Sabotagem
Contra sabotagem
Inverte o efeito de uma sabotagem inimiga, como se você tivesse usado a sabotagem no outro jogador
Sabotagem
Pane total
Faça com que um jogador a sua escolha devolva 2 cartas da mão de volta para o deck
Sabotagem
Ninguém é melhor do que nós
Escolha um jogador que tenha mais bugs que você e faça com que ele perca 2 pontos no bugômetro
Sabotagem
Terceirização de serviços de teste
Perca 1 de produtividade na rodada para ganhar 2 moedas de outro jogador


Atividade
Fazer acordo sobre os objetivos e entregáveis das interações
8
Plano de teste
Comunicação 3
Escrita 2
Atividade
Avaliar a efetividade e produtividade e fazer melhorias
4
Relatório de teste
Solução de problemas 5
Atenção a detalhes 4
Atividade
Executar o conjunto de testes para verificar o produto
5
Log de teste
Atenção a detalhes 3
Programação 2
Atividade
Implementar conjunto de testes significativo
6
Script de teste
Programação 5
Atenção a detalhes 4
Atividade
Estruturar a suíte de testes automatizados
6
Suíte de teste
Programação 5
Atenção a detalhes 4
Atividade
Fazer sumário dos resultados de teste para propor correções
4
Resultado de teste
Escrita 4
Atenção a detalhes 3
Atividade
Analisar falhas que ocorreram durante a execução dos testes
4
Atenção a detalhes 4
Solução de problemas 3
Atividade
Escrever guia para execução da suíte de testes
4
Guia de teste
Escrita 3
Atenção a detalhes 2

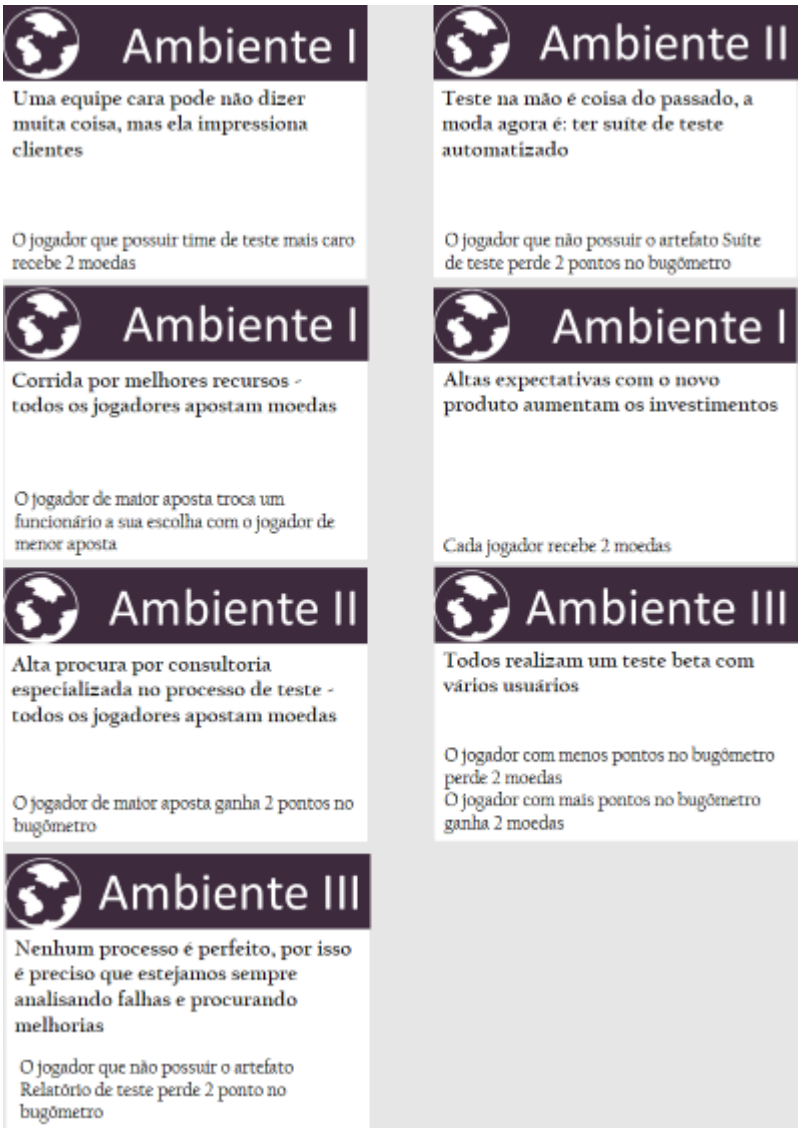

Atividade
Identificar eventos e artefatos que influenciarão nos testes
3
Comunicação 4
Solução de problemas 3
Artefato
Suíte de teste
Coleção de casos de teste utilizados para testar um produto
2
2
Artefato
Guia de teste
Registra padrões e boas práticas de teste a serem seguidos em um projeto
1
1
Artefato
Script de teste
Detalham o passo-a-passo e os dados de entrada necessários para executar um teste
2
2
Artefato
Log de teste
Apresenta o resultado obtido após a execução dos casos de teste
2
2
Artefato
Resultado de teste
Apresenta uma análise dos resultados de um ou mais logs de teste
2
2
Artefato
Relatório de teste
Apresenta uma visão geral dos resultados e pode fornecer recomendações para testes futuros
2
2
Artefato
Plano de teste
Define métodos e procedimentos a serem utilizados e como será a avaliação dos resultados
3
3


Solução
Reportar resultados das atividades frequentemente com o cliente
1
Solução
Fazer reuniões diárias rápidas para que todos falem o que estão fazendo
1
Solução
Procurar ferramenta de automação mais estável e que compense
1
Solução
A equipe participa de um treinamento sobre a nova tecnologia
1
Solução
Usar ferramenta de gerenciamento de tarefas para organizar as atividades do time
1
Solução
Reportar problemas de design para serem corrigidos
1
Sabotagem
Você usa sua influência no mercado para denegrir a imagem de outro jogador e se promover
Receba 2 moedas de outro jogador
Sabotagem
Você implanta um espião no seu concorrente e ele consegue inserir vários bugs no programa
Escolha um jogador para roubar 1 ponto no bugômetro


Solução
Reportar resultados das atividades frequentemente com o cliente
1
Solução
Fazer reuniões diárias rápidas para que todos falem o que estão fazendo
1
Solução
Procurar ferramenta de automação mais estável e que compense
1
Solução
A equipe participa de um treinamento sobre a nova tecnologia
1
Solução
Usar ferramenta de gerenciamento de tarefas para organizar as atividades do time
1
Solução
Reportar problemas de design para serem corrigidos
1
Sabotagem
Você usa sua influência no mercado para denegrir a imagem de outro jogador e se promover
Receba 2 moedas de outro jogador
Sabotagem
Você implanta um espião no seu concorrente e ele consegue inserir vários bugs no programa
Escolha um jogador para roubar 1 ponto no bugômetro


Sabotagem
Use sua influência no mercado para que o cliente de outro jogador mude a meta da fase atual
Escolha um jogador que ainda não tenha cumprido a meta da fase atual para trocar de meta – a meta do jogador escolhido é devolvida ao deck de metas
Sabotagem
Você implanta um vírus nos computadores de outro jogador, afetando sua produtividade
Escolha um jogador para perder 2 produtividade na rodada
Sabotagem
Impeça qualquer sabotagem de outro jogador contra você
Anula a sabotagem de outro jogador
Sabotagem
Mantenha um funcionário inimigo ocupado com algumas falsas negociações contratuais
Impede que um outro jogador possa usar um de seus funcionários
Sabotagem
Contra sabotagem
Inverte o efeito de uma sabotagem inimiga, como se você tivesse usado a sabotagem no outro jogador
Sabotagem
Pane total
Faça com que um jogador a sua escolha devolva 2 cartas da mão de volta para o deck
Sabotagem
Ninguém é melhor do que nós
Escolha um jogador que tenha mais bugs que você e faça com que ele perca 2 pontos no bugômetro
Sabotagem
Terceirização de serviços de teste
Perca 1 de produtividade na rodada para ganhar 2 moedas de outro jogador


Ambiente I
A comunicação hierárquica da empresa dificulta a comunicação com o cliente
Comunicação -2
Ambiente I
Há um desentendimento na equipe sobre as atividades sendo realizadas
Comunicação -1
Solução de problemas -1
Ambiente II
A ferramenta de automação de testes está apresentando problemas
Solução de problemas -1
Programação -1
Ambiente II
A equipe não tem experiência com as tecnologias utilizadas
Programação -1
Solução de problemas -1
Ambiente III
O time recebe várias solicitações de tarefas de alta de prioridade
Atenção a detalhes -1
Escrita -1
Ambiente III
O produto tem muitos problemas de design
Solução de problemas -1
Atenção a detalhes -1
Ambiente II
Qualquer coisa serve para quem não planeja nada
Jogador que não possuir o artefato Plano de teste perde 2 pontos de bugômetro
Ambiente III
Reclamações por previsão de atrasos causados pelos testes
O jogador com menos atividades realizadas perde 2 moedas e 1 ponto no bugômetro

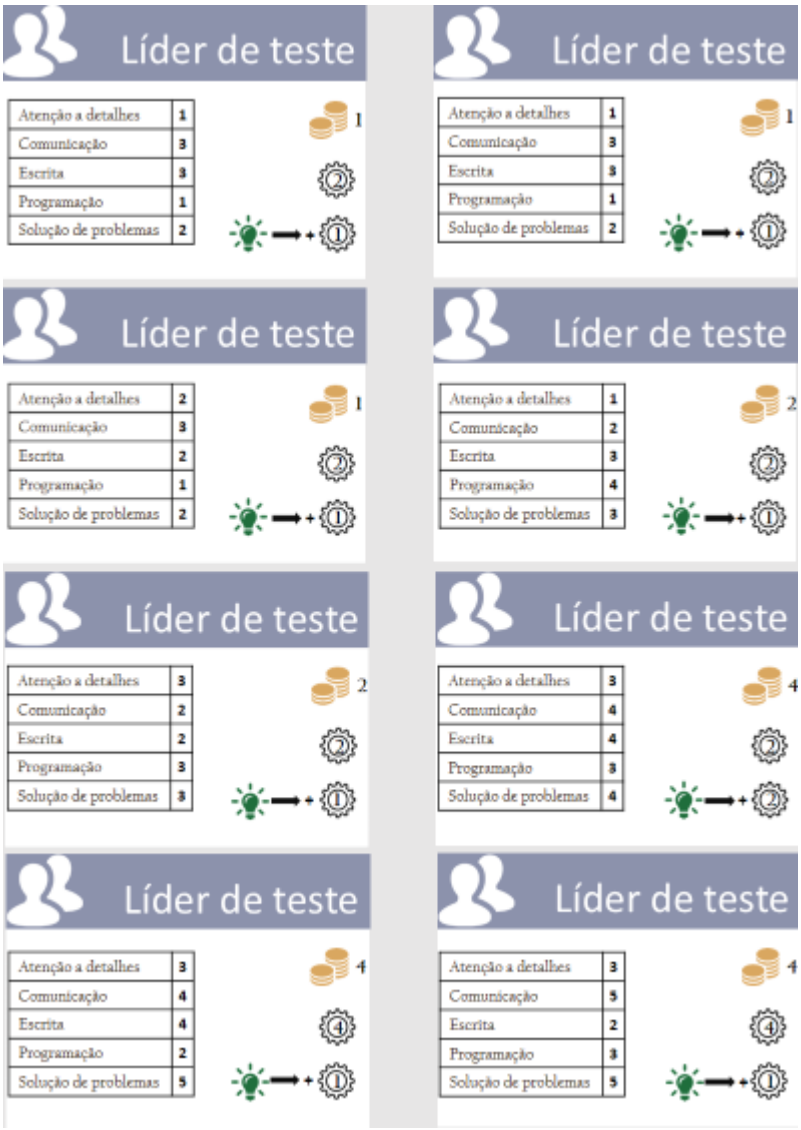

Ambiente I
Uma equipe cara pode não dizer muita coisa, mas ela impressiona clientes
O jogador que possuir time de teste mais caro recebe 2 moedas
Ambiente II
Teste na mão é coisa do passado, a moda agora é: ter suíte de teste automatizado
O jogador que não possuir o artefato Suíte de teste perde 2 pontos no bugômetro
Ambiente I
Corrida por melhores recursos - todos os jogadores apostam moedas
O jogador de maior aposta troca um funcionário a sua escolha com o jogador de menor aposta
Ambiente I
Altas expectativas com o novo produto aumentam os investimentos
Cada jogador recebe 2 moedas
Ambiente II
Alta procura por consultoria especializada no processo de teste - todos os jogadores apostam moedas
O jogador de maior aposta ganha 2 pontos no bugômetro
Ambiente III
Todos realizam um teste beta com vários usuários
O jogador com menos pontos no bugômetro perde 2 moedas
O jogador com mais pontos no bugômetro ganha 2 moedas
Ambiente III
Nenhum processo é perfeito, por isso é preciso que estejamos sempre analisando falhas e procurando melhorias
O jogador que não possuir o artefato Relatório de teste perde 2 ponto no bugômetro

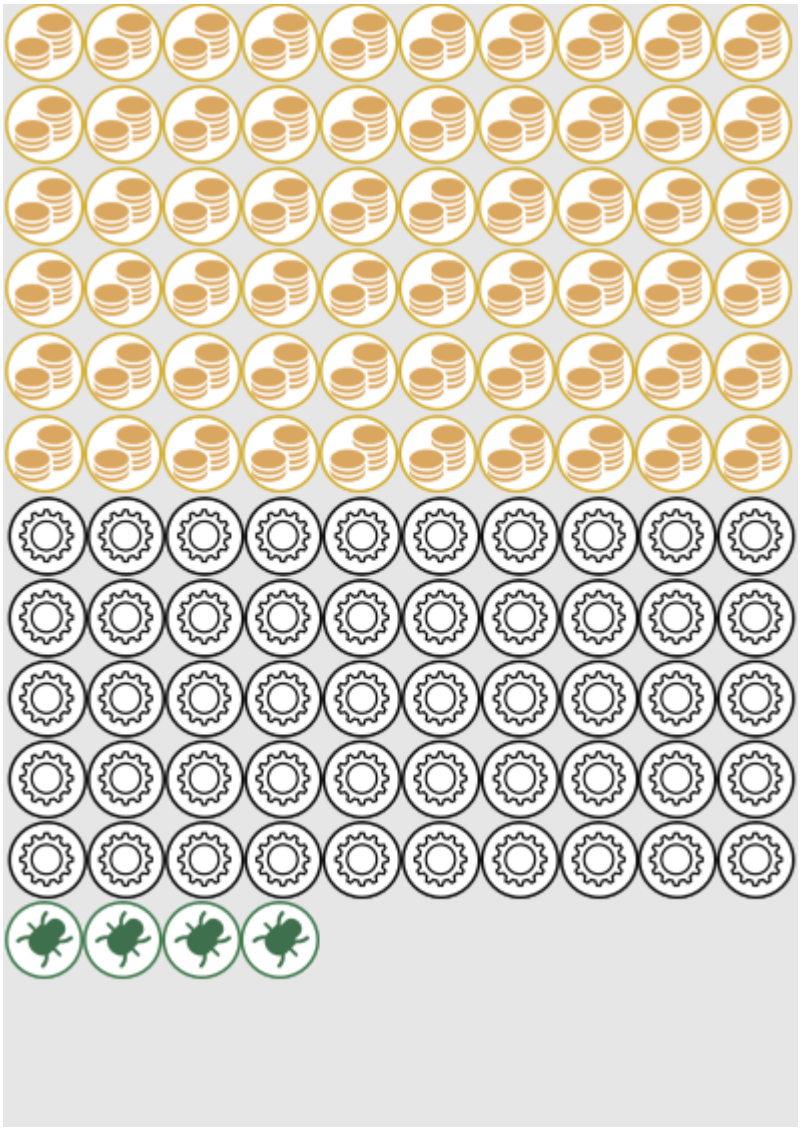

Meta I
Obter o artefato Plano de teste e realizar pelo menos uma atividade de preparação
2
Meta I
Fazer duas atividades de planejamento
2
Meta II
Obter o artefato Guia de teste e o artefato Script de teste
2
Meta III
Terminar todas as atividades de teste
2
Meta I
Realizar duas atividades de planejamento e obter o artefato Plano de teste
2
Meta I
Concluir pelo menos 2 atividades planejamento e 1 de preparação
2
Meta II
Obter o artefato Script de teste e o artefato Suíte de teste
2
Meta III
Possuir mais de 8 moedas
2

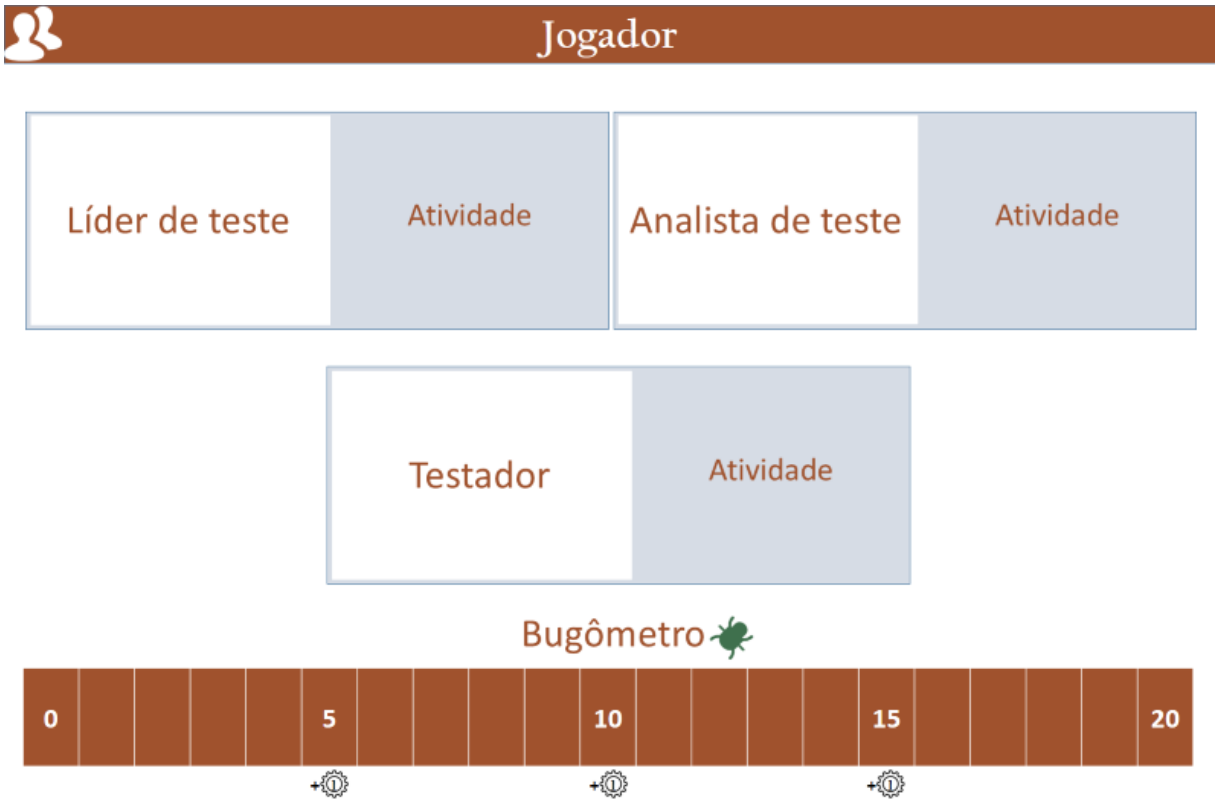

Meta I
Obter o artefato Plano de teste e o artefato Guia de teste
2
Meta III
Concluir todas as atividades de execução e obter o artefato Relatório de teste
2
Meta II
Realizar todas as atividades de preparação
2
Meta II
Concluir 2 atividades de preparação e 1 atividade de execução
2
Meta II
Obter os artefatos Suíte de teste e Guia de teste
2
Meta III
Concluir todas as atividades de execução
2
Meta III
Obter os artefatos Resultado de teste e Relatório de teste
2

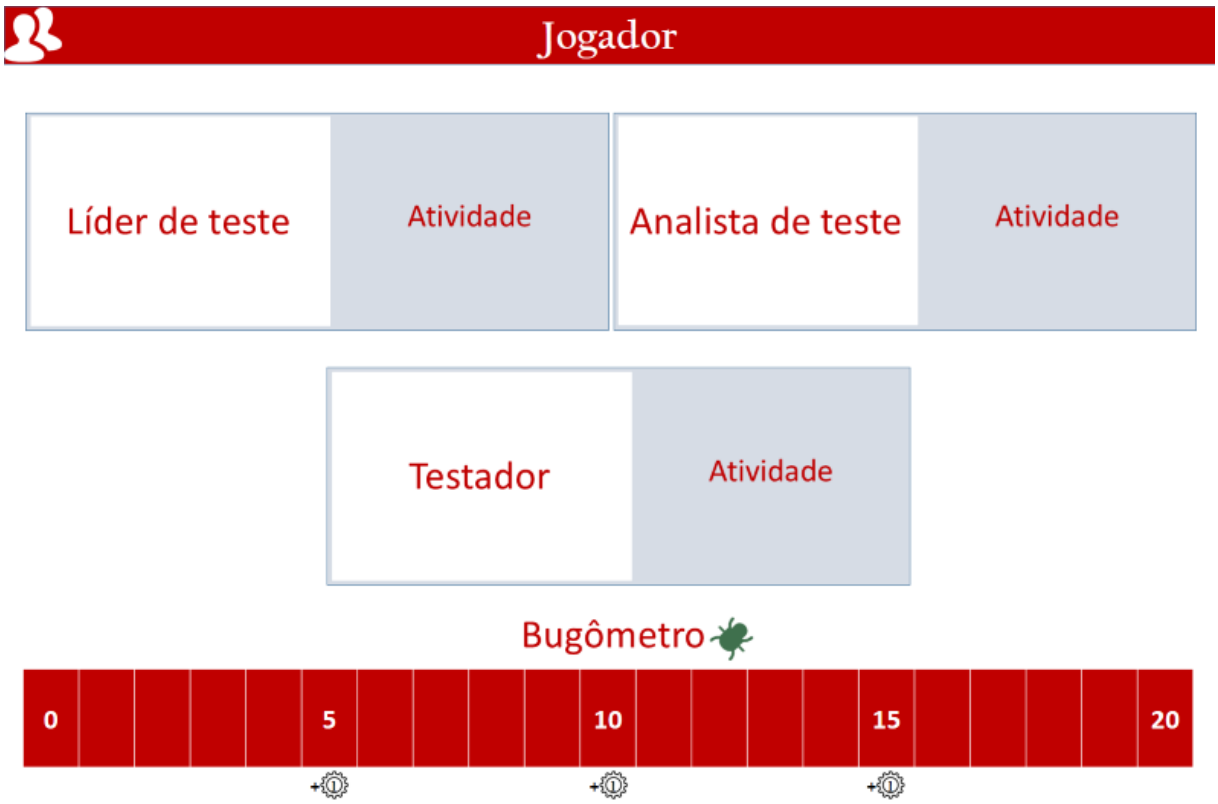

Testador
Atenção a detalhes 2
Comunicação 1
Escrita 1
Programação 3
Solução de problemas 3
1
Testador
Atenção a detalhes 3
Comunicação 1
Escrita 1
Programação 3
Solução de problemas 2
1
Testador
Atenção a detalhes 3
Comunicação 1
Escrita 1
Programação 2
Solução de problemas 3
1
Testador
Atenção a detalhes 3
Comunicação 3
Escrita 2
Programação 2
Solução de problemas 3
2
Testador
Atenção a detalhes 2
Comunicação 1
Escrita 2
Programação 3
Solução de problemas 5
2
Testador
Atenção a detalhes 5
Comunicação 2
Escrita 2
Programação 5
Solução de problemas 4
4
Testador
Atenção a detalhes 4
Comunicação 3
Escrita 2
Programação 4
Solução de problemas 5
4
Testador
Atenção a detalhes 5
Comunicação 2
Escrita 2
Programação 4
Solução de problemas 5
4

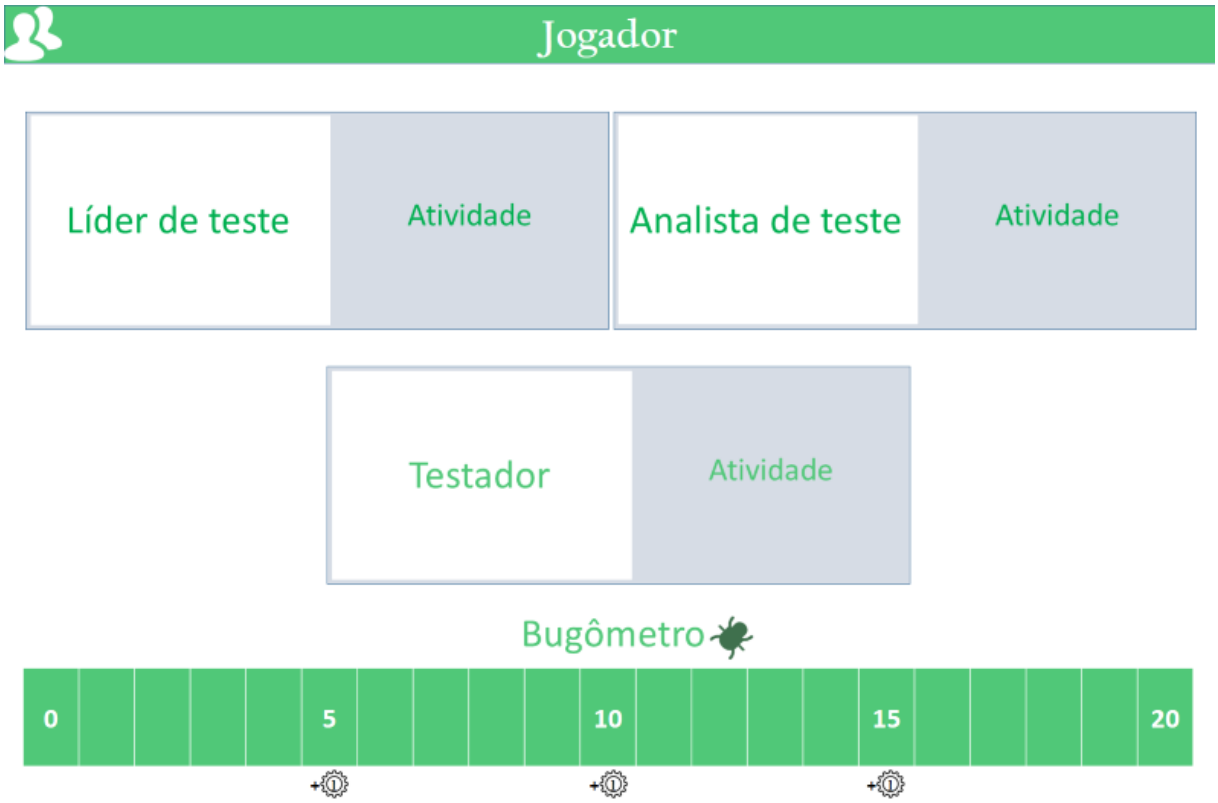

Analista de teste
Atenção a detalhes 3
Comunicação 1
Escrita 2
Programação 3
Solução de problemas 1
1
Analista de teste
Atenção a detalhes 3
Comunicação 1
Escrita 3
Programação 2
Solução de problemas 1
1
Analista de teste
Atenção a detalhes 2
Comunicação 1
Escrita 3
Programação 3
Solução de problemas 1
1
Analista de teste
Atenção a detalhes 3
Comunicação 2
Escrita 3
Programação 3
Solução de problemas 2
2
Analista de teste
Atenção a detalhes 3
Comunicação 2
Escrita 4
Programação 2
Solução de problemas 2
2
Analista de teste
Atenção a detalhes 4
Comunicação 3
Escrita 4
Programação 5
Solução de problemas 3
4
Analista de teste
Atenção a detalhes 4
Comunicação 3
Escrita 4
Programação 5
Solução de problemas 3
4
Analista de teste
Atenção a detalhes 5
Comunicação 2
Escrita 4
Programação 5
Solução de problemas 3
4

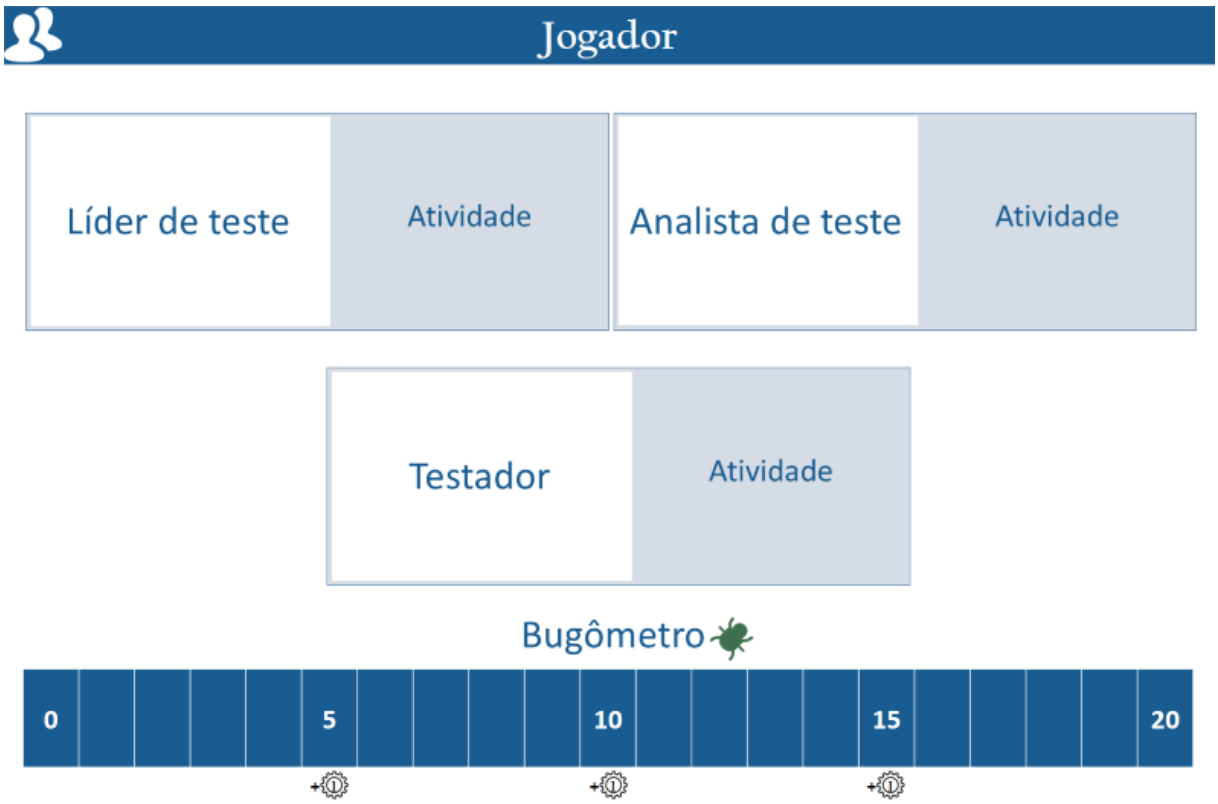

Líder de teste
Atenção a detalhes 1
Comunicação 3
Escrita 3
Programação 1
Solução de problemas 2
1
Líder de teste
Atenção a detalhes 1
Comunicação 3
Escrita 3
Programação 1
Solução de problemas 2
1
Líder de teste
Atenção a detalhes 2
Comunicação 3
Escrita 2
Programação 1
Solução de problemas 2
1
Líder de teste
Atenção a detalhes 1
Comunicação 2
Escrita 3
Programação 4
Solução de problemas 3
2
Líder de teste
Atenção a detalhes 3
Comunicação 2
Escrita 2
Programação 3
Solução de problemas 3
2
Líder de teste
Atenção a detalhes 3
Comunicação 4
Escrita 4
Programação 3
Solução de problemas 4
4
Líder de teste
Atenção a detalhes 3
Comunicação 4
Escrita 4
Programação 2
Solução de problemas 5
4
Líder de teste
Atenção a detalhes 3
Comunicação 5
Escrita 2
Programação 3
Solução de problemas 5
4




Jogador
Líder de teste
Atividade
Analista de teste
Atividade
Testador
Atividade
Bugômetro
0
5
10
15
20


Jogador
Líder de teste
Atividade
Analista de teste
Atividade
Testador
Atividade
Bugômetro
0
5
10
15
20


Jogador
Líder de teste
Atividade
Analista de teste
Atividade
Testador
Atividade
Bugômetro
0
5
10
15
20


Jogador
Líder de teste
Atividade
Analista de teste
Atividade
Testador
Atividade
Bugômetro
0
5
10
15
20

APÊNDICE C – Manual do jogo ProTesters

ProTesters

Livro de regras

Resumo do jogo

ProTesters é um jogo de tabuleiro jogado em rodadas, no qual os jogadores trabalham a atribuição de atividades, solução de problemas e simulam a execução do processo de teste.

ProTesters é um jogo competitivo que aborda o ambiente de desenvolvimento de *software*, em que empresas têm que colocar seus produtos no mercado com rapidez e qualidade. Os jogadores são responsáveis por gerenciar uma equipe para executar o processo de teste para testar um novo produto que deverá ser inserido no mercado e garantir a qualidade do produto.

Os jogadores deverão realizar as atividades de teste, seguindo o processo de teste e garantindo a qualidade do produto. Os jogadores serão responsáveis pela montagem da equipe de testes, pela atribuição e ordenamento das atividades, produção de artefatos e resolução de problemas que podem ocorrer durante os testes.

Com isso, ganhará o jogo quem melhor executar as atividades do processo dentro de 6 rodadas de jogo.

Informações de jogo

Quantidade de jogadores	2 - 4 jogadores
Idade sugerida	Maiores de 10 anos
Público-alvo	Estudantes da área de computação
Tempo de jogo	1-2 horas

Baixe o material

Cartas (A4):
https://drive.google.com/file/d/1yuAv0PoZCI0xogi53eKHrXk2inOObLf-/view?usp=sharing

Cartões de jogadores (A3):
https://drive.google.com/file/d/1CAexNBw194cI_kxaj37ImNIDRfKY3xaf/view?usp=sharing

Componentes do jogo

4 cartões de jogadores

36 cartas de atividade
(9 para cada jogador)

28 cartas de artefato
(7 para cada jogador)

64 cartas do *deck*
(16 para cada jogador)

15 cartas de ambiente
(divididos em 3 níveis)

15 cartas de meta
(divididos em 3 níveis)

24 cartas de
funcionário
(8 para cada papel)

50 recursos de moeda

50 recursos de esforço

4 marcadores de
bugômetro

Como jogar

ProTesters é dividido em 3 fases principais, que dividem a jogabilidade do jogo, que são: preparação, contratações e execução das rodadas.

Preparação

Antes de jogar uma partida, siga os próximos passos em ordem. O diagrama de preparação na página 5 mostra um exemplo desses passos.

1. **Escolha do time**
Cada jogador deve selecionar a cor do seu time, que pode ser amarelo, vermelho, verde e marrom. Ao selecionar o time, o jogador também recebe as cartas de atividades e artefatos do time.

2. **Recursos iniciais**
Cada jogador ganha 7 moedas logo no início do jogo, que deverão ser usadas na fase de contratação.

3. **Embaralhar cartas de meta**
Os jogadores devem embaralhar as cartas de meta pelo número, por exemplo: cartas Meta I serão embaralhadas em conjunto e as cartas Meta II serão embaralhadas em outro conjunto.

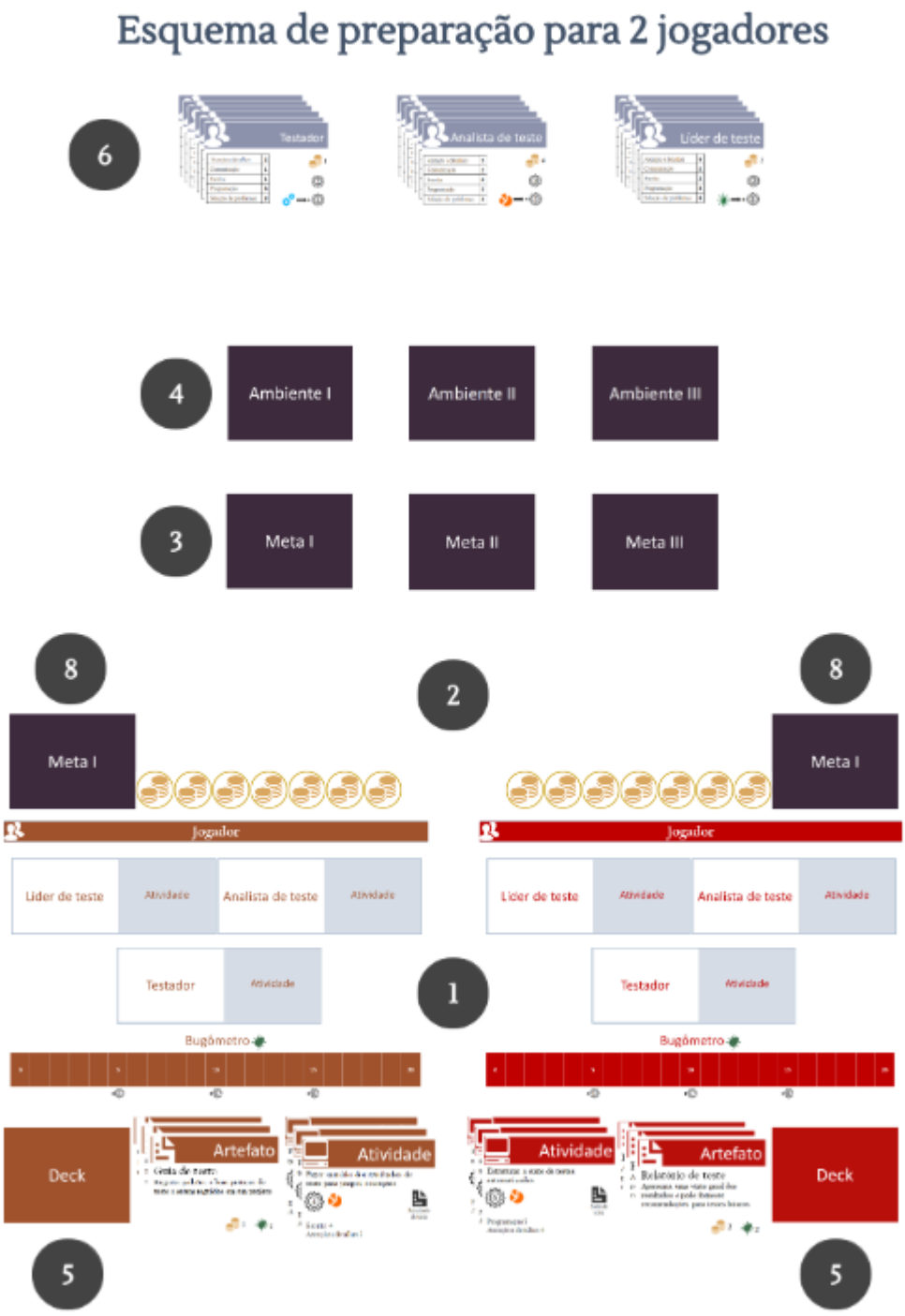
4. **Embaralhar cartas de ambiente**
Os jogadores devem embaralhar as cartas de ambiente pelo número, por exemplo: cartas Ambiente I serão embaralhadas em conjunto e as cartas Ambiente II serão embaralhadas em outro conjunto.

5. **Embaralhar cartas de *deck***
Cada jogador deve embaralhar seu *deck* de cartas.

6. **Organizar funcionários por papel**
A fim de melhor organizar a fase de contratações, os jogadores devem separar as cartas de funcionário pelo seu papel.

7. **Definir ordem de jogo**
Os jogadores devem definir a ordem de jogo em que cada jogador irá jogar.

8. **Puxar meta inicial**
Cada jogador deve puxar uma carta Meta I, que será seu objetivo para o início do jogo e guiará suas contratações de funcionários na próxima fase.


Esquema de preparação para 2 jogadores
6
Testador
Analista de teste
Líder de teste
4
Ambiente I
Ambiente II
Ambiente III
3
Meta I
Meta II
Meta III
8
2
8
Meta I
Meta I
Jogador
Jogador
Líder de teste
Atividade
Analista de teste
Atividade
Líder de teste
Atividade
Analista de teste
Atividade
Testador
Atividade
1
Testador
Atividade
Bugômetro
Bugômetro
Deck
Artefato
Atividade
Atividade
Artefato
Deck
5
5

Contratações

1. **Contratar funcionário**

Cada jogador, na ordem definida na fase anterior, deve selecionar um funcionário por vez até que todos os espaços de funcionário do cartão de jogador estejam preenchidos com uma carta de funcionário.

2. **Devolver moedas das contratações**

Os jogadores devolvem para o banco a quantidade de moedas usadas para fazer as contratações dos funcionários.

3. **Obter esforço por rodada da equipe**

Os jogadores devem obter do banco a quantidade de esforço por rodada que a equipe formada por eles tem.

Exemplo Contratações

Suponha que um jogador tenha realizadas as seguintes contratações:

Após realizar as contratações, os jogadores devem devolver para o banco a quantidade de recursos usados para contratar os funcionários. No exemplo acima cada funcionário custa 2 moedas cada, que é indicado pelo ícone . Com isso o jogador deve devolver 6 moedas para o banco.

Devolvendo para o banco os recursos usados para contratar os funcionários, os jogadores devem pegar do banco a quantidade de esforços por rodada da equipe formada. No exemplo acima, cada funcionário adiciona 2 de esforço por rodada, que é indicado pelo ícone . Com isso o jogador deve pegar 6 recursos de esforço do banco.

Execução das rodadas

1. **Puxar carta do *deck***

Cada jogador começa com 5 cartas do *deck* na mão. A cada início de rodada, cada jogador deve puxar uma nova carta do *deck* para a sua mão. As cartas do *deck* podem ser de dois tipos: sabotagens, que podem ser usadas para prejudicar outro jogador; soluções, que podem ser usadas para solucionar problemas que podem surgir ao virar cartas de ambiente.

Cartas de sabotagem apresentam um título na parte superior e o efeito da carta é apresentado na parte inferior. Elas devem ser enviadas de viradas para baixo durante a rodada para outro jogador. No final da rodada o efeito das cartas é aplicado.

Cartas de solução apresentam um título na parte superior, seu tipo no canto esquerdo, que está relacionado a tipos de problemas e os pontos de bugômetro que ela fornece ao ser usada.

2. **Atualizar meta**

A cada duas rodadas, os jogadores devem atualizar suas metas. Os jogadores começam com a Meta I na primeira rodada e devem concluí-la em 2 rodadas. Na terceira rodada os jogadores devem descartar sua Meta I e puxar uma carta Meta II, que será a meta para o meio do jogo e deve ser concluída em duas rodadas. Na quinta rodada os jogadores descartam a Meta II e puxam a Meta III, que é a última meta a ser cumprida.

3. **Virar carta de ambiente**

A cada rodada os jogadores devem virar uma carta de ambiente. A carta de

ambiente varia, acompanhando a carta de meta:

Rodada	Carta a ser virada
Rodadas 1 e 2	Ambiente I
Rodadas 3 e 4	Ambiente II
Rodadas 5 e 6	Ambiente III

4. Aplicar efeito do ambiente

Após virar a carta de ambiente da rodada, o efeito do ambiente é aplicado. As cartas de ambiente são divididas em cartas de efeito geral, que podem ser benéficas ou maléficas para os jogadores, ou podem ser cartas de problema, que devem ser solucionadas pelos jogadores com uma carta de solução adequada a ser jogada pelo jogador.

As cartas de problema apresentam o problema na parte superior, o tipo de problema no canto inferior direito e o efeito negativo na rodada que elas trazem para os funcionários do jogador que não as solucionar. No exemplo abaixo, a carta de solução apropriada para o problema apresentado é a carta à direita.

As cartas de efeito geral trazem o título na parte superior e o efeito da carta na parte inferior.

5. Atribuição de atividades

Os jogadores devem atribuir as atividades que serão realizadas na rodada para os funcionários e gastar o esforço necessário para cumprir as atividades. Para isso, devem ter em vista o esforço que a equipe possui, os bônus de esforço dos seus funcionários, os níveis de habilidade requeridos por cada atividade para serem cumpridos e os efeitos das cartas de ambiente.

As cartas de atividade possuem um título da atividade na parte superior, o esforço necessário para realizá-la e o tipo da atividade à esquerda da carta, o artefato que ela produz ao ser realizada

e as habilidades necessárias que um funcionário precisa ter para realizá-la.

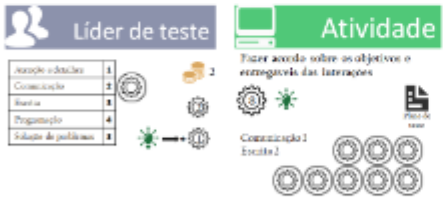
Cada carta de funcionário possui seu papel, que fica na parte superior, suas habilidades na parte esquerda e o custo para contratá-lo, o esforço por rodada que ele fornece e o bônus por tipo de atividade que ele tem na parte direita da carta.

No exemplo abaixo é apresentado um exemplo de atribuição de uma atividade a um funcionário. Suponha para esse exemplo que a equipe de teste tem 8 esforços por rodada disponíveis.

O primeiro passo para atribuir uma atividade é observar as habilidades que são requisitadas pela atividade e as habilidades que o funcionário que vai realizá-la possui. No exemplo, podemos ver que o funcionário possui 2 de habilidade de comunicação e a atividade requer que ele tenha 3 comunicação. Nesse caso, mesmo o funcionário ainda não tendo as habilidades suficiente para realizar a atividades, é possível gastar um recurso de esforço a mais com esse funcionário para aumentar nessa rodada a habilidade dele em comunicação em mais um ponto.

O segundo passo na atribuição da atividade é observar os bônus que um funcionário tem ao realizar uma atividade que ele tem capacidade de realizar. No exemplo, o funcionário é do tipo Líder de teste e tem mais habilidades em atividades de planejamento. Como o tipo da atividade a ser realizada também é planejamento o funcionário ganha mais um recurso de esforço para essa rodada.

Por fim, o jogador tem que observar a quantidade de esforço necessário para cumprir a atividade e a quantidade de esforço disponível para ele na rodada. Como a quantidade requer 8 de esforço e o jogador possui, após o recebimento do bônus pela atribuição correta da atividade, 8 de esforço disponível na rodada, ele pode atribuir os esforços necessários para completar a atividade.



6. Envio de sabotagem

Em cada rodada os jogadores têm a escolha de enviar sabotagens para outros jogadores. Cada jogador pode enviar quantas sabotagens quiser para outros jogadores por rodada. Para isso eles devem ter uma sabotagem do *deck* em sua mão e jogar a sabotagem para outro jogador com a face voltada para baixo.

7. Aplicação das sabotagens

Quando todos tiverem enviado sabotagens, as cartas são viradas e o efeito de cada sabotagem é aplicado no jogador que a recebeu. O efeito de cada sabotagem é informado na parte inferior da carta.

Cartas anti-sabotagem são aplicadas a apenas uma carta de sabotagem no jogador para qual foi jogada. Cartas anti-sabotagem apenas revertem o efeito de uma sabotagem jogada por outro jogador. Ou seja, ela não pode ser usada se o jogador que a possui não tiver recebido na rodada uma carta de sabotagem de outro jogador.

8. Término da rodada

A rodada é terminada ao finalizar as atividades que foram planejadas para a

rodada, obtendo os artefatos gerados pelas atividades e atualizando o marcador de bugômetro e os recursos de moedas e esforços.

As cartas de artefato possuem o artefato que é representado e sua descrição no centro. Na parte inferior da carta são apresentados os bônus fornecidos por produzir os artefatos, que são em forma de moedas e pontos no bugômetro.

No fim de cada rodada, cada jogador também pode fazer contratações de novos funcionários que ainda não foram contratados. Contratações de funcionários entre jogadores também é permitida se ambos os jogadores concordarem. Ao realizar uma contratação de um funcionário de outro jogador, a quantidade de recursos moeda para contratar o funcionário deverá ser dada ao jogador que possui o funcionário.

Condição de vitória

O marcador de bugômetro do cartão do jogador é o marcador de vitória. Ele indica o quão bem o processo de teste foi realizado pelo jogador e reflete a capacidade que o processo executado pelo jogador tem de encontrar bugs no produto testado.

Vence o jogo o jogador que até ao final da sexta rodada tiver obtido mais de 20 pontos no marcador de bugômetro. Caso nenhum jogador vença o jogo em 6 rodadas, o jogador com mais pontos no marcador de bugômetro ganha. Em caso de empate, os seguintes critérios de desempate serão usados na seguinte sequência: quantidade de atividades realizadas, quantidade de artefatos produzidos e quantidade de moedas.

APÊNDICE D – Questionário de pré-teste

Sessão 1 – Termo esclarecido de aceitação

Pré-teste

Apresentação

Prezado (a),

Solicitamos sua participação para a condução de um experimento como parte de um trabalho de dissertação de mestrado do Programa de Pós-Graduação em Informática (PPGIa) da Pontifícia Universidade Católica do Paraná (PUCPR). Esta pesquisa está sendo desenvolvida pelo estudante de mestrado Gabriel Gonçalves Moreira com orientação da professora Dra. Sheila Reinehr, da professora Dra. Andreia Malucelli e do professor Dr. Frederick van Amstel.

Esclarecimentos

Você foi convidado a participar deste estudo porque é estudante em curso superior na área de Computação que ainda não participou de disciplinas que abordem o tópico teste de software.

Os dados aqui fornecidos serão utilizados apenas e exclusivamente para os fins desta pesquisa. Os pesquisadores se comprometem a não repassar informações que identifiquem os participantes da presente pesquisa.

E-mail para contato: gabriel.moreira3@pucpr.edu.br (Gabriel Gonçalves Moreira).

Você aceita participar desse estudo?
( ) Sim
( ) Não

Sessão 2 – Informações demográficas

Informações Demográficas	
Codinome	
Faixa etária:	☐ Menos de 18 anos ☐ 18 a 28 anos ☐ 29 a 39 anos ☐ 40 a 50 anos ☐ Mais de 50 anos
Sexo:	☐ Masculino

	☐ Feminino
Com que frequência você costuma jogar jogos de tabuleiro (RPG, jogos de estratégia)?	☐ Nunca: nunca jogo. ☐ Raramente: jogo de tempos em tempos. ☐ Mensalmente: jogo pelo menos uma vez por mês. ☐ Semanalmente: jogo pelo menos uma vez por semana. ☐ Diariamente: jogo todos os dias.
Com que frequência você costuma jogar jogos de carta?	☐ Nunca: nunca jogo. ☐ Raramente: jogo de tempos em tempos. ☐ Mensalmente: jogo pelo menos uma vez por mês. ☐ Semanalmente: jogo pelo menos uma vez por semana. ☐ Diariamente: jogo todos os dias.
Com que frequência você utilizou testes de software no trabalho ou em projetos da universidade?	☐ Nunca: nunca utilizei. ☐ Raramente: já utilizei algumas poucas vezes. ☐ Ocasionalmente: utilizei algumas vezes. ☐ Frequentemente: utilizei várias vezes. ☐ Sempre: sempre utilizei testes.

Sessão 3 – pré-teste

1. Há várias formas de se organizar o processo de teste de software. Entretanto todas as formas devem seguir uma ordem lógica para que seja executado corretamente. Com base nisso, selecione a opção que melhor apresenta a ordem de atividades do processo de teste.

a) **Planejamento dos testes, Desenvolvimento dos casos de teste, Preparação do ambiente, Execução dos testes e Avaliação dos critérios de qualidade.**
b) Desenvolvimento dos casos de teste, Execução dos testes, Avaliação dos critérios de qualidade e Planejamento dos testes.
c) Planejamento dos testes, Execução dos testes, Desenvolvimento dos casos de teste e Avaliação dos critérios de qualidade.
d) Planejamento dos testes, Desenvolvimento dos casos de teste, Avaliação dos critérios de qualidade e Execução dos testes.
e) Desenvolvimento dos casos de teste, Planejamento dos testes, Execução dos testes e Avaliação dos critérios de qualidade.

2. João é um funcionário eficiente, detalhista e de certa forma antissocial. João tem muitas habilidades com programação, é muito atencioso e sabe solucionar problemas em seus programas e de seus colegas, o que o ajuda muito em seu trabalho. Dentro da empresa onde trabalha, João é responsável pela implementação de casos de teste e com a execução dos testes para testar os produtos de software da empresa.
   Tendo em vista a descrição acima da persona de João, é possível afirmar que as características apresentadas mais se relacionam a qual papel do processo de teste de software?
   a) **Testador**
   b) Analista de teste
   c) Líder de teste
   d) Inspecionador
   e) Auditor

3. Luiza é uma funcionária extremamente dedicada e comunicativa. Ela gosta de saber de tudo o que acontece na empresa. Luiza tem grandes habilidades de planejamento, negociação e gestão, especialmente gestão de pessoas. Por conta disso, Luiza assumiu um papel importante na empresa onde trabalha. Ela é responsável por fazer acordos sobre os objetivos e entregáveis do processo do qual é responsável. Além disso, como forma de promover melhoria contínua desse processo, ela sempre busca avaliar a efetividade e produtividade e manter dados sobre a execução dos processos que coordena.
   Tendo em vista a descrição acima da persona de Luiza, é possível afirmar que as características apresentadas mais se relacionam a qual papel do processo de teste de software?
   a) **Líder de teste**
   b) Testador
   c) Inspetor
   d) Analista de teste
   e) Auditor

4. Há vários papéis responsáveis por executar o processo de teste. Qual são as principais atribuições do analista de teste?
   a) **Estruturar a suíte de testes automatizados**
   b) **Escrever guia para execução da suíte de testes**
   c) Analisar falhas que ocorreram durante a execução dos testes
   d) Executar o conjunto de testes para verificar o produto
   e) Corrigir bugs encontrados
   f) Fazer acordo sobre os objetivos e entregáveis das interações

5. Para que os testes possam ser devidamente executados, é necessário se ter produzido o artefato ____1____, que é uma coleção de casos de teste utilizados para testar um

produto. Esse artefato é produzido pelo Analista de teste, ou designer de teste, que, para auxiliar na aplicação desses casos de teste cria também o artefato ____2____, que ajuda os testadores na execução dos testes.
Em relação ao processo de teste, marque a opção que apresenta as respectivas palavras que corretamente completam o texto.

a) **1 – Suíte de teste, 2 – Guia de teste.**
b) 1 – Log de teste, 2 – Guia de teste.
c) 1 – Plano de teste, 2 – Log de teste.
d) 1 – Suíte de teste, 2 – Resultado dos testes.
e) 1 – Log de teste, 2 – Plano de teste.

6. O artefato Script de teste é responsável especificamente por:
   a) **Detalhar o passo-a-passo e os dados de entrada necessários para executar um teste**
   b) Apresentar o resultado obtido após a execução da suíte de teste
   c) Apresentar uma visão geral dos resultados e pode fornecer recomendações para testes futuros
   d) Definir os métodos e procedimentos utilizados e como será a avaliação dos resultados
   e) Registrar padrões e boas práticas de teste a serem seguidos em um projeto

7. O artefato ____1____ é produto do processo de teste e deve ser produzido logo após a execução dos testes. Este artefato é produzido pelo ____2____ tem o objetivo principal de ____3____.
   Em relação ao processo de teste, marque a opção que apresenta as respectivas palavras que corretamente completam o texto.
   a) **1 – Resultado dos testes, 2 – Analista de teste, 3 - apresentar uma análise dos resultados.**
   b) 1 – Relatório de teste, 2 – Testador, 3 – melhorar os casos de teste.
   c) 1 – Plano de teste, 2 – Líder de teste, 3 – apresentar o plano final que foi executado.
   d) 1 – Script de teste, 2 – Analista de teste, 3 – os casos de teste utilizados.
   e) 1 – Resultado dos testes, 2 – Testador, 3 – preparar os testes para as futuras manutenções.

8. A execução do conjunto de testes para verificar o produto no processo de teste é realizada principalmente pelo ____1____, que também será responsável por analisar as falhas encontradas por esses testes. Como resultado da execução dos testes, é gerado um artefato ____2____, que poderá ser utilizado futuramente para as recomendações de correções.
   Em relação ao processo de teste, marque a opção que apresenta as respectivas palavras que corretamente completam o texto.
   a) **1 – Testador, 2 – Log de teste.**
   b) 1 – Testador, 2 – Script de teste.

c) 1 – Analista de teste, 2 – Log de teste.
d) 1 – Analista de teste, 2 – Suíte de teste.
e) 1 – Analista de teste, 2 – Script de teste.

9. O acordo com o cliente sobre os objetivos dos testes e os entregáveis de cada interação é realizado pelo ____1____, e por isso esse papel exige muito ____2____. Como resultado deste acordo, é gerado o ____3____, que definirá os métodos e procedimentos a serem utilizados bem como será utilizado futuramente para a verificação do atingimento do nível acordado de qualidade do software testado.
Em relação ao processo de teste, marque a opção que apresenta as respectivas palavras que corretamente completam o texto.
a) **1 – Líder de teste, 2 – Habilidades pessoais, 3 – Plano de teste**
b) 1 – Líder de teste, 2 – Habilidades técnicas, 2 – Guia de teste
c) 1 – Analista de teste, 2 – Habilidades gerenciais, 2 – Plano de teste
d) 1 – Analista de teste, 2 – Habilidades analíticas, 2 – Guia de teste
e) 1- Testador, 2 – Habilidades pessoais, 2 – Guia de teste

10. Analise a seguinte situação:

Gilberto é o líder da equipe de testes de uma empresa. A empresa em que Gilberto trabalha mantém diversos sistemas e atualmente está avançando com a implementação de um novo sistema de gerenciamento de estoques. Gilberto, ao voltar de suas férias, nota que há uma grande confusão sobre as atividades que estão sendo realizadas. Ninguém da equipe de teste sabe o que está sendo feito, o que vem impedindo as atividades de teste de serem realizadas.

Analisando a situação, quais ações Gilberto poderia fazer para corrigir tal problema?

a) **Fazer reuniões diárias rápidas para que todos falem o que estão fazendo**
b) **Usar ferramenta de gerenciamento de tarefas para organizar as atividades do time**
c) Fazer com que todos participem de um treinamento novas tecnologias e tendências do mercado
d) Procurar uma ferramenta de automação de testes
e) Reportar resultados das atividades frequentemente com o cliente

APÊNDICE E – Questionário de pós-teste

Sessão 1- Pós-teste

Informações Demográficas	
Informe seu codinome	

1. Há várias formas de se organizar o processo de teste de software. Entretanto todas as formas devem seguir uma ordem lógica para que seja executado corretamente. Com base nisso, selecione a opção que melhor apresenta a ordem de atividades do processo de teste.
   a) **Planejamento dos testes, Desenvolvimento dos casos de teste, Preparação do ambiente, Execução dos testes e Avaliação dos critérios de qualidade.**
   b) Desenvolvimento dos casos de teste, Execução dos testes, Avaliação dos critérios de qualidade e Planejamento dos testes.
   c) Planejamento dos testes, Execução dos testes, Desenvolvimento dos casos de teste e Avaliação dos critérios de qualidade.
   d) Planejamento dos testes, Desenvolvimento dos casos de teste, Avaliação dos critérios de qualidade e Execução dos testes.
   e) Desenvolvimento dos casos de teste, Planejamento dos testes, Execução dos testes e Avaliação dos critérios de qualidade.

2. João é um funcionário eficiente, detalhista e de certa forma antissocial. João tem muitas habilidades com programação, é muito atencioso e sabe solucionar problemas em seus programas e de seus colegas, o que o ajuda muito em seu trabalho. Dentro da empresa onde trabalha, João é responsável pela implementação de casos de teste e com a execução dos testes para testar os produtos de software da empresa.
   Tendo em vista a descrição acima da persona de João, é possível afirmar que as características apresentadas mais se relacionam a qual papel do processo de teste de software?
   a) **Testador**
   b) Analista de teste
   c) Líder de teste
   d) Inspecionador
   e) Auditor

3. Luiza é uma funcionária extremamente dedicada e comunicativa. Ela gosta de saber de tudo o que acontece na empresa. Luiza tem grandes habilidades de planejamento, negociação e gestão, especialmente gestão de pessoas. Por conta disso, Luiza assumiu um papel importante na empresa onde trabalha. Ela é responsável por fazer acordos sobre os objetivos e entregáveis do processo do qual é responsável. Além disso, como forma de promover melhoria contínua desse processo, ela sempre busca avaliar a

efetividade e produtividade e manter dados sobre a execução dos processos que coordena.
Tendo em vista a descrição acima da persona de Luiza, é possível afirmar que as características apresentadas mais se relacionam a qual papel do processo de teste de software?

a) **Líder de teste**
b) Testador
c) Inspetor
d) Analista de teste
e) Auditor

4. Há vários papéis responsáveis por executar o processo de teste. Qual são as principais atribuições do analista de teste?
   a) **Estruturar a suíte de testes automatizados**
   b) **Escrever guia para execução da suíte de testes**
   c) Analisar falhas que ocorreram durante a execução dos testes
   d) Executar o conjunto de testes para verificar o produto
   e) Corrigir bugs encontrados
   f) Fazer acordo sobre os objetivos e entregáveis das interações

5. Para que os testes possam ser devidamente executados, é necessário se ter produzido o artefato ____1____, que é uma coleção de casos de teste utilizados para testar um produto. Esse artefato é produzido pelo Analista de teste, ou designer de teste, que, para auxiliar na aplicação desses casos de teste cria também o artefato ____2____, que ajuda os testadores na execução dos testes.
   Em relação ao processo de teste, marque a opção que apresenta as respectivas palavras que corretamente completam o texto.

   a) **1 – Suíte de teste, 2 – Guia de teste.**
   b) 1 – Log de teste, 2 – Guia de teste.
   c) 1 – Plano de teste, 2 – Log de teste.
   d) 1 – Suíte de teste, 2 – Resultado dos testes.
   e) 1 – Log de teste, 2 – Plano de teste.

6. O artefato Script de teste é responsável especificamente por:

a) **Detalhar o passo-a-passo e os dados de entrada necessários para executar um teste**
b) Apresentar o resultado obtido após a execução da suíte de teste
c) Apresentar uma visão geral dos resultados e pode fornecer recomendações para testes futuros
d) Definir os métodos e procedimentos utilizados e como será a avaliação dos resultados
e) Registrar padrões e boas práticas de teste a serem seguidos em um projeto

7. O artefato ____1____ é produto do processo de teste e deve ser produzido logo após a execução dos testes. Este artefato é produzido pelo ____2____ tem o objetivo principal de ____3____.
   Em relação ao processo de teste, marque a opção que apresenta as respectivas palavras que corretamente completam o texto.
   a) **1 – Resultado dos testes, 2 – Analista de teste, 3 - apresentar uma análise dos resultados.**
   b) 1 – Relatório de teste, 2 – Testador, 3 – melhorar os casos de teste.
   c) 1 – Plano de teste, 2 – Líder de teste, 3 – apresentar o plano final que foi executado.
   d) 1 – Script de teste, 2 – Analista de teste, 3 – os casos de teste utilizados.
   e) 1 – Resultado dos testes, 2 – Testador, 3 – preparar os testes para as futuras manutenções.

8. A execução do conjunto de testes para verificar o produto no processo de teste é realizada principalmente pelo ____1____, que também será responsável por analisar as falhas encontradas por esses testes. Como resultado da execução dos testes, é gerado um artefato ____2____, que poderá ser utilizado futuramente para as recomendações de correções.
   Em relação ao processo de teste, marque a opção que apresenta as respectivas palavras que corretamente completam o texto.
   a) **1 – Testador, 2 – Log de teste.**
   b) 1 – Testador, 2 – Script de teste.
   c) 1 – Analista de teste, 2 – Log de teste.
   d) 1 – Analista de teste, 2 – Suíte de teste.
   e) 1 – Analista de teste, 2 – Script de teste.

9. O acordo com o cliente sobre os objetivos dos testes e os entregáveis de cada interação é realizado pelo ____1____, e por isso esse papel exige muito ____2____. Como resultado deste acordo, é gerado o ____3____, que definirá os métodos e procedimentos a serem utilizados bem como será utilizado futuramente para a verificação do atingimento do nível acordado de qualidade do software testado.
   Em relação ao processo de teste, marque a opção que apresenta as respectivas palavras que corretamente completam o texto.
   a) **1 – Líder de teste, 2 – Habilidades pessoais, 3 – Plano de teste**
   b) 1 – Líder de teste, 2 – Habilidades técnicas, 2 – Guia de teste
   c) 1 – Analista de teste, 2 – Habilidades gerenciais, 2 – Plano de teste
   d) 1 – Analista de teste, 2 – Habilidades analíticas, 2 – Guia de teste
   e) 1- Testador, 2 – Habilidades pessoais, 2 – Guia de teste

10. Analise a seguinte situação:

A Sys é uma empresa antiga, com funcionários fiéis e empenhados em suas atividades. Entretanto, em busca de novos mercados, a Sys iniciou um trabalho inovador dentro da

empresa focado no mercado de aplicativos móveis. Só que há um problema: a equipe de testes está tendo dificuldade para testar e entender as novas tecnologias utilizadas no projeto.

Analisando a situação, qual ação a empresa Sys poderia fazer para corrigir tal problema?

a) **Investir em um rápido treinamento da nova tecnologia para facilitar os testes**
b) Convencer a equipe de desenvolvimento a mudar a tecnologia utilizada, focando em algo para a Web
c) Delegar todos os testes para os programadores
d) Utilizar ferramentas automatizadas
e) Fazer apenas análises heurísticas com a interface da aplicação

Sessão 2 - Questionário para a avaliação da qualidade de jogos

Nome do jogo: ProTesters
Gostaríamos que você respondesse as questões abaixo sobre a sua percepção da qualidade do jogo para nos ajudar a melhorá-lo. Todos os dados são coletados anonimamente e somente serão utilizados no contexto desta pesquisa. Algumas fotografias poderão ser feitas como registro desta atividade, mas não serão publicadas em nenhum local sem autorização.
Nome do pesquisador responsável: Pontifícia Universidade Católica do Paraná
Local e data: Curitiba, Paraná – 18/09/2021

Por favor, **marque uma opção** de acordo com o quanto você concorda ou discorda de cada afirmação abaixo.

Usabilidade					
Afirmações	**Marque uma opção conforme sua avaliação**				
	Discordo totalmente	**Discordo**	**Nem discordo, nem concordo**	**Concordo**	**Concordo totalmente**
O design do jogo é atraente (tabuleiro, cartas, interfaces, gráficos, etc.).	☐	☐	☐	☐	☐
Os textos, cores e fontes combinam e são consistentes.	☐	☐	☐	☐	☐
Eu precisei aprender poucas coisas para poder começar a jogar o jogo.	☐	☐	☐	☐	☐
Aprender a jogar este jogo foi fácil para mim.	☐	☐	☐	☐	☐
Eu acho que a maioria das pessoas aprenderiam a jogar este jogo rapidamente.	☐	☐	☐	☐	☐
Eu considero que o jogo é fácil de jogar.	☐	☐	☐	☐	☐
As regras do jogo são claras e compreensíveis.	☐	☐	☐	☐	☐
As fontes (tamanho e estilo) utilizadas no jogo são legíveis.	☐	☐	☐	☐	☐
As cores utilizadas no jogo são compreensíveis.	☐	☐	☐	☐	☐

Experiência do Jogador	
Afirmações	**Marque uma opção conforme sua avaliação**

	Discordo totalmente	Discordo	Nem discordo, nem concordo	Concordo	Concordo totalmente
A organização do conteúdo me ajudou a estar confiante de que eu iria aprender com este jogo.	☐	☐	☐	☐	☐
Este jogo é adequadamente desafiador para mim.	☐	☐	☐	☐	☐
O jogo oferece novos desafios (oferece novos obstáculos, situações ou variações) com um ritmo adequado.	☐	☐	☐	☐	☐
O jogo não se torna monótono nas suas tarefas (repetitivo ou com tarefas chatas).	☐	☐	☐	☐	☐
Completar as tarefas do jogo me deu um sentimento de realização.	☐	☐	☐	☐	☐
É devido ao meu esforço pessoal que eu consigo avançar no jogo.	☐	☐	☐	☐	☐
Me sinto satisfeito com as coisas que aprendi no jogo.	☐	☐	☐	☐	☐
Eu recomendaria este jogo para meus colegas.	☐	☐	☐	☐	☐
Eu pude interagir com outras pessoas durante o jogo.	☐	☐	☐	☐	☐
O jogo promove momentos de cooperação e/ou competição entre os jogadores.	☐	☐	☐	☐	☐
Eu me senti bem interagindo com outras pessoas durante o jogo.	☐	☐	☐	☐	☐
Eu me diverti com o jogo.	☐	☐	☐	☐	☐
Aconteceu alguma situação durante o jogo (elementos do jogo, competição, etc.) que me fez sorrir	☐	☐	☐	☐	☐
Houve algo interessante no início do jogo que capturou minha atenção.	☐	☐	☐	☐	☐
Eu estava tão envolvido no jogo que eu perdi a noção do tempo.	☐	☐	☐	☐	☐
Eu esqueci sobre o ambiente ao meu redor enquanto jogava este jogo.	☐	☐	☐	☐	☐
O conteúdo do jogo é relevante para os meus interesses.	☐	☐	☐	☐	☐
É claro para mim como o conteúdo do jogo está relacionado com a disciplina.	☐	☐	☐	☐	☐
O jogo é um método de ensino adequado para esta disciplina.	☐	☐	☐	☐	☐
Eu prefiro aprender com este jogo do que de outra forma (outro método de ensino).	☐	☐	☐	☐	☐
O jogo contribuiu para a minha aprendizagem na disciplina.	☐	☐	☐	☐	☐
O jogo foi eficiente para minha aprendizagem, em comparação com outras atividades da disciplina.	☐	☐	☐	☐	☐
O jogo contribuiu para que eu entendesse o ordenamento das atividades do processo de teste de software.	☐	☐	☐	☐	☐
O jogo contribuiu para que eu pudesse distinguir quais são os principais artefatos do processo de teste.	☐	☐	☐	☐	☐
O jogo contribuiu para meu entendimento sobre as responsabilidade do processo de teste.	☐	☐	☐	☐	☐

O jogo contribuiu para meu reconhecimento e solução de situações e problemas que podem ocorrer no processo de teste.	☐	☐	☐	☐	☐

Nos conte sobre o que você aprendeu com o jogo ProTesters?_____
_____
_____
_____

Quais tópicos você acha que o jogo te motivou a estudar?
[ ] As atividades de teste
[ ] Os artefatos que são produzidos
[ ] As responsabilidades dos papéis
[ ] As situações e problemas que podem ocorrer no ambiente de teste
[ ] Os objetivos do processo de teste
[ ] Outra opção