UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Gianlucca Assunção Leoncini

Risks Dungeon: jogo educativo para o ensino de gerenciamento de riscos de um projeto de software

São José do Rio Preto
2021

Gianlucca Assunção Leoncini

Risks Dungeon: jogo educativo para o ensino de gerenciamento de riscos de um projeto de software

Trabalho de Conclusão de Curso (TCC) apresentado como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, junto ao Conselho de Curso de Bacharelado em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Orientadora:

Profa. Dra. Carina Alexandra Rondini

São José do Rio Preto
2021

L582r

Leoncini, Gianlucca Assunção

Risks Dungeon: jogo educativo para o ensino de gerenciamento de riscos de um projeto de software / Gianlucca Assunção Leoncini. -- São José do Rio Preto, 2022

70 p.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (Unesp), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto

Orientadora: Carina Alexandra Rondini

1. Jogos educativos. 2. Engenharia de software. 3. Agentes inteligentes (Software). 4. Fuzzy sets. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Gianlucca Assunção Leoncini

Risks Dungeon: jogo educativo para o ensino de gerenciamento de riscos de um projeto de software

Trabalho de Conclusão de Curso (TCC) apresentado como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, junto ao Conselho de Curso de Bacharelado em Ciência da Computação, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Campus de São José do Rio Preto.

Comissão Examinadora:

Profa. Dra. Carina Alexandra Rondini
UNESP – Campus de São José do Rio Preto
Orientadora

Profa. Dra. Rogéria Cristiane Gratão de Souza
UNESP – Campus de São José do Rio Preto

Prof. Dr. Rodrigo Capobianco Guido

UNESP – Campus de São José do Rio Preto

São José do Rio Preto
2021

Agradecimentos

Agradeço principalmente a meus pais Edson e Gracia, que sempre estiveram me apoiando nas minhas decisões e projetos e a meu irmão Gabriel que sempre me fez companhia.

A meus amigos que confiaram e me motivaram nos momentos mais turbulentos.

E a todos os professores que me guiaram, em especial à professora Carina por abraçar meu projeto e minhas ideias e me ajudar a realizá-las.

"Nada contribui tanto para tranquilizar a mente como um propósito sólido, um ponto no qual se possa fixar a alma"

- Mary Shelley

Resumo

Este trabalho apresenta o desenvolvimento de um jogo educativo cujo objetivo geral visa o ensino de gerenciamento de riscos. Especificamente seus objetivos consistem em chamar a atenção para a disciplina em tela, bem como dos alunos para a disciplina, promovendo maior integração entre os aspectos - ludismo, mecânicas de jogo e conteúdo disciplinar. Para tanto, foram pesquisadas as vantagens de se utilizar de meios lúdicos na educação em geral e do uso de jogos educacionais no ensino superior, onde se situa o público alvo, além de metodologias para desenvolver esse tipo de jogo. Também foram pesquisados jogos educativos sobre gerenciamento de riscos, sucessos e falhas obtidos nesses jogos e técnicas usadas. Tomando como base o jogo eRiskGame, um dos trabalhos pesquisados, foram utilizados agentes inteligentes e sistemas *fuzzy* para complementar sistemas do jogo. O planejamento do jogo foi realizado utilizando a metodologia ENgAGED, que fornece diretrizes específicas para o desenvolvimento de jogos educativos, seguindo seus passos até a elaboração do *game design document* (GDD). O jogo foi desenvolvido utilizando a *game engine* Unity e após sua implementação, foi disponibilizado para teste e avaliação, seguindo o modelo MEEGA+, um modelo para avaliação de jogos educativos amplamente usado. O jogo implementou uma combinação de jogabilidade com métodos e modelos de gerenciamento de riscos e de projetos, como a matriz de probabilidade/consequência e os modelos incrementais e SCRUM. Embora não tenha sido possível alcançar um número de avaliações satisfatório para chegar-se a uma conclusão sobre o desempenho do jogo, os resultados preliminares indicam que o conteúdo e interface são promissores e há bastante espaço para melhoria e diversificação de conteúdo disciplinar.

Palavras-chave: jogo educativo, gerenciamento de riscos, desenvolvimento de jogos

Abstract

This work presents the development of an educational game whose general objective is to teach risk management. Specifically, its objectives are to draw attention to the subject on screen, as well as students' attention to the subject, promoting greater integration between aspects - ludism, game mechanics and disciplinary content. In order to do so, the advantages of using recreational means in education in general and the use of educational games in higher education were researched, where the target audience is located, as well as methodologies to develop this type of game. Educational games on risk management, successes and failures obtained in these games, and techniques used were also researched. Based on the eRiskGame game, one of the researched works, intelligent agents and *fuzzy* systems were used to complement game systems. Game planning was carried out using the ENgAGED methodology, which provides specific guidelines for the development of educational games, following its steps until the elaboration of the *game design document* (GDD). The game was developed using the Unity *game engine* and after its implementation, it was made available for testing and evaluation, following the MEEGA+ model, a widely used model for evaluating educational games. The game implemented a combination of gameplay with risk and project management methods and models, such as the probability/consequence matrix and the incremental and SCRUM models. Although it was not possible to reach a satisfactory number of evaluations to reach a conclusion about the performance of the game, the preliminary results indicate that the content and interface are promising and there is a lot of room for improvement and diversification of disciplinary content.

Keywords: educational game, risk management, game development,

Lista de ilustrações

Figura 1 – Publicações relacionadas a *serious games* na educação de 2009 a 2018 . 1

Figura 2 – Representação de conjuntos de altura de uma pessoa comparando lógica tradicional (esquerda) com lógica *fuzzy* (direita) 11
Figura 3 – Fuzzyficação das variáveis x e y resultando em seus respectivos graus de pertinência 12
Figura 4 – Conjuntos *fuzzy* e graus de pertinência da variável de saída *z* e respectivos graus de pertinência, produzidos pelas regras *fuzzy* 1, 2 e 3 13
Figura 5 – Conjunto resultado da agregação das regras *fuzzy* 1, 2 e 3 13
Figura 6 – Taxonomias do Domínio Cognitivo - 1956 x 2001 14
Figura 7 – Processo de Gerenciamento de Riscos 17
Figura 8 – Metodologia *Bow Tie* 18
Figura 9 – Matriz de Riscos (Probabilidade/Consequências) 19

Figura 10 – Relatório Final 31
Figura 11 – Fase de Seleção da Equipe 32
Figura 12 – Fase de Avaliação de Riscos 33
Figura 13 – Fase de Planejamento de Riscos 34
Figura 14 – Mapa do Primeiro Nível da Masmorra do Projeto de Implantação de Sistema ERP 34
Figura 15 – Mapa da Masmorra do Projeto Desenvolvimento de App 35
Figura 16 – Tela de Apresentação de Risco 35
Figura 17 – Tela de Apresentação de Oportunidade 36
Figura 18 – Curva do Conjunto Fuzzy de Moral Alta 46
Figura 19 – Curva do Conjunto Fuzzy de Moral Normal 46
Figura 20 – Curva do Conjunto Fuzzy de Moral Baixa 47
Figura 21 – Curva do Conjunto Fuzzy de Probabilidade Muito Alta 49
Figura 22 – Curva do Conjunto Fuzzy de Probabilidade Alta 50
Figura 23 – Curva do Conjunto Fuzzy de Probabilidade Média 50
Figura 24 – Curva do Conjunto Fuzzy de Probabilidade Baixa 51
Figura 25 – Curva do Conjunto Fuzzy de Probabilidade Muito Baixa 51

Lista de quadros

Quadro 1 – Resumo das fases do método ENgAGED 6
Quadro 2 – Variáveis e valores linguísticos de análise de riscos em um projeto . . 12
Quadro 3 – Definição das Dimensões/Subdimensões do MEEGA+ 16
Quadro 4 – Exemplo de Aplicação da Análise SWOT 20
Quadro 5 – Campos de um Risco 39
Quadro 6 – Campos de uma Oportunidade 41
Quadro 7 – Campos do Gerenciador do Jogador 43
Quadro 8 – Campos do *Game Manager* 44

Lista de Abreviaturas

AI Agentes Inteligentes

GDD *Game Design Document*

UI Unidade Instrucional

ERP Enterprise Resource Planning

EVM *Earned Value Management*

GUI Game User Interface

MEEGA *Model for the Evaluation of Educational GAmes*

PMP *Project Management Professional*

SWIFT *Structured What-If Technique*

SWOT *Strenghts, Weaknesses, Opportunities, Threats*

Sumário

Lista de ilustrações . . . . . **ix**

Lista de quadros . . . . . **ix**

Sumário . . . . . **xii**

1 INTRODUÇÃO **1**
1.1 Justificativa **1**
1.2 Objetivos **3**
1.3 Organização da Monografia **3**

2 FUNDAMENTAÇÃO TEÓRICA **4**
2.1 Educação Lúdica e Jogos Educativos **4**
2.2 Desenvolvimento de Jogos Educacionais **5**
2.2.1 Metodologias de Desenvolvimento de Jogos Educacionais 5
2.2.2 Agentes Inteligentes 8
2.2.3 Lógica *Fuzzy* 10
2.2.4 Avaliação de Jogos Educacionais 14
2.3 Gerenciamento de Riscos **16**
2.3.1 Metodologias de Identificação de Riscos 17
2.3.2 Análise de Riscos 20
2.3.3 Planejamento de Riscos 21
2.3.4 Monitoramento de Riscos 22
2.4 Trabalhos Correlatos **23**
2.5 Análise do Estado da Arte **25**

3 EXECUÇÃO DO TRABALHO **27**
3.1 Análise da Unidade Instrucional (UI) **27**
3.2 Projeto da Unidade Instrucional **27**
3.2.1 Definir avaliação do aluno e conteúdo da estratégia instrucional 27
3.2.2 Revisar o modelo de avaliação do jogo 28
3.3 Desenvolvimento do Jogo Educacional **28**
3.3.1 Levantar requisitos do jogo 28
3.4 Concepção do Jogo **29**
3.4.1 História e Gameplay 29
3.4.2 Personagens e Habilidades 31
3.4.3 Riscos e Oportunidades 32

3.4.4 Interface . . . 33
3.5 Design do Jogo . . . 36
3.5.1 Definir linguagem de programação ou *game engine* . . . 36
3.5.2 Produzir ilustrações ou imagens dos elementos do jogo . . . 36
3.5.3 Modelar o Jogo . . . 36
3.6 Implementação do Jogo . . . 37
3.6.1 Os Riscos . . . 38
3.6.2 As Oportunidades . . . 40
3.6.3 As Prevenções . . . 42
3.6.4 As Habilidades . . . 42
3.6.5 O Jogador . . . 42
3.6.6 O Game Manager . . . 43
3.6.7 As Salas . . . 44
3.6.8 Os Agentes . . . 45
3.6.8.1 Team Agent . . . 45
3.6.8.2 Rooms Agent . . . 47
3.6.8.3 Dungeon Agent . . . 48
3.6.8.4 Risks Agent . . . 48

4 EXECUÇÃO E AVALIAÇÃO DA UNIDADE INSTRUCIONAL . . . 52
4.1 Resultados . . . 52
4.2 Trabalhos Futuros . . . 52

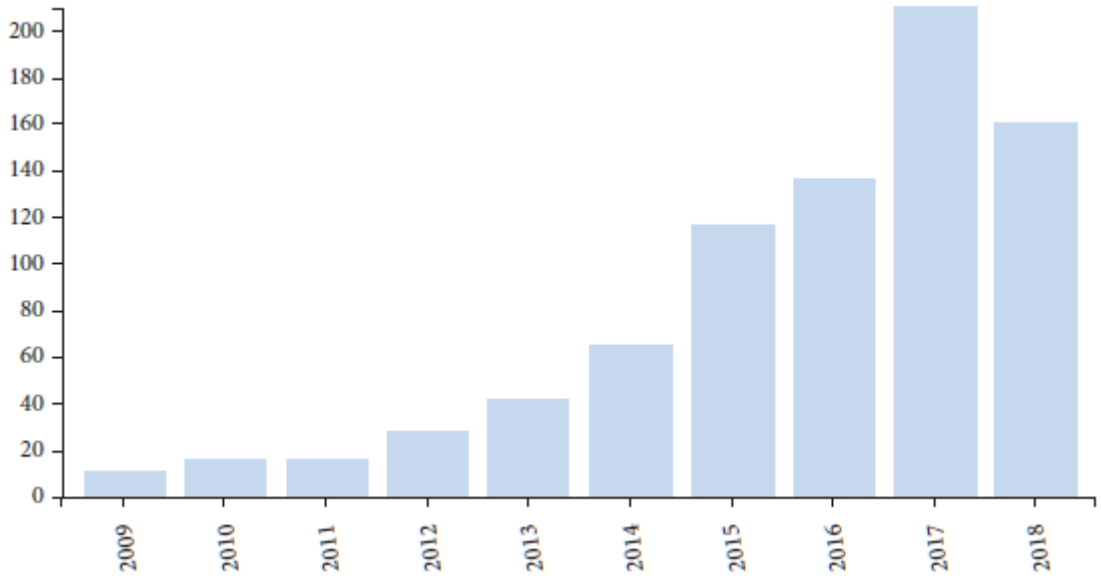
5 CONCLUSÃO . . . 54

REFERÊNCIAS . . . 55

1 Introdução

Jogos vêm sendo utilizados cada vez mais no processo de ensino e aprendizagem, como pode ser observado na Figura 1, em especial na educação para crianças e adolescentes (ZHONGGEN, 2019). Porém há também o interesse e um bom uso deles no ensino superior, foco do presente trabalho, como forma de auxiliar a formação intelectual e principalmente para reter interesse e atenção dos discentes (HOPPE; KROEFF, 2014), visto a recorrente desconexão entre docente e discente, em que este se desmotiva com demasiado conteúdo teórico e aquele se queixa da desmotivação deste (D'ÁVILA, 2014; SINDRE; NATVIG; JAHRE, 2008).

Figura 1 – Publicações relacionadas a *serious games* na educação de 2009 a 2018



Fonte: Baseado em (ZHONGGEN, 2019, p. 2)

1.1 Justificativa

Os jogos educacionais são uma subcategoria de *serious games*, termo definido como: jogos cujo propósito ou uso primário seja outro ao de entretenimento (BACKLUND; HENDRIX, 2013; ZHONGGEN, 2019). No entanto, eles não necessariamente precisam perder seu caráter de entretenimento, ou lúdico, sendo este um componente essencial para o aprendizado.

O lúdico, ou seja, atividade relativa a jogos (DICTIONARIES, 2013), é uma forma de ação inerente ao ser humano, independentemente de sua idade (HOPPE; KROEFF, 2014), e está atrelado a cada setor cultural e social como um fator de construção de cultura, inclusive na busca e transmissão de conhecimento (HUIZINGA, 2020). Assim, o ludismo pode ser usado como fator de auxílio ao aprendizado utilizando de jogos educativos, dado que as atividades lúdicas

permitem a ponte entre cognitivo, emoção e ação, proporcionando uma experiência plena e integrada (LUCKESI, 2017). Estes elementos são valiosos para a formação intelectual, unindo a arte com a educação, trazendo um estado de ânimo e prontidão para aprender, resultando em uma experiência mais satisfatória e saudável para o usuário (D'ÁVILA, 2014; LUCKESI, 2017).

Reter interesse dos alunos é importante, pois o processo de atenção é imprescindível para absorção e interpretação de informações (LADEWIG, 2000). Além disso, o discente aprende quando tem uma mudança de comportamento e a sensação de construção e de descobrimento sobre a forma de enxergar o conteúdo aprendido (GHELLI, 2004). Assim, o aprendizado pode ser facilitado por jogos educativos, já que eles propõem cenários diferentes, sendo uma ótima ferramenta para a construção de novos comportamentos (HOPPE; KROEFF, 2014).

Em complemento ao seu caráter lúdico, o ambiente de jogo é um ambiente controlado, com um cenário sem consequências no mundo real, o que permite ao aluno a possibilidade de aprender com eventuais erros cometidos e melhorar conforme joga, tirando valor da tentativa e erro, experimentando novos caminhos a cada jogada (CAULFIELD et al., 2011).

Uma área em que se vê crescente uso de jogos para colocar em prática o ambiente controlado é a de programação e engenharia de *software*, em que o conteúdo na sala de aula é massivamente textual e necessita de prática, porém não existe muito espaço e tempo para praticar e um projeto real pode ser muito hostil para os discentes (OLIVEIRA; CINTRA; NETO, 2013).

Os jogos educacionais nessa área são cada vez mais presentes, com destaque para jogos digitais (XIE; TILLMANN; HALLEUX, 2013; KOSA et al., 2016; SANTOS et al., 2020; SOUZA et al., 2017). Analisando diversos *surveys* de trabalhos na área de *serious games* e de jogos educacionais, com enfoque para a área de engenharia de *software* e abrangendo o cenário brasileiro, como os de Zhonggen (2019), de Backlund e Hendrix (2013), de Craig et al (2011), Caulfield et al. (2011), Kosa et al. (2016) e Santos et al. (2020), é possível observar que os jogos mostraram resultados interessantes quanto ao desempenho cognitivo, colaborativo e de retenção de atenção e interesse.

Adicionalmente, existe uma preferência para o desenvolvimento de jogos abordando gerenciamento de projetos de *software* e suas metodologias, deixando um espaço para temas como engenharia de requisitos, arquitetura e gerenciamento de riscos (SANTOS et al., 2020; OLIVEIRA; CINTRA; NETO, 2013). Caulfield et al. (2011) asseveram ainda uma desconexão entre a jogabilidade do jogo com os conceitos de engenharia de *software* abordados, tornando alguns jogos muito técnicos, sobrando quase apenas a programação, perdendo parte de sua característica lúdica e do componente visual, necessário para um melhor *feedback* entre a ação do jogador e seu impacto no jogo.

Por fim, o trabalho de Oliveira, Cintra e Neto (2013) pontuou que, com pouco espaço e tempo para prática no ensino tradicional universitário, que simplifica os problemas do planejamento de projetos e gerenciamento de riscos, muitos projetos falham por conta da falta de experiência com riscos que possam ocorrer durante um projeto. Assim, um jogo que possa cobrir esses quesitos seria bem-vindo.

1.2 Objetivos

Dessa forma, este trabalho, se propõe a desenvolver um jogo educacional de um jogador (*single player*) abordando a área de gerenciamento de riscos, usando seus conteúdos disciplinares, colocando-os em prática, unidos a uma jogabilidade que recorra ao elemento lúdico e de entretenimento, para estudantes do ensino superior da área de computação e engenharia de *software*. Por meio do jogo desenvolvido deseja-se atrair e reter a atenção de estudantes de engenharia de *software* para a área de gerenciamento de riscos, além de despertar maior interesse pela área, ajudando a enxergar além da transmissão tradicional do conteúdo.

1.3 Organização da Monografia

Esta seção foram apresentados os objetivos deste trabalho, desenvolver um jogo educativo, e o porquê de tal escolha, introduzindo como e porquê jogos educativos estão se tornando mais comuns.

As seções a seguir estão organizadas desta forma:

- Seção 2 - Fundamentação Teórica: apresenta e aprofunda os conceitos essenciais para a base de conceituação e desenvolvimento do trabalho, como o uso do ludismo e de jogos educativos como suporte de ensino, exploração de modelos de desenvolvimento de jogos educacionais, conceitos e modelos de gerenciamento de riscos e de projeto e sistemas *fuzzy* e agentes inteligentes.
- Seção 3 - Desenvolvimento do Trabalho: descreve o levantamento dos requisitos do jogo, sua conceituação e seu desenvolvimento.
- Seção 4: avaliação do jogo, utilizando um modelo de avaliação específico para avaliar jogos educativos.
- Seção 5: conclusão do trabalho, apresentação do resultado final do jogo e considerações finais.

2 Fundamentação Teórica

Nesta seção são detalhadas a fundamentação teórica, incluindo estudos sobre gerenciamento de riscos, metodologias para desenvolvimento de jogos educacionais e métodos de avaliação destes, além de trabalhos correlatos e estado da arte da área de jogos educacionais.

2.1 Educação Lúdica e Jogos Educativos

Serious games são jogos cujo propósito principal não é o entretenimento, mas não necessariamente têm como intenção a educação, então os jogos educacionais, subgrupo de *serious games*, têm esse como propósito principal. Essa categoria de jogo está sendo utilizada cada vez mais como forma alternativa de ensino e de aumentar interesse dos alunos.

A educação no modelo tradicional tem se mostrado exaustiva e monótona para os estudantes, em destaque para a educação superior, onde o docente muitas vezes tem pouco acesso a métodos pedagógicos e se limita apenas à transmissão clássica de conhecimento verbal e escrito (GALVÃO et al., 2011; D'ÁVILA, 2014; GARRIS; AHLERS; DRISKELL, 2017). Isso somado à digitalização cada vez maior ocorrendo em diversos setores culturais e que transforma diretamente o comportamento dos jovens, distancia estes do aprendizado tradicional (ANASTASIADIS; LAMPROPOULOS; SIAKAS, 2018). Assim sendo, o uso de atividades lúdicas vem sendo estudado para renovar e transformar a educação.

A capacidade de retenção de interesse e de adesão de jogos educativos pode ser observada no trabalho de (CARDOSO, 2019), um estudo de diversos trabalhos com jogos apresentados para pacientes de tratamento em saúde mental, que mostrou maior adesão ao tratamento pelos pacientes após o jogo.

Também foi pontuado que o potencial de reter interesse dos jogos tem sido estudado desde a década de 1980, com o surgimento dos fliperamas e *arcades* (SQUIRE, 2003). Além desses estudos, outros também apontaram que a presença das mecânicas de jogos, como desafios, progressão, objetivos claros e *feedback*, são elementos fundamentais para se tornarem tão atrativos (CARDOSO, 2019; GARRIS; AHLERS; DRISKELL, 2017). Estudos demonstraram que até mesmo os professores tiveram mais engajamento com o conteúdo que precisavam transmitir e até conseguiram focar nos tópicos mais importantes por conta do auxílio visual e de mecânicas de jogo, que ajudaram a destacar pontos importantes dos conteúdos lecionados (CAMPANHA; CAMPOS, 2019). Outros pontos positivos foram observados, como a melhoria de habilidades cognitivas e uma retenção de conteúdo prolongada (ZHONGGEN, 2019; CAULFIELD et al., 2011; KOSA et al., 2016; SANTOS et al., 2020).

Backlund e Hendrix (2013) realizaram um estudo por diversos trabalhos, chegando a um total de 40 estudos que apresentaram resultados empíricos, para verificar a eficiência de jogos

educacionais. O estudo obteve resultados que apontam para uma significativa melhora com a aplicação dos jogos, chegando a um total de 29 estudos com resultados positivos, 7 neutros, 2 negativos e 2 incertos.

Jogos educativos se mostraram eficientes em diversos cenários e o uso no ensino superior também encontrou seu espaço, principalmente para renovar o interesse e para praticar os conhecimentos obtidos nas aulas. D'Ávila (2014) apontou a dificuldade de muitos docentes em obter a concentração dos discentes, havendo uma desconexão da didática aplicada. Assim, o estudo analisou métodos lúdicos, trazendo o uso de jogos educativos para o centro da pesquisa. O que se observou foi que uma atividade prática e que exercita não só a leitura de conteúdo, como os jogos, obtiveram um resultado positivo na visão pedagógica de professores.

O uso dos jogos no ensino superior apresentou resultados além da retenção de interesse. No estudo de Hoppe e Kroeff (2014) foi apontado que o aprendizado cognitivo obteve uma melhora considerável e os jogos se mostraram um ótimo recurso para apresentar cenários novos, ensinando através da mudança de comportamento, o que coloca o estudante em uma situação de atenção e entrega maior.

Outros trabalhos apontam que o desenvolvimento e uso de jogos educativos não conseguem abranger uma grande quantidade de conteúdos sem um desenvolvimento custoso e também que é muito difícil transpor diretamente o que é aprendido no jogo para a realidade, sendo necessária uma interpretação com auxilio de mais informações, o que levou diversos estudos à conclusão de que os jogos educativos poderiam servir como excelentes materiais de apoio (CAULFIELD et al., 2011; KOSA et al., 2016; OLIVEIRA et al., 2018; BACKLUND; HENDRIX, 2013).

2.2 Desenvolvimento de Jogos Educacionais

Foram pesquisadas metodologias de desenvolvimento de jogos educacionais no intuito de ajudar a obter o melhor design, unindo o conteúdo de interesse deste trabalho, gerenciamento de riscos, com mecânicas de jogo e objetivos pedagógicos e de engajamento.

Alguns métodos e conceitos recorrentes foram observados, como a utilização de *fuzzy systems* e agentes inteligentes (OLIVEIRA; CINTRA; NETO, 2013; GALVÃO et al., 2011) para controlar eventos e, no campo de pedagogia e educação, a taxonomia de Bloom (SANTOS et al., 2020).

2.2.1 Metodologias de Desenvolvimento de Jogos Educacionais

Na pesquisa de Santos et al. (2020) foram apontados diferentes *frameworks* utilizados em jogos educativos brasileiros, como o ENgAGED (BATTISTELLA; WANGENHEIM, 2016), um processo de desenvolvimento iterativo, unindo características de design de jogos a instrucionais, e o AIMED (ROCHA et al., 2017), que propõe uma abordagem ágil para o desenvolvimento. Os jogos analisados nesse estudo também se basearam na taxonomia de Bloom para destacar seus

objetivos educacionais e o sistema de avaliação de jogos educativos MEEGA+ foi utilizado em alguns dos trabalhos.

O método AIMED consiste em 5 macroprocessos e 14 microprocessos iterativos, englobando processos organizacionais, de produção, de pré-produção, pós-produção e de suporte.

O modelo ENgAGED foi escolhido para este trabalho por apresentar uma estrutura bem definida e apresentar resultados de testes de aplicação, obtendo boas análises quanto a não ambiguidade, consistência, completude, compreensibilidade, corretude, flexibilidade, usabilidade e utilidade (BATTISTELLA; WANGENHEIM, 2016). O modelo é formado por fases e atividades dos processos de desenvolvimento de jogos educacionais, produzindo uma Unidade Instrucional (UI), envolvendo as fases de análise, projeto, desenvolvimento, execução e avaliação, que é um conjunto de atividades organizadas em tópicos ou temas (BATTISTELLA; WANGENHEIM, 2016). O Quadro 1 resume as fases e atividades do método ENgAGED.

Quadro 1 – Resumo das fases do método ENgAGED

Fase 1: Descrição da Unidade Instrucional (UI)	
A1.1 Especificar UI do jogo	Especificar conteúdo da UI (conteúdo, curso, disciplina, etc).
A1.2 Caracterizar aprendizes	Classificar público alvo (idade, jogos favoritos, gêneros de jogos favoritos etc) e caracterizar o ambiente de aplicação.
A1.3 Definir objetivo(s) de desempenho	Definir o(s) objetivo(s) que avaliam o desempenho do aluno ao final da UI.
Fase 2: Projeto da Unidade Instrucional	
A2.1 Definir avaliação do aluno	Definir como será estruturada a avaliação, para o aluno aprender com o jogo (avaliação dentro do jogo, com *feedback* de acertos e erros).
A2.2 Definir conteúdo da estratégia instrucional	A estratégia instrucional é Jogo Educacional. Assim, nesta atividade define-se o conteúdo e o sequenciamento do conteúdo ao longo do jogo.
A2.3 Decidir pelo desenvolvimento ou utilizar jogo desenvolvido	Decidir pelo desenvolvimento de um jogo educacional, ou pela utilização de um jogo existente.
A2.4 Revisar o modelo de avaliação do jogo	Revisar o modelo utilizado para avaliar o jogo educacional.
Fase 3: Desenvolvimento de um Jogo Educacional	
Fase 3.1: Análise do Jogo	
A3.1.1 Levantar requisitos do jogo	Levantar os requisitos para identificação das funções e funcionalidades do jogo. Define-se também a distribuição do conteúdo pelo jogo.

Fase 3.2: Concepção do Jogo	
A3.2.1 Conceber o jogo	Conceber o jogo, descrevendo as principais características, como objetivos do jogo, narrativa, regras, mecânica, elementos do jogo, pontuações e *feedback* educacional.
Fase 3.3: Design do Jogo	
A3.3.1 Definir linguagem de programação ou *game engine*	Definir a linguagem de programação ou *game engine* que será utilizada para o desenvolvimento do jogo.
A3.3.2 Produzir ilustrações ou imagens dos elementos do jogo	Produzir as ilustrações que representam os elementos do jogo. Normalmente os elementos são personagens, cenários, objetos, artefatos, menus ou janelas de opções/configurações do jogo, tabuleiros, cartas, fichas, dados, peças de montar (lego), etc.
A3.3.3 Modelar o jogo	Modelar os níveis do jogo, as bibliotecas adicionadas à *game engine* ou à linguagem de programação, os *feedbacks* educacionais e os diálogos dos personagens.
Fase 3.4: Implementação do Jogo	
A3.4.1 Produzir elementos do jogo	Codificação e programação dos elementos do jogo ou produção dos elementos não-digitais.
Fase 3.4: Testes do Jogo	
A3.5.1 Realizar testes do jogo	Realizar testes para detecção de erros e *feedbacks* para melhoria do jogo.
Fase 4: Execução da Unidade Instrucional	
A4.1 Planejar a execução do jogo	Planejar a execução do jogo definindo data para jogar, local aonde será jogado, e equipamentos que serão utilizados ou materiais que devem ser impressos.
A4.2 Instalar o jogo digital	Instalar o jogo conforme a sua plataforma caso seja necessário.
A4.3 Executar o jogo	Executar o jogo no ambiente escolhido.
Fase 4: Avaliação da Unidade Instrucional	
A5.1 Conduzir avaliação	Conduzir a avaliação após a execução do jogo utilizando o instrumento para coleta de dados definidos na atividade A2.4 Revisar o modelo de avaliação do jogo.

A5.2 Analisar dados da avaliação	Realizar a análise dos dados coletados por meio de estatística descritiva, respondendo as questões de análise definidas na atividade

Fonte: Baseado em (BATTISTELLA; WANGENHEIM, 2016, p. 383-385)

2.2.2 Agentes Inteligentes

Os agentes inteligentes (AI) têm sido procurados para uso em jogos digitais pela sua capacidade de interação entre o ambiente e vários agentes, além de seus atributos de reatividade e pró-atividade, permitindo um controle adaptável sobre os elementos de jogo (OLIVEIRA; CINTRA; NETO, 2013; DZIUK; MIIKKULAINEN, 2011; GALVÃO et al., 2011).

Agentes, em computação, são sistemas de computador autônomos situados em um ambiente, seja físico ou digital, capaz de realizar ações sobre este ambiente através de um ator (ZADEH, 1965; WOOLDRIDGE; JENNINGS, 1995; FRANKLIN; GRAESSER, 1996). Essa definição, porém, não diz respeito a agentes inteligentes, pois falta o atributo da autonomia. Dessa forma, um AI é definido como um agente que possui os atributos: reatividade, pró-atividade e habilidades sociais.

Os conceitos apresentados a seguir foram retirados do trabalho de (WOOLDRIDGE, 1999).

Reatividade é a capacidade de observar o ambiente em que o agente está situado e responder-lhe; pró-atividade é a capacidade do agente, tendo funções e métodos definidos, realizar ações em busca de um objetivo; e habilidades sociais é a qualidade de um agente se comunicar com outros agentes ou com um humano. Segundo ele, a reatividade nem sempre será efetiva, já que por si só não busca um objetivo, apenas reage e a pró-atividade também nem sempre será efetiva sozinha, pois não reage bem a um ambiente dinâmico, por isso é preciso ter uma combinação dos dois atributos. Além disso, as habilidades sociais são importantes para integrar o sistema e até comunicar com outros sistemas diferentes.

Wooldridge também destaca a diferença entre AI e objetos de orientação a objeto. A principal diferença está na forma em que estes dois conceitos agem: objetos possuem métodos internos e públicos, que podem ser acessados, assim como os AI, porém, a comunicação com tais métodos públicos dos objetos não possuí controle rígido, podendo ocorrer acessos feitos por qualquer entidade e que nem sempre serão de interesse do objeto. Já com os agentes, os acessos são feitos a partir de requisições, permitindo maior controle das informações trocadas entre os agentes.

Tendo em vista que um dos principais atributos de um AI é o de habilidades sociais, ou seja, comunicação com outros agentes, é implícita a existência de um sistema multi-agente, onde existem diferentes AI exercendo suas atividades e se comunicando. Uma característica importante de sistemas multi-agente é que um agente possui suas próprias tarefas e metas, não existindo apenas em função de um outro. Assim, cada AI estará exercendo seus atributos de

reatividade, pró-atividade e de habilidades sociais no sistema, escolhendo quando requisitar ou enviar informações para outros agentes.

A arquitetura básica de um AI é dividida nos elementos **ambiente**, **ação** e **percepção**. $S = s_1, s_2, ...$ é o conjunto de estados de ambiente e a qualquer momento o ambiente pode assumir qualquer destes estados. O conjunto das ações capazes de serem realizadas por um AI é $A = a_1, a_2, ...$

A abstração de um agente pode ser dada por 2.1, que mapeia as sequências de estados de ambiente para ações.

$$ação : S^* \rightarrow A \tag{2.1}$$

Assim, é feito um histórico de estados possíveis, baseando-se nas ações que podem ser tomadas, ou seja, o comportamento dos estados pode ser modelado como a função 2.2, que, sendo $s \in S$ e $a \in A$, mapeia um conjunto de estados $ambiente(s,a)$.

$$ambiente : S \times A \rightarrow \rho(S) \tag{2.2}$$

Assim, definimos o histórico h como a sequência:

$$h : s_0 \rightarrow^{a_0} s_1 \rightarrow^{a_1} s_2 \rightarrow^{a_2} ... \rightarrow^{a_{u-1}} s_{u-1} \rightarrow^{a_u} s_u \tag{2.3}$$

Essa sequência caracteriza um possível histórico de estados conquanto se mantenham as condições 2.4 e 2.5:

$$\forall u \in N, a_u = ação((s_0, s_1, s_2, ..., s_u)) \tag{2.4}$$

$$\forall u \in N / u > 0, s_u \in ambiente(s_{u-1}, a_{u-1}) \tag{2.5}$$

O comportamento característico de um AI $ação : S^* \rightarrow A$ em $ambiente : S \times A \rightarrow \rho(S)$ é o conjunto de todos os históricos que satisfaçam essas propriedades. O conjunto de históricos é denotado como $hist(agente, ambiente)$.

Apesar dessa abstração ajudar a enxergar como funciona um agente inteligente, é preciso refiná-la. Isso é feito criando subsistemas. Uma visão superficial de uma dessas subdivisões é a divisão da função de decisão em *percepção* e *ação*, com as funções *ver* e *ação*. Assim, a função *ver* representa a capacidade do agente de perceber o ambiente em que está e a função *ação* representa o processo de decisão de ação. A saída da função *ver* é um conjunto *percepção* P e é denotada como:

$$ver : S \rightarrow P \tag{2.6}$$

que então mapeia os estados, agora com *ação* sendo uma função

$$ver : P^* \rightarrow A \tag{2.7}$$

que converte uma sequência de percepções em ações.

Essa definição baseada em históricos, no entanto, não é intuitiva. Portanto, transpõe-se esse conceito em um outro em que os agentes mantêm *estados*. Seja I um conjunto de estados de um AI, a decisão de um agente é baseada, em parte, neste estado.

Com esse conceito de estados, a função *ver* não se altera, permanecendo $ver : P^* \rightarrow A$, e agora a seleção de ação será dada como um mapeamento de estados internos em ações:

$$ação : I \rightarrow A \tag{2.8}$$

Então uma nova função, *prox*, é introduzida para fazer o mapeamento de um estado a partir da *percepção*:

$$ambiente : I \times P \rightarrow I \tag{2.9}$$

Em resumo, um agente inicia em um estado inicial i_0 e então observa o estado do ambiente s, gerando uma *percepção* $ver(s)$. O estado interno do agente então é atualizado pela função *prox*, gerando o conjunto $prox(i_0, ver(s))$. A ação selecionada será $ação(prox(i_0, ver(s)))$, sendo então realizada e o agente inicia um novo ciclo.

2.2.3 Lógica *Fuzzy*

A lógica *fuzzy* vem sendo utilizada em diversas aplicações, o que fez crescer bastante a sua disseminação e popularidade na indústria e entre a comunidade acadêmica (DEMASI; CRUZ, 2002). Sua versatilidade traz muitas vantagens para aplicações que precisam de flexibilidade ou trabalhem com incertezas.

Jogos digitais são aplicações que estão explorando essa lógica para compor a inteligência artificial de seus sistemas para obter maior versatilidade e para criar sensação de incerteza, conceito bastante utilizado em jogos para passar a sensação de imprevisibilidade e causar surpresa nos jogadores, sem também perder o controle das ações do jogador (AUDI, 2014).

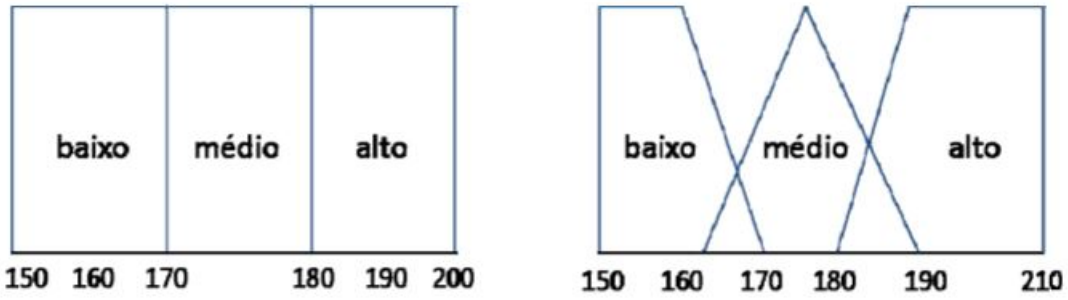
Um dos exemplos mais comuns para explicar a lógica *fuzzy* é a classificação de altura de uma pessoa. Tomando 3 classes para altura, baixo, mediano e alto, e tomando alguém como baixo se sua altura for menor que 1,70 metros, mediana se estiver entre 1,70 metros e 1,80 metros e alta se a altura for maior que 1,80. Na lógica tradicional, uma pessoa com 1,69 metros seria classificada como baixa, mas ela está bem próxima de ser mediana e é "mais mediana"que uma pessoa com 1,51 metros, e uma pessoa com 1,71 é classificada como mediana, mas está bem próxima de ser baixa e é "mais baixa"que uma alguém com 1,80 metros. Na lógica *fuzzy* usa-se graus de pertinência para classificação, com graus entre 0 e 1 (MARRO et al., 2010).

Marro et al. (2010) dá como exemplo de comparação, considerando um conjunto A e um elemento x com relação a esse conjunto, uma função da teoria tradicional (2.10) e outra da teoria *fuzzy* (2.11) e a Figura 2 ajuda a visualizar o conceito da lógica *fuzzy*.

$$f(x) = \begin{cases} 1, & se, \text{e somente se, } x \in A \\ 0, & se, \text{e somente se, } x \notin A \end{cases} \tag{2.10}$$

$$\mu(x) = \begin{cases} 1, & \text{se, e somente se, } x \in A \\ 0, & \text{se, e somente se, } x \notin A \\ 0 \leq \mu(x) \leq 1, & \text{se, } x \text{ pertence parcialmente a } A \end{cases} \tag{2.11}$$

Figura 2 – Representação de conjuntos de altura de uma pessoa comparando lógica tradicional (esquerda) com lógica *fuzzy* (direita)



Fonte: (MARRO et al., 2010, p. 3)

Regras *fuzzy* são determinadas para que a lógica opere da maneira correta. Para determinar as regras, é necessário um raciocínio divido em duas partes: determinação do antecedente e aplicação do resultado no antecedente, gerando um consequente. Isso se dá na análise dos graus de pertinência e, dependendo da lógica pretendida (*e* ou *ou*) obter um grau de pertinência resultante. No caso do conectivo *e*, o grau de pertinência resultante não pode ultrapassar o valor do menor grau de pertinência individual. Já para o conectivo *ou*, o grau de pertinência resultante não pode ser menor que o maior valor individual de um grau de pertinência.

O processo de avaliar as entradas e, através das regras predefinidas, obter conclusões utilizando a teoria *fuzzy* é feito através de modelos de inferência e o modelo mais utilizado é o modelo de Mandami, composto das etapas: fuzzyficação, avaliação das regras fuzzy, agregação das regras fuzzy e defuzzyficação.

Na etapa de fuzzyficação obtêm-se os graus de pertinência de cada entrada de cada conjunto *fuzzy*. A etapa de avaliação das regras aplica as regras nos antecedentes, gerando os consequentes. No caso de antecedentes compostos, aplica-se um operador *e* ou *ou* para obter um resultado único. O operador *e* realiza uma operação de interseção (obtém o menor grau de pertinência) e o operador *ou* realiza a operação de união (obtém o maior grau de pertinência). Em seguida, um método de correlação é aplicado sobre o consequente, um dos métodos mais utilizados é

o *clipped*, onde o valor obtido é simplesmente passado para o consequente da regra aplicada. Na agregação das regras *fuzzy*, todas as funções dos consequentes são agrupadas em um único conjunto *fuzzy*. Na etapa de defuzzificação será obtida uma saída numérica a partir da saída da etapa anterior através da técnica do centroide, que obtém um ponto onde uma linha vertical divide ao meio um conjunto agregado. A fórmula é dada a seguir (2.12).

$$COG = \frac{\sum_{x=a}^{b} \mu(x) \times x}{\sum_{x=a}^{b} \mu(x)} \tag{2.12}$$

Marro et al. (2010) ilustrou esse método da seguinte forma: vamos considerar a análise de riscos em um projeto e 3 variáveis (duas de entrada e uma de saída), apresentadas no Quadro 2.

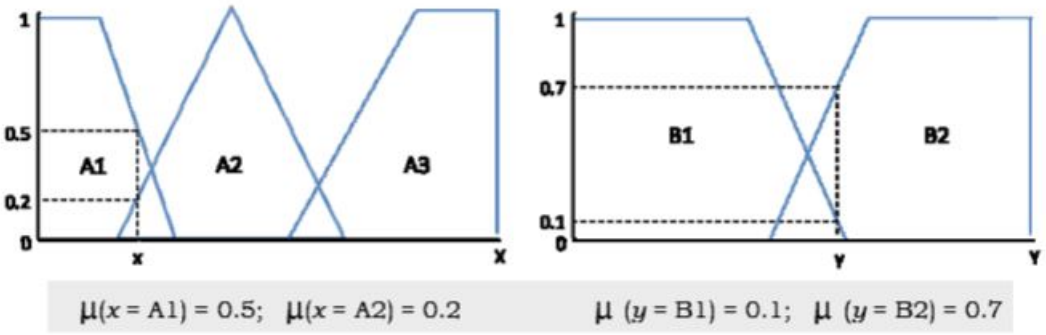
Quadro 2 – Variáveis e valores linguísticos de análise de riscos em um projeto

Fundos do Projeto (x)	
Valor Linguístico	Notação
Inadequado	A1
Razoável	A2
Adequado	A3
Funcionários do Projeto (y)	
Valor Linguístico	Notação
Pequeno	B1
Grande	B2
Risco do Projeto (z)	
Valor Linguístico	Notação
Baixo	C1
Normal	C2
Alto	C3

Fonte: (MARRO et al., 2010, p. 8-9)

Na etapa de fuzzyficação, têm-se os conjuntos fuzzy e graus de pertinência para cada uma das variáveis de entrada, conforme mostrado na Figura 3.

Figura 3 – Fuzzyficação das variáveis x e y resultando em seus respectivos graus de pertinência



Fonte: (MARRO et al., 2010, p. 9)

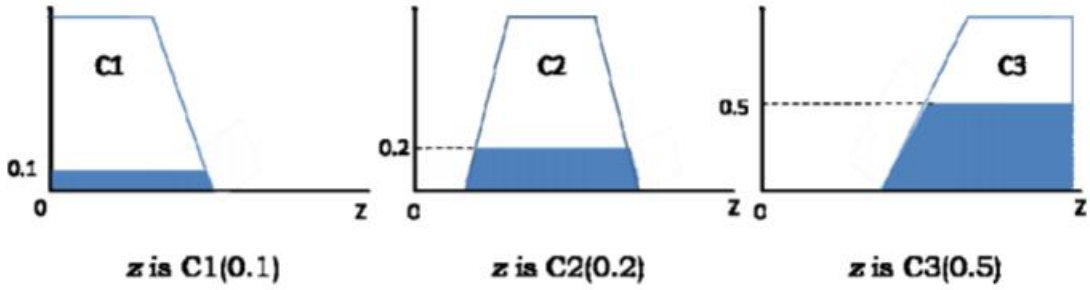
Na avaliação das regras *fuzzy* temos as regras:

1. SE (x é $A3(0)$ ou y é $B1(0.1)$) ENTÃO (z é $C1(0.1)$)
2. SE (x é $A2(0.2)$ e y é $B2(0.7)$) ENTÃO (z é $C2(0.2)$)
3. SE (x é $A1(0.5)$ ENTÃO (z é $C3(0.5)$)

Observa-se que na regra 1, com o operador *ou*, obtêm-se grau de pertinência 0.1 para z, ou seja, o maior dos graus de pertinência de entrada, e na regra 2, com o operador *e*, obtêm-se grau de pertinência 0.2 para z, o menor grau de pertinência de entrada. Já na regra 3, foi aplicado o *clipped*, já que existe apenas um grau de pertinência de entrada, é passada a variável de saída para o consequente.

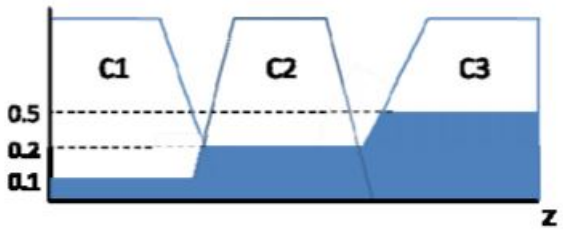
Pela Figura 4 vê-se os conjuntos consequentes para cada grau de pertinência produzido pelas regras e a Figura 5 apresenta a agregação desses conjuntos em um único conjunto *fuzzy*.

Figura 4 – Conjuntos *fuzzy* e graus de pertinência da variável de saída z e respectivos graus de pertinência, produzidos pelas regras *fuzzy* 1, 2 e 3



Fonte: (MARRO et al., 2010, p. 10)

Figura 5 – Conjunto resultado da agregação das regras *fuzzy* 1, 2 e 3



Fonte: (MARRO et al., 2010, p. 11)

Por fim, na etapa de defuzzyficação, considerando o conjunto da Figura 5, o resultado é dado, considerando intervalos percentuais de 10%, de 0% até 100%, como na equação

$$COG = \frac{(0+10+20)\times 0.1+(30+40+50)\times 0.2+(60+70+80+90+100)\times 0.5}{0.1+0.1+0.1+0.2+0.2+0.2+0.5+0.5+0.5} = 67.4 \tag{2.13}$$

Assim temos que o risco do projeto em questão é de 67.4%

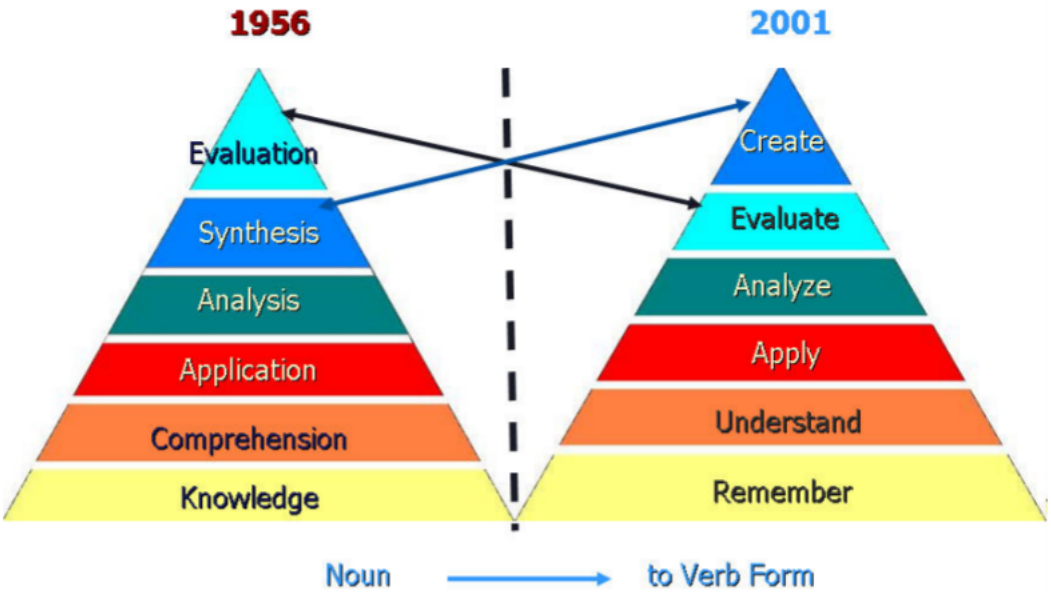
2.2.4 Avaliação de Jogos Educacionais

Apesar do uso de jogos educacionais estar se popularizando e apresentar resultados promissores, estes resultados nem sempre são confiáveis por falta de um processo científico de avaliação. Por isso, alguns métodos de avaliação foram desenvolvidos.

A taxonomia de Bloom (BLOOM; KRATHWOHL; MASIA, 1984) foi criada para classificar objetivos e metas educacionais no domínio cognitivo e pode ser usada para definir pontos importantes a se avaliar em um jogo educacional. Bloom et al. (1956) arranjaram o que professores queriam que estudantes aprendessem em uma hierarquia de menos para mais complexo. Essas divisões foram ordenadas como conhecimento, compreensão, aplicação, análise, síntese e avaliação. Essa classificação perdurou até o começo dos anos 2000, quando foi feita uma revisão da taxonomia de Bloom (ANDERSON; KRATHWOHL, 2001), que também considerou críticas e preocupações do próprio Bloom para a reformulação (WILSON, 2016).

Nessa revisão, Anderson e Krathwohl (2001) transformaram a nomenclatura de substantivos para verbos e alteraram a ordem de dois níveis para se adequar a uma abordagem mais moderna com objetivos voltados a resultados (HUITT, 2011). Na Figura 6 é possível ver as mudanças realizadas.

Figura 6 – Taxonomias do Domínio Cognitivo - 1956 x 2001



Fonte: (WILSON, 2001, p. 4)

As principais diferenças entre a taxonomia de Bloom e a revisada por Anderson e Krathwohl (2001) são a renomeação e reposicionamento de níveis e em como a revisada se organiza em função de maior compreensão e utilidade de definições sobre como ela age sobre diferentes tipos de conhecimento.

Para avaliação de jogos educacionais, a escala EGameFlow (FU; SU; YU, 2009) utiliza uma metodologia sistemática avaliando análise de itens, confiabilidade e validade, apresentando resultados satisfatórios, porém esta escala foi descontinuada pelos autores.

Outros estudos utilizando *learning analytics* foram feitos, porém, eles apenas avaliam o aprendizado, desconsiderando avaliações de engajamento, motivação e diversão (FREIRE et al., 2016; SERRANO-LAGUNA et al., 2018)

O MEEGA (*Model for the Evaluation of Educational GAmes*) (SAVI; WANGENHEIM; BORGATTO, 2011) é um dos modelos mais utilizados (PETRI; WANGENHEIM, 2017), porém, ele ainda apresentava possibilidades de melhoria segundo avaliação de alunos. Então foi desenvolvido o MEEGA+ (PETRI; WANGENHEIM; BORGATTO, 2019), para avaliar a percepção da qualidade do ponto de vista dos discentes de computação. Este modelo de avaliação divide fatores em dimensões e subdimensões como é visto no Quadro 3.

Quadro 3 – Definição das Dimensões/Subdimensões do MEEGA+

Atenção Focada		Avaliar a atenção, concentração focada, absorção e dissociação temporal dos alunos
Diversão		Avaliar a sensação de prazer, felicidade, relaxamento e distração dos alunos
Desafio		Avaliar quanto o jogo é suficientemente desafiador em relação ao nível de competência do aluno. Novos obstáculos e situações devem ser apresentados ao longo do jogo para minimizar a fadiga e manter os alunos interessados
Interação Social		Avaliar se o jogo promove a sensação de um ambiente compartilhado e conexão com outras pessoas em atividades de cooperação ou competição
Confiança		Avaliar se os alunos são capazes de progredir no estudo do conteúdo educacional por meio de seu esforço e habilidade
Relevância		Avaliar se os alunos percebem que a proposta educacional é consistente com seus objetivos e que podem vincular o conteúdo ao futuro profissional ou acadêmico
Satisfação		Avaliar se os alunos sentem que o esforço dedicado resulta em aprendizagem
Usabilidade	Aprendizibilidade	Avaliar se o jogo permite que os usuários aprendam a jogá-lo de forma fácil e rápida
	Operabilidade	Avaliar o grau em que um jogo possui atributos que facilitam a operação e o controle
	Estética	Avaliar se a interface do jogo permite uma interação agradável e satisfatória com o usuário
	Acessibilidade	Avaliar se o jogo pode ser usado por pessoas com deficiência visual baixa/moderada e/ou com daltonismo
	Proteção contra erros do usuário	Avaliar se o jogo protege os usuários de cometer erros. Aplicado apenas para avaliação de jogos digitais.
Aprendizagem Percebida		Avaliar as percepções do efeito geral do jogo na aprendizagem dos alunos na disciplina

Fonte: Baseado em (PETRI; WANGENHEIM; BORGATTO, 2019, p. 59)

No trabalho de Petri, Von Wangenheim e Borgatto (2019) também foi definido o questionário do modelo MEEGA+ para avaliação dos jogos educacionais, consistindo de 35 questões distribuídas pelas dimensões definidas e complementa que itens referentes ao conteúdo específico sendo abordado pelo jogo devem ser incluídos no questionário.

2.3 Gerenciamento de Riscos

O gerenciamento de riscos é uma das tarefas mais importantes de um gerente de projeto. É o processo de antecipar e mitigar riscos que possam afetar o desenvolvimento e entrega de um *software* (SOMMERVILLE, 2016). Os gerentes de projeto são os responsáveis por organizar o levantamento de possíveis riscos, analisar seus impactos, sejam positivos ou negativos, e elaboram um planejamento para lidar com eles.

Secundo (SOMMERVILLE, 2016), os riscos podem ser divididos em três categorias:

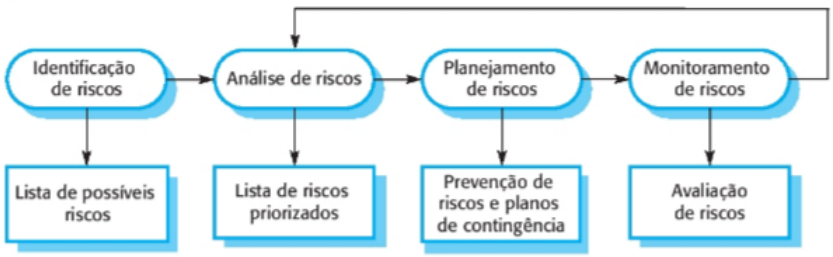
- **Riscos de Projeto:** afetam cronograma ou recursos utilizados durante o projeto;
- **Riscos de Produto:** afetam a qualidade ou desempenho do *software* desenvolvido;
- **Riscos de Negócios:** afetam a organização que está desenvolvendo o *software*.

Ainda de acordo com Sommerville (2016), o gerenciamento de riscos é composto de 4 etapas:

1. **Identificação de riscos:** identificar possíveis riscos de projeto, produto ou negócios;
2. **Análise de riscos:** avaliar probabilidades e consequências dos riscos;
3. **Planejamento de riscos:** fazer planos para abordar os riscos;
4. **Monitoramento de riscos:** avaliar regularmente os riscos e agir se necessário.

Ao passo de cada etapa, documentos sobre os resultados são gerados para ajudar durante o gerenciamento e monitoramento no processo de gerenciamento de riscos. O fluxo do gerenciamento de riscos pode ser visualizado na Figura 7.

Figura 7 – Processo de Gerenciamento de Riscos



Fonte: (SOMMERVILLE, 2016, p. 612)

2.3.1 Metodologias de Identificação de Riscos

Para auxiliar no processo de gerenciamento de riscos, diversos métodos foram desenvolvidos e padronizados. Alguns dos mais usados são o método Delphi, SWIFT, árvore de decisão, análise *bow tie*, matriz de riscos (ou matriz de probabilidade/consequência) e a análise SWOT (*Strenghts, Weaknesses, Opportunities, Threats*) (ISO/IEC, 2019).

O método Delphi envolve coletar a opinião de diversos especialistas de forma individual e não presencial e então fazer estes especialistas revisarem os riscos apontados pelos outros (DALKEY; HELMER, 1963). Este método é bastante popular, principalmente no meio empresarial, por ser possível a coleta de informações à distância, porém é mais lento para chegar a conclusões.

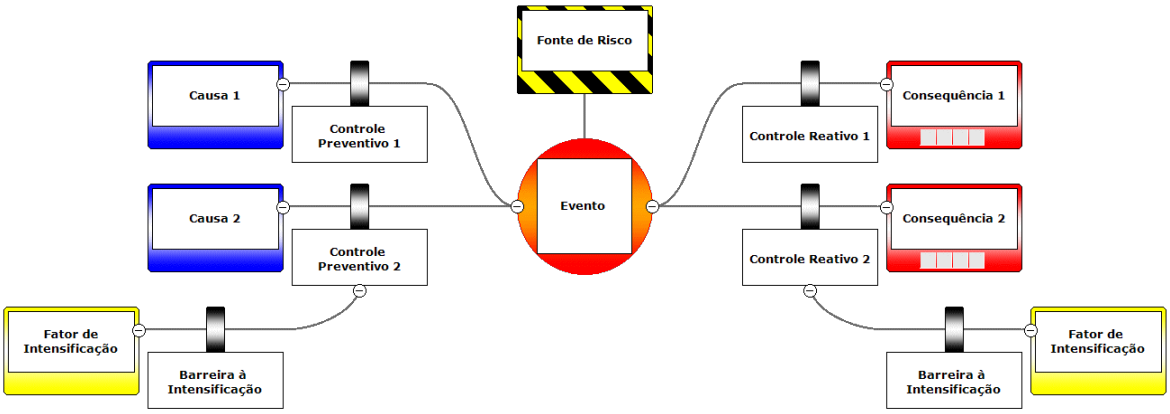
O método SWIFT (*Structured What-If Technique*) (CARD; WARD; CLARKSON, 2012) se baseia em um *brainstorm* em equipe para realizar diversas questões de *"what if"* ("e se") para

investigar possíveis riscos. Ele é especialmente útil para identificar oportunidades (riscos com consequências positivas).

O método de árvore de decisão é um processo de análise em que se propõem decisões que levam a uma cadeia de consequências e novas decisões, mapeando os diferentes caminhos possíveis a seguir e as probabilidades de cada um desses caminhos.

A análise *bow tie* é uma das mais práticas para identificar mitigação de riscos. Ela se baseia em olhar para os possíveis riscos, individualmente, em duas direções, esquerda para potenciais causas de riscos e direita para potenciais meios de combater as suas consequências. O modelo *bow-tie* é exemplificado na Figura 8.

Figura 8 – Metodologia *Bow Tie*



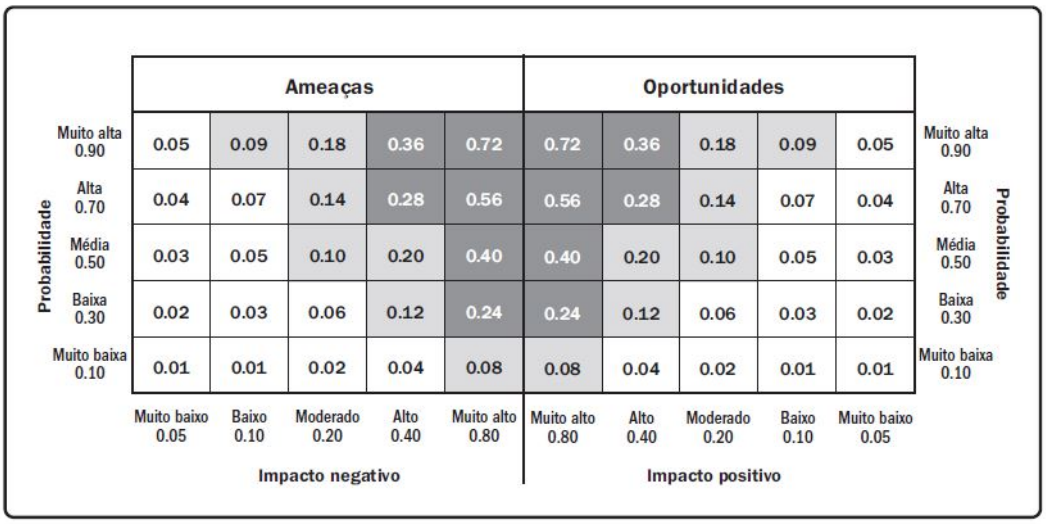
Fonte: https://iso31000.net/bowtie/

[1]

A matriz de riscos é um meio de combinar meios qualitativos e semiquantitativos de consequências e probabilidades (ISO/IEC, 2019). É útil quando muitos riscos são identificados e fornece uma visualização mais ampla e ajuda a decidir quais riscos necessitam de uma análise mais aprofundada ou quais necessitam de tratamento imediato, além de ajudar a determinar riscos aceitáveis e oportunidades. Em contrapartida, sua utilização pode ser ambígua e subjetiva e é difícil de analisar riscos combinados. A estrutura básica de uma matriz de riscos, incluindo uma separação para oportunidades, é mostrada na Figura 9.

[1] Acessado em 07/2021

Figura 9 – Matriz de Riscos (Probabilidade/Consequências)



Fonte: (INSTITUTE, 2017, p. 408)

A análise SWOT é uma estratégia para identificar pontos fortes e fracos que podem ajudar ou atrapalhar uma organização. Ela se divide em: *strenghts* - de competência interna à organização, são recursos ou atributos de valor que a empresa pode utilizar; *weaknesses*: falta de competência, recursos ou atributos internos à empresa que poderiam ser úteis; *opportunities*: uma oportunidade externa à organização a qual ela pode buscar para obter benefícios; *threats*: fatores externos à organização que podem afetar negativamente o desempenho (LEIGH, 2009). A utilidade dessa análise está na divisão em quatro categorias, facilitando a identificação individual dos fatores e categorizando-os, além de identificar atributos importantes da própria empresa, como pode ser visto no Quadro 4.

Quadro 4 – Exemplo de Aplicação da Análise SWOT

Internal Factors	
Strengths	**Weaknesses**
Market dominance Core skills Economies of scale Low-cost position Leadership and management Manufacturing ability Age of equipment Innovative processes and products Architecture network Reputation Differentiated products Product or service quality	Market share weakness Few core strengths and low key skills Old equipment, higher costs than the competition Weak finances and poor cash flow skills Management skills and leadership lacking Poor record of innovation Weak organization with poor architecture Low quality and poor reputation Products not differentiated Dependent on few products
External Factors	
Opportunities	**Threats**
New markets and segments New products Diversification opportunities Market growth Competitor weakness Demographic and social change Change in political, economic environment New takeover or partnership opportunities Economic upturn International growth	New market entrants Increased competition Increased pressure from customers and suppliers Substitutes Low market growth Economic cycle downturn Technological threat Change in political or economic environment Demographic change New international barriers to trade

Fonte: (LEIGH, 2009, p. 119)

2.3.2 Análise de Riscos

Os estudos sobre análise de riscos a seguir foram efetuados a partir do Guia PMBOK (2017).

A análise qualitativa de riscos se trata de selecionar riscos individuais para maior análise, prosseguindo com cálculos de probabilidades, análise de impacto ou outras características. Essa tarefa é subjetiva, já que os riscos e suas características são dadas pela equipe e outros *stakeholders* e estabelece as prioridades para o plano de respostas aos riscos e a análise pode ser feita ao longo de todo o projeto. Alguns métodos podem ser usados para organizar os riscos, como a matriz de probabilidade/consequência, que pode ser utilizada apenas com descritores qualitativos ao invés de quantitativos.

A análise quantitativa de riscos, fornece uma visão dos riscos através de números e assim provendo uma análise robusta de cada risco individualmente. Essa análise não é necessária em todos os projetos e sua realização requer tempo e custos adicionais, além de especialização com modelos de riscos e de probabilidades. Normalmente é utilizada em grandes projetos.

Poderia ainda ser feita uma escala qualitativa de "muito improvável"até "altamente provável". Essa escala pode ser numérica, como, por exemplo $(0.1, 0.3, 0.5, 0.7, 0.9)$, onde cada valor

representa uma escala do risco ocorrer (LIMA, 2009).

Uma escala de impacto também pode ser feita, utilizando-se os descritores "muito baixo", "baixo", "moderado", "alto"e "muito alto", refletindo impactos cada vez maiores conforme definidos pela organização. Os valores dessa escala podem ser lineares ou não-lineares. Por exemplo, os valores lineares podem ser $(0.1, 0.3, 0.5, 0.7, 0.9)$ e os não lineares podem ser $(0.05, 0.1, 0.2, 0.4, 0.8)$. As escalas não-lineares podem representar o interesse da organização de evitar ameaças de alto impacto e explorar oportunidades de alto impacto.

Segundo o Guia PMBOK (2017), além da probabilidade e impacto dos riscos e oportunidades, pode-se realizar uma análise mais aprofundada e robusta utilizando outros parâmetros dos riscos. Alguns desses parâmetros são: urgência, proximidade, gerenciabilidade, controlabilidade, detectabilidade, conectividade e impacto estratégico. Outro fator que deve ser levado em consideração é o quanto os *stakeholders* desejam se arriscar para objetivos específicos, pois isso pode afetar as decisões de planejamento de respostas a determinados riscos que envolvem tais objetivos.

Em razão da complexidade de realizar a análise quantitativa e do foco do jogo desenvolvido neste trabalho ser de trabalhar a adesão e atenção de discentes para com o gerenciamento de riscos, essa análise será desconsiderada, efetuando-se apenas a análise qualitativa e trabalhando-se a partir dela enquadrando os riscos nas escalas, como apresentadas anteriormente.

2.3.3 Planejamento de Riscos

Nesta etapa são analisadas as probabilidades de ocorrência e graus de impacto dos riscos e oportunidades no projeto e então desenvolvidas respostas aos impactos buscando evitá-los ou mitigá-los se forem riscos, e aproveitá-los se forem oportunidades. É importante que as respostas sejam adequadas ao nível de importância de cada risco e que os recursos disponíveis sejam suficientes para respondê-los, considerando planos alternativos de contingência (INSTITUTE, 2017; LIMA, 2009).

O Guia PMBOK (2017) apresenta estratégias para planejar as respostas às ameaças e oportunidades. Para as ameaças, pode-se:

- **Evitar**: consiste em mudar o plano do projeto para eliminar o risco. É importante saber que nem todo risco pode ser evitado, mas realizando a identificação cedo no planejamento do projeto, a maioria pode ser impedida;
- **Transferir**: se trata de transferir o impacto do risco para terceiros com a propriedade da resposta. Essa estratégia transfere a responsabilidade do gerenciamento do risco para outra parte e geralmente é vantajosa com transações financeiras. Normalmente o serviço dessa outra parte é pago;

- **Mitigar**: essa estratégia é a mais adequada por abranger modelos e poder contar com experiências de trabalhos anteriores. Essa estratégia necessita que o risco seja antecipado pelo gerente de projetos.

Para as oportunidades, também são dadas três estratégias:

- **Explorar**: essa estratégia tenta eliminar as incertezas relacionadas à oportunidade com intuito de que ela de fato aconteça e inclui designação de recursos mais capacitados.
- **Compartilhar**: envolve compartilhar a oportunidade com terceiros, de modo a melhor aproveitá-la em prol do projeto. Ações compartilhadas são sugeridas como parcerias, equipes ou empresas de propósito específico.
- **Melhorar**: esta estratégia tem por objetivo aumentar as probabilidades de impactos positivos pela identificação dos principais causadores desses impactos e maximizá-los.

Existem ações que podem ser tomadas tanto para oportunidades quanto para ameaças, sendo elas:

- **Escalar**: quando uma ameaça está fora do escopo do projeto, escalando-o para que a resposta seja ao nível de programa, portfólio ou outra parte relevante da organização;
- **Aceitar**: ocorre quando um risco é identificado, mas nenhuma ação proativa é tomada. Pode ser adotada para riscos de baixa probabilidade e/ou baixo impacto, ou quando não existe a capacidade de lidar com o risco. Uma ação de aceitação proativa é garantir recursos de contingência, incluindo valores monetários e de tempo.

2.3.4 Monitoramento de Riscos

O monitoramento de riscos é a observação, feita ao longo do projeto, da implementação dos planos de resposta aos riscos identificados, além da identificação e análise de novos riscos ao decorrer do projeto e da avaliação de eficácia das soluções aplicadas. O monitoramento de riscos usa informações de desempenho geradas durante o projeto para determinar se as respostas aos riscos são efetivas, se o nível de risco do projeto ou de riscos individuais sofreram alterações, se surgiram novos riscos, se as premissas e abordagens do projeto ainda são válidas, se a estratégia e planos de contingência ainda são apropriados, etc. (INSTITUTE, 2017).

O monitoramento é feito a partir de análise de dados e de reuniões e auditorias. A análise dos dados pode ser feita em duas formas:

- **Análise do desempenho técnico**: defini-se medidas quantitativas de realizações técnicas do projeto, como prazos,números de defeitos entregues, capacidade de armazenamento, etc., e comparam-se essas métricas com as metas.

- **Análise de reservas**: a análise de reservas compara os recursos de contingência com os riscos restantes a qualquer momento no projeto para verificar se as reservas estão adequadas.

Além das análises de dados, é importante realizar auditorias e reuniões com frequência para a situação do projeto ser analisada e ser feita a revisão dos riscos.

2.4 Trabalhos Correlatos

Jogos educativos têm conquistado espaço na área de engenharia de *software* por sua capacidade de propor um cenário onde podem ser feitas atividades práticas dos conteúdos apresentados em sala de aula, atuando como complemento ao ensino ou até como treinamento (XIE; TILLMANN; HALLEUX, 2013).

DELIVER! é um jogo de tabuleiro (jogo não-digital) para o ensino de *Earned Value Management* (EVM) e gerenciamento de projetos de *software*. O jogo consiste em um tabuleiro segmentado em casas e dividido em setores, um para cada etapa do processo de desenvolvimento (requisitos, design de sistema, implementação e testes), onde os jogadores se deslocam uma quantidade de casas a partir da rolagem de um dado, e cartas representando humanos da equipe, com atributos de produtividade e salário, ou seja, o custo daquele funcionário e cartas de riscos com seus impactos e custos ao projeto. O jogo é jogado por quatro pares de jogadores, que devem competir para entregar um projeto primeiro sem ficar sem recursos. Os jogadores contam com tabelas de planejamento de projeto para fazer anotações do tempo e recursos gastos em cada etapa e fazer o planejamento do projeto. O jogo foi avaliado utilizando o modelo de quatro níveis de Kirkpatrick (reação, aprendizado, comportamento, resultados), coletando respostas dos jogadores através de um questionário, construindo gráficos para as respostas de cada questão (WANGENHEIM; SAVI; BORGATTO, 2012).

Sindre, Natvig Jahre (2008) desenvolveram o jogo ***Age of Computers*** em *web pages* com o conteúdo organizado em mapas com salas que continham objetos e imagens fornecendo uma formação de base para problemas e questões, numéricas ou de múltipla escolha, que os jogadores terão de resolver para conseguir pontos para liberar outras salas e vencer o jogo. As questões utilizadas cobrem variadas áreas do ensino da computação que envolvem inclusive programação. O jogador recebe *feedback* se acerta ou erra a resposta para os problemas e questões, bloqueando os problemas por alguns minutos caso a resposta seja errada, incentivando o estudo e leitura cautelosa antes de enviar a solução. Para manter interesse dos alunos, foi criada uma história simples sobre viagem no tempo para a época da criação do computador e implementado um placar com as pontuações de outros alunos, para incentivar a competitividade e a busca pela melhora. Uma forma de avaliar o desempenho do jogo foi a aplicação de um pré-teste e um pós-teste idênticos para serem comparadas as respostas de antes e depois de o jogo ser utilizado (SINDRE; NATVIG; JAHRE, 2008).

CleanGame (SANTOS et al., 2019) explora uma área pouco falada e abordada nas salas de aula, o de refatoração de código. Apesar de servir para aprendizado sobre o tema em questão, o trabalho não é um jogo, apenas sendo desenvolvida uma plataforma de treinamento utilizando gamificação, que se trata do uso de elementos de jogos em outra atividade com o intuito de torná-la mais interessante (DICTIONARIES, 2013), se utilizando de *quizes* e identificação de erros diretamente em códigos fornecidos. Os resultados coletados apontaram uma atitude positiva dos usuários à ferramenta e suas estratégias enquanto ferramenta de suporte, mas classificaram "motivação"e "foco"como pontos fracos.

Na área de engenharia de requisitos, Petri et al., (2016) desenvolveram, utilizando a metodologia ENgAGED, o jogo **Kahoot! PMQuiz**, um *quiz game* usado para revisão de conteúdo exposto previamente em sala de aula. O jogo incluiu perguntas de exames de certificação PMP (*Project Management Professional*) e de provas anteriores. O jogo exibe a quantidade de questões respondidas corretamente pelos alunos, mostrando quais alternativas foram selecionadas como resposta por outros jogadores e quantas vezes, e, em caso de erros, mostra as respostas corretas. O jogo foi avaliado utilizando o método MEEGA e apresentou melhora para o aprendizado e demonstrou ajudar com a motivação de estudo (PETRI et al., 2016).

Paludo, Raabe e Benitti (2013) desenvolveram o jogo educativo **RSKManager** para gerenciamento de riscos em projetos de *software* em que os jogadores podem detalhar o tipo de projeto que desejam trabalhar e então realizar o gerenciamento de riscos para cada etapa e acompanhar a situação do projeto e monitorar os riscos, recebendo um *feedback* parcial ao fim de cada uma delas, além de um relatório geral que pode ser acessado a qualquer momento durante o jogo. O jogo não permite que ações sejam desfeitas ou refeitas, carregando as consequências, custos e prazos para cada escolha tomada no jogo. Dessa forma, os erros e acertos do jogador ficam evidenciados pelo *feedback* e existe um limiar do quanto as decisões foram erradas, apresentado na definição do projeto, que irá ditar quando o jogo termina e é feito de forma a permitir margens para evitar que a experiência acabe muito cedo. Também é providenciada uma seção de apoio ao jogador com conteúdo sobre gerenciamento de riscos em projetos de *software* e informações sobre o jogo. Quando o jogo é finalizado, ele apresenta uma tela de fim de jogo mostrando o desempenho quanto ao prazo e custo do projeto. O jogo foi posto a teste e seus resultados mostraram que, apesar de não ter sido observado melhora expressiva no desempenho do aluno, estes se sentiram que o jogo ajudou a entender, lembrar e aplicar as atividades de gerenciamento de riscos (PALUDO; RAABE; BENITTI, 2013).

Galvão et al. (2011) apresentou um jogo educativo para suporte no treinamento de gerenciamento de riscos utilizando-se de agentes inteligentes e *fuzzy systems*. O jogo apresentado é acessado através do navegador e os jogadores têm um projeto para o qual irão identificar os riscos e suas probabilidades de ocorrer, além de monitorá-los ao longo do projeto (GALVÃO et al., 2011). O jogo foi criado para simular a experiência de gerenciar um projeto de *software*, com controle de gastos, busca de objetivos e cumprimento de cronograma e simulando mudanças de ambiente e de requisitos, além de riscos aparecendo durante o projeto, utilizando inteligência

artificial com os agentes inteligentes e a lógica *fuzzy*. O jogador controla também os funcionários trabalhando no projeto, escolhendo a equipe, onde cada integrante tem uma habilidade específica, e demitindo e contratando funcionários quando julgar necessário. O desempenho do jogador pode ser observado por gráficos. Assim que um projeto é iniciado, pode ser feito um detalhamento extenso do planejamento do projeto com cronograma, orçamento, artefatos e códigos a serem produzidos. Assim que o jogo termina, seja de forma bem sucedida ou não, o jogador recebe uma pontuação, utilizada para gerar um *ranking* e pode ser comparada à de outros jogadores.

2.5 Análise do Estado da Arte

Observando os trabalhos selecionados na seção 2.4, percebe-se que os jogos educativos para ensino de engenharia de *software* se concentram na área de gerenciamento de projetos e na utilização de questionários como mecânica de jogo. Além disso, os jogos se aproximam bastante de uma simulação, apenas tangenciando o âmbito de mecânicas de jogo, e possuem uma jogabilidade bastante linear. O jogo *Age of Computers* se diferencia oferecendo flexibilidade de opções ao ter um sistema de mapa e exploração, e o jogo não-digital DELIVER! usa mais de elementos de jogos, com utilização de cartas com atributos. Evitando a linearidade do gerenciamento de projetos, o eRiskGame utiliza-se de *machine learning* para oferecer eventos pseudo-aleatórios e imprevisibilidade, além de também oferecer mecânicas de "compra e venda"de membros da equipe.

Um ponto importante a se atentar é que nem todos os trabalhos sobre jogos educacionais forneceram imagens do jogo ou formas de acesso ao jogo. No *survey* realizado por Santos et al. (2020) existem *links* que levam a endereços vazios ou então trabalhos que nem disponibilizam uma forma de acesso. No trabalho do jogo DELIVER! um link para acesso ao jogo foi fornecido, mas o endereço não encontrava o jogo. Por outro lado, aqueles que fornecem imagens demonstram que a interface se distancia da interface de um jogo, lembrando bastante ferramentas de suporte ao desenvolvimento de *software* e gerenciamento de projetos.

A partir do exposto, pode-se entender o comentário de Caulfield et al. (2011) sobre o distanciamento dos conceitos abordados no jogo das mecânicas de jogo. Dessa forma, este trabalho teve como objetivo geral utilizar-se das propriedades de simulação de um projeto real, mas aprimorando o lado de mecânicas de jogo, para oferecer certa liberdade de exploração, além de implementar uma interface mais próxima de um jogo digital e outras mecânicas para enfatizar a jogabilidade e o aspecto lúdico.

Para tal, teve-se como objetivos específicos utilizar a estratégia de implementar situações pseudo-aleatórias com uso de agentes inteligentes e lógica *fuzzy* e explorar o sistema de mapas, como visto no jogo *Age of Computers*. Quanto às mecânicas de jogo, buscar a união de jogabilidade e conceitos de gerenciamento de riscos e de projetos, como habilidades de funcionários e recursos de escopo, orçamento e cronograma que podem ser gerenciados para evitar riscos e explorar oportunidades.

As mecânicas de jogo sendo evidenciadas permitem variedade de estilo de jogo e de lidar com os desafios propostos, levando o jogador a explorar, a cada jogada, estratégias diferentes de realizar os projetos, passando assim mais tempo jogando e explorando as mecânicas e, consequentemente, o conteúdo de gerenciamento de riscos.

3 Execução do Trabalho

Este trabalho foi desenvolvido seguindo o modelo ENgAGED, um modelo de desenvolvimento de jogos educacionais. Esta seção foi organizada de modo a seguir o modelo, na estrutura apontada na seção 2.2.1, pelo Quadro 1.

3.1 Análise da Unidade Instrucional (UI)

Esta seção é dividida em três partes: especificação da UI do jogo, caracterização dos aprendizes e definir objetivos de desempenho.

A UI do jogo é o tema de gerenciamento de riscos em projetos de *software*. Além de ser considerado um tema importante, é um tema difícil de pôr em prática em aula, o que pode levar a um distanciamento do aluno para com o conteúdo.

Os aprendizes alvo do jogo desenvolvido são estudantes de ensino superior das áreas de tecnologia, mais especificamente estudantes da disciplina Engenharia de *Software*.

Corrigir para definição da avaliação O jogo visa trazer o gerenciamento de riscos a um olhar mais atento dos alunos, fazer com que estes consigam fixar melhor o conteúdo aprendido em sala de aula, que tenham mais atenção a esse processo tão importante do desenvolvimento de *software* e que consigam visualizar melhor os riscos de projeto.

3.2 Projeto da Unidade Instrucional

3.2.1 Definir avaliação do aluno e conteúdo da estratégia instrucional

O jogo tem como objetivo de avaliação do aluno a recordação e retenção do conteúdo dado em sala de aula. Dessa forma, o jogo é dividido para avaliar pontualmente os conteúdos.

Além disso, são avaliadas habilidades de preparação para projetos, determinando conhecimentos que são necessários para determinados projetos de *software*, de gerenciamento de recursos disponíveis e de adaptação a mudanças e imprevistos.

Para o desenvolvimento do jogo foi decidido adotar uma estratégia dividida em etapas que referenciam as etapas do gerenciamento de riscos, Identificação de Riscos, Análise de Riscos, Planejamento de Riscos e Monitoramento de Riscos, apresentadas por Sommerville no livro Engenharia de *Software* (SOMMERVILLE, 2016).

Cada etapa possui uma forma de avaliar o aluno. A etapa de identificação avalia a identificação de riscos específicos para o projeto escolhido pelo jogador; na etapa Avaliação dos Riscos, os alunos são avaliados quanto a sua capacidade de estimativa de probabilidade e impacto dos riscos, baseado nas descrições dadas de cada risco e do projeto em questão, tendo os valores

esperados baseados em trabalhos publicados; a etapa de planejamento tem a função de avaliar a identificação do tipo de cada risco além de uma forma de preveni-los; por fim a etapa de monitoramento, também chamada etapa de exploração, é a etapa onde o jogador percorre o mapa do jogo e se encontra com os riscos e oportunidades, sendo avaliado quanto a sua tomada de decisão e gerenciamento de recursos, além de mostrar os efeitos das etapas anteriores, indicando quando o planejamento e avaliações dos riscos foram adequadas.

Além disso, na etapa de avaliação, o jogador recebe um *feedback* através de ganho de recursos sobre seu desempenho na avaliação.

A próxima etapa do modelo é decidir sobre o desenvolvimento de um jogo ou uso de um já existente. Como o intuito deste trabalho é desenvolver um jogo com características próprias e identidade única, o jogo foi desenvolvido desde o início.

3.2.2 Revisar o modelo de avaliação do jogo

Nesta etapa é feita a verificação e revisão do modelo de avaliação do jogo. Como padrão, o ENgAGED propõe o modelo MEEGA, então foi decidido utilizar o modelo proposto, porém em sua versão atualizada, o MEEGA+.

O modelo MEEGA+, além de determinar questões a serem feitas aos alunos, também diz para o avaliador desenvolver suas próprias questões, abordando o conteúdo específico do jogo educacional.

3.3 Desenvolvimento do Jogo Educacional

3.3.1 Levantar requisitos do jogo

Quanto ao fluxo de jogo, foi decidido que seria *single player* e, para incorporar o conteúdo da disciplina de gerenciamento de riscos, utilizar recursos conhecidos, como os modelos e conceitos introduzidos na Seção 2.3, servindo como tarefas de preparação para reduzir as probabilidades e impactos dos riscos. Também foi decidido pela divisão em etapas que representassem tanto modelos de gerenciamento de riscos, como modelos de projeto, buscando a integração desses conceitos diretamente com a jogabilidade. A divisão do fluxo geral deve seguir as etapas de gerenciamento de riscos e as etapas de cada mapa devem seguir os modelos de projeto escolhidos, pois além de fazer uma ligação direta a conteúdos aprendidos, ajuda o jogador a se localizar e realizar suas decisões.

Quanto às etapas de preparação, seguindo o modelo de fluxo de gerenciamento de riscos proposto por (SOMMERVILLE, 2016), é preciso uma etapa de identificação de riscos, de avaliação de riscos, de planejamento e monitoramento, para atender tanto à distribuição do conteúdo, quanto a uma progressão de jogo em fases. As três primeiras citadas compõem a preparação, enquanto a fase de monitoramento é a etapa do jogo onde o jogador vai ter maior interação com os elementos dinâmicos do jogo.

É importante que o jogador tenha liberdade de escolha quanto ao tipo de projeto que deseja executar no jogo e quanto as habilidades que deseja ter, compondo uma equipe que vai ajudá-lo durante o jogo e que deve ser pensada para ser efetiva no projeto escolhido.

Seguindo exemplo do jogo *Age of Computers*, detalhado na Seção 2.4, deve ser feita a criação de mapas para permitir maior interação do jogador além de ajudar na visualização e enfatizar uma conexão com modelos que são utilizados nos projetos disponíveis para escolha. Os mapas são presentes na etapa de monitoramento dos riscos e são neles onde o jogador vai se situar na etapa do projeto, além de visualizar sua rota e possíveis decisões de caminhos a seguir.

O jogador deve ter recursos que usará para pagar por prevenções e pelas consequências dos riscos, além de comporem os pontos do jogador ao final do jogo. Esses recursos foram divididos em conceitos de projeto: escopo, orçamento e cronograma.

Quanto ao conteúdo do jogo, deve-se ter uma divisão de riscos e oportunidades, com três classificações de risco: risco de projeto, de produto e de negócios; além de probabilidades e impactos para os riscos e custos e recompensas para as oportunidades. Também é necessário ter prevenções condizentes para os riscos, que deverão ser escolhidas pelo jogador para reduzir a probabilidade e impacto deles. Além disso, é importante que o jogador seja avaliado quanto a decisão de quais riscos deve investir seus recursos para realizar as prevenções, disponibilizando mais opções com custos e consequências diferentes.

3.4 Concepção do Jogo

A concepção do jogo consiste na descrição das principais atividades e características do jogo. Esta etapa, basicamente, consiste na elaboração do *Game Design Document* (GDD), que terá alguns tópicos resumidos nesta seção. O GDD pode ser visualizado na íntegra através do link <https://qrco.de/bcipxC>.

Para a elaboração dos mapas, imagens de fundo e de personagens, foram utilizadas a plataforma Inkarnate ("INKARNATE", 2012) e o *software* Krita (FOUNDATION, 1998).

3.4.1 História e Gameplay

Um grupo de aventureiros vai adentrar uma masmorra para procurar tesouros, mas lá dentro existem muitos monstros e o grupo tem que se preparar para enfrentá-los.

Antes de percorrer a masmorra, o jogador deverá se preparar para os riscos que pode encontrar. Ele deve escolher um tipo de masmorra dentre duas opções, atreladas a dois tipos de projetos: Implantação de um Sistema Enterprise Resource Planning (ERP) e Desenvolvimento de *App*. A preparação é baseada no processo de gerenciamento de riscos, que contém quatro fases, Identificação de Riscos, Análise de Riscos, Planejamento de Riscos e Monitoramento de Riscos. As fases de preparação do jogo foram atreladas às três primeiras do gerenciamento de riscos, enquanto a quarta foi atribuída a parte de exploração.

O jogador possui três recursos, Escopo, Orçamento e Cronograma, e começa com uma quantidade definida dependendo da dificuldade escolhida. Recursos serão ganhos durante as fases de preparação ao identificar, avaliar e planejar riscos corretamente. Durante a fase de exploração, os riscos vão reduzir os recursos e as oportunidades poderão aumentá-los, cobrando um preço.

Nas fases de preparação, o jogador irá identificar os riscos possíveis no projeto específico que foi escolhido, avaliar a grau de probabilidade e impacto do risco utilizando uma matriz de probabilidade/consequência. Então deverá classificar cada risco identificado como um de três tipos, depois selecionar uma prevenção que achar condizente com o risco sendo planejado e por fim escolher uma reação caso o risco ocorra durante a exploração.

Na fase de monitoramento, ou seja, a fase de exploração da masmorra, o jogador terá um mapa por onde pode percorrer determinadas salas e traçar um caminho, dependendo do projeto e da estratégia que ele decidir usar. A cada sala que o jogador entrar, haverá a possibilidade de encontrar um risco. Caso não encontre um risco, ele encontrará uma oportunidade e pode decidir aproveitá-la ou não, observando os recursos que possui. Caso algum recurso chegue a zero, o jogador perde o jogo. O jogo termina quando o jogador atravessa toda a masmorra.

Para a escolha dos riscos utilizados no jogo, foi feita uma pesquisa por diversos trabalhos e artigos que descrevem e classificam riscos de projetos de *software* (LEOPOLDINO, 2004; MACHADO; JÚNIOR; COSTA, 2014). Estes trabalhos apresentaram uma grande variedade de riscos e muitos riscos em comum, além de diferentes classificações para eles. Quanto aos tipos de risco, para manter o jogo mais simples, foram escolhidos os tipos apresentados por (SOMMERVILLE, 2016), sendo: riscos de projeto, riscos de produto e riscos de negócios.

Também foram encontrados trabalhos que apontavam riscos específicos para o projeto de implantação de Sistema ERP (UMA...,), e para o projeto de desenvolvimento de app (OLIVEIRA; GOMES; LIMA, 2014).

Em seguida, foi analisada a probabilidade que cada material base atribuiu aos riscos, não considerando números, mas sim uma escala, adaptada para caber em uma progressão de cinco passos, além dos níveis de impacto dos riscos, também adaptados em uma escala de cinco passos. Chegou-se à seleção final dos riscos, em um total de 44 riscos, 25 compartilhados para os dois projetos, dez específicos para o projeto de Sistema ERP e nove específicos para o Projeto de *App*.

Por fim, será calculada uma pontuação para o jogador a partir de seus recursos restantes e algumas informações são apresentadas em um relatório final (Figura 10), como porcentagem de riscos corretamente prevenidos e oportunidades aproveitadas pelo jogador.

Figura 10 – Relatório Final

Fonte: Elaborado pelo autor

3.4.2 Personagens e Habilidades

Os personagens do jogo são aventureiros que representam membros de uma equipe de desenvolvimento de *software*. Cada personagem conta com uma habilidade única que combate riscos, diminuindo suas probabilidades e diminuindo o impacto caso o risco possa ser resolvido com alguma das habilidades presentes no grupo. Cada personagem também conta com um valor de Moral, que pode diminuir ou aumentar durante a fase de monitoramento/exploração.

As habilidades são: liderança, comunicação, inovação, dedicação, organização, entusiasmo, disciplina, adaptabilidade, conhecimento de tecnologia, conhecimento de mercado, mentalidade ágil e experiência.

A tela de seleção de personagem pode ser vista na Figura 11, com alguns personagens sorteados para o jogador escolher.

Figura 11 – Fase de Seleção da Equipe

Fonte: Elaborado pelo autor

3.4.3 Riscos e Oportunidades

Os riscos compõem o principal sistema do jogo, eles possuem três tipos nos quais serão classificados, possuem valores de impacto para cada recurso para cobrar o jogador caso venha a acontecer, sua probabilidade, a qual servirá para sortear se o risco vai acontecer ou não, e uma probabilidade de reincidência, valor que a probabilidade assume após um risco acontecer. A probabilidade de um risco acontecer pode ser diminuída pelas habilidades da equipe, pelas prevenções feitas corretamente ou completando etapas durante o Monitoramento. Mas a probabilidade também pode aumentar, quando um risco acontece, existem riscos decorrentes que terão sua probabilidade aumentada, ou também caso uma etapa não seja concluída completamente.

Durante a fase de Monitoramento, ou seja, na fase de exploração dos mapas, quando o jogador entra em uma sala, ou seja, clica em uma sala um risco dessa lista será escolhido baseado na dificuldade (Fácil, Normal e Difícil) e será sorteado se vai ou não acontecer.

As oportunidades são riscos especiais, que não são prevenidos, mas podem ser aproveitados quando ocorrem, mas os jogadores devem tomar cuidado, pois algumas delas possuem custos que devem ser considerados. Uma Oportunidade aparece quando um Risco não ocorre e o jogador pode então aproveitar ou recusar a oportunidade caso não deseje gastar seus recursos. Porém, são as oportunidades que dão recursos para que o jogador continue a exploração e para somar pontos.

3.4.4 Interface

A interface do jogo foi elaborada para aproximar os jogos educacionais de uma aparência de jogo de entretenimento e menos uma interface de um programa de treinamento, como foi observado na seção 2.5. Então, unindo-se ao tema de fantasia, foram utilizados *assets* que lembrassem uma estética medieval, como é o padrão estético do tema. Além disso, a disposição dos elementos na interface de cada fase foi inspirada em modelos de identificação e avaliação de riscos reais, como a matriz de probabilidade/consequência para a fase de avaliação (Figura 12) e o modelo *bow-tie* para a fase de planejamento (Figura 13).

Quanto à fase de monitoramento, inspirada no modelo de mapas citado no jogo *Age of Computers*, na seção 2.4, foram elaborados mapas que lembrassem mapas de masmorras utilizados em jogos interpretativos de fantasia, caracterizados e feitos para refletir os modelos de projeto incremental (a Figura 14 mostra a fase de requisitos), utilizado no projeto de implantação de sistema ERP, e o modelo SCRUM (Figura 15), utilizado no projeto Desenvolvimento de App.

Figura 12 – Fase de Avaliação de Riscos

Fonte: Elaborado pelo autor

Figura 13 – Fase de Planejamento de Riscos

Fonte: Elaborado pelo autor

Figura 14 – Mapa do Primeiro Nível da Masmorra do Projeto de Implantação de Sistema ERP

Fonte: Elaborado pelo autor

Figura 15 – Mapa da Masmorra do Projeto Desenvolvimento de App

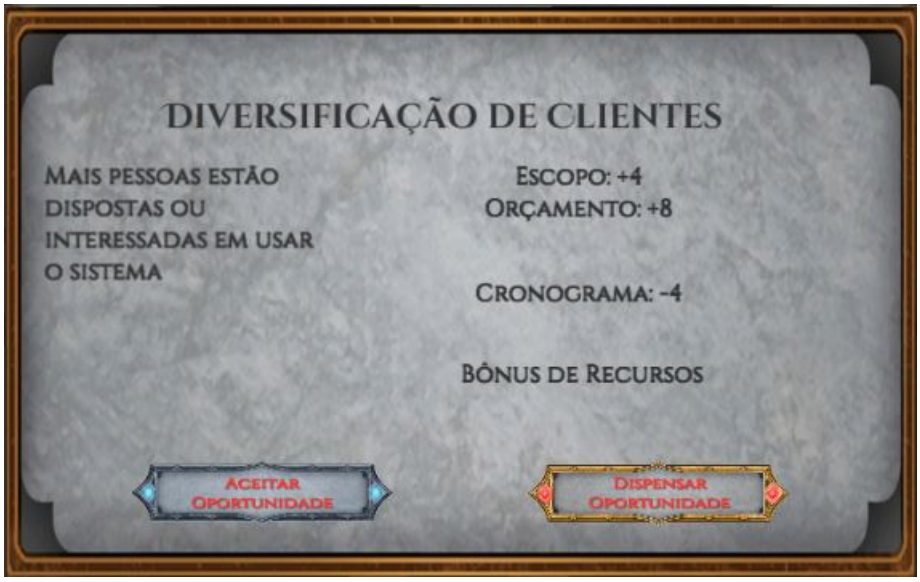
Fonte: Elaborado pelo autor

Para apresentar os riscos e oportunidades durante o jogo foram criadas telas que apresentam suas informações mais importantes, como probabilidade do risco e a ação tomada contra ele, ou os bônus e custos de uma oportunidade. As telas de apresentação de risco e oportunidade na fase de exploração da masmorra podem ser observadas nas Figuras 16 e 17, respectivamente.

Figura 16 – Tela de Apresentação de Risco

Fonte: Elaborado pelo autor

Figura 17 – Tela de Apresentação de Oportunidade



Fonte: Elaborado pelo autor

3.5 Design do Jogo

3.5.1 Definir linguagem de programação ou *game engine*

Como já foi dito anterioremente, a *game engine* escolhida para o desenvolvimento do jogo educacional foi a Unity, que utiliza da linguagem C#.

3.5.2 Produzir ilustrações ou imagens dos elementos do jogo

A ferramenta utilizada para o desenvolvimento das ilustrações e imagens do jogo foi a plataforma Inkarnate.

As ilustrações utilizadas no jogo foram os mapas de masmorra, criados através da plataforma, e os elementos de interface extraídos da plataforma já prontos. As ilustrações extraídas não possuíam o propósito de serem elementos de interface, mas foram reapropriados para tal função no jogo.

As ilustrações podem ser visualizadas no documento do GDD, no Apêndice A.

3.5.3 Modelar o Jogo

Nesta etapa foram definidas a sequência lógica do jogo e a definição dos recursos importantes para o desenvolvimento, como definição de bibliotecas necessárias para gerenciamento de listas, para contas matemáticas, geração pseudo-aleatória de números, além da quantidade de cenas que o jogo deve ter e as condições de passagem de fases.

As cenas são os espaços da *game engine* Unity onde as etapas do jogo são construídas e configuram as fases jogáveis. Elas foram baseadas nas etapas que o jogador deveria realizar durante um gerenciamento de riscos e andamento do projeto, este de acordo com o tipo de projeto que for escolhido. Existe uma cena de início, com o menu principal, e uma de fim, apresentado o relatório final. Além disso, as cenas para cada etapa do jogo: seleção de equipe, identificação, avaliação, planejamento e monitoramento, este sendo variável, possuindo quatro cenas caso o projeto escolhido seja o de desenvolvimento de sistema ERP, ou apenas uma cena, caso o projeto escolhido seja o de desenvolvimento e *app*.

A condição de passagem de fase também muda de acordo com a etapa. Nas etapas de preparação, basta concluir a seleção de personagens ou terminar as tarefas que devem ser feitas com cada risco. Na etapa de monitoramento, onde o jogador pode controlar seu caminho e se movimentar através de um mapa, existem duas condições, uma para cada projeto: para o projeto de sistema ERP, o jogador segue para a próxima cena ao chegar na saída indicada, até que chegue na última saída e vá para a cena de final de jogo; no projeto de desenvolvimento de *app*, o jogador possuí uma única cena, então para que o jogo não termine cedo, é exigido um número mínimo de voltas a serem percorridas pelo mapa, até chegar ao fim indicado, sendo levado até a cena de final de jogo.

3.6 Implementação do Jogo

Primeiramente, foram desenvolvidos os *scripts* para os riscos, oportunidades, habilidades, empregados e prevenções. Também foram criados *scripts* para apresentar esses elementos, com os riscos possuindo 3 formas diferentes de apresentação, uma para a apresentação do nome do risco, uma para a apresentação de múltiplos riscos em uma unidade pequena e um para uma apresentação em detalhes do risco.

Os riscos, oportunidades, prevenções e habilidades foram implementadas como *scriptable objects*, que servem como um modelo, que contém todas as informações necessárias para cada um e podem ser utilizados para criar elementos individuais. Estes elementos individuais serão cada risco, oportunidade, prevenção e habilidade individuais.

Após a criação dos *scriptable objects*, foram criados cada risco, oportunidade, prevenção e habilidades escolhidos para o jogo, dividindo os riscos em riscos gerais, para o projeto de implantação de um sistema ERP e para o projeto de desenvolvimento de app.

Como no jogo será utilizado apenas o mouse para executar as ações, toda a estrutura será feita em *canvas*, ou seja, inteiramente na GUI (*game user interface*. Dessa forma, o jogador não controla um personagem, apenas interage com elementos apresentados pela UI. Com essa característica, todos os objetos do jogo são apresentados na interface.

Acerca da questão da estrutura, é importante notar que existe na fase de exploração da masmorra um espaço físico em que o jogador controla um peão, porém, por conveniência, este espaço também foi implementado em *canvas*.

Os principais objetos do jogo são os referentes aos empregados e aos objetos de apresentação de riscos, oportunidades e prevenções. Outros objetos implementados foram botões para gerenciamento de seleção de riscos, que possuem funções diferentes para cada fase.

Em sequência, foi criado o *Game Manager* (GM), um *script* que irá gerenciar vários dados relativos aos riscos, oportunidades, empregados e prevenções, além de controlar as informações da masmorra escolhida, as trocas de cenas (fases) e reinicialização do jogo e das informações gerais. Com o GM criado, pôde-se guardar em listas sob ele para os riscos, oportunidades e prevenções, além de listas auxiliares para cada, que serão detalhadas mais a frente. Estas listas servirão para manter controle dos dados que devem ser usados para o projeto escolhido e para cada fase ter acesso mais fácil às informações que precisa.

A partir disso, os objetos dos empregados foram criados e armazenados em uma lista no GM, para que na etapa de seleção da equipe, possa ser feito o controle de quais empregados já foram selecionados e quais serão apresentados. Os empregados selecionados pelo jogador são também armazenados em um *script* próprio para o jogador, que gerencia os valores dos recursos que são utilizados no jogo, além dos empregados e suas habilidades.

Para cada uma das etapas Identificação, Avaliação e Planejamento foi criado um *script* que inicializa informações importantes para a fase e prepara os objetos necessários naquela cena. A cena de *feedback* final, que aparece assim que o jogador completa o jogo, também possui um *script* próprio que irá calcular a pontuação e controlar a apresentação das informações. Para a parte de exploração, foram criadas as cenas com os mapas e os objetos das salas foram criados, com um *script* para controlar movimentação. Havendo uma diferença importante no projeto de desenvolvimento de app, que possui um único mapa, foi necessária a criação de um *script* para o mapa, que serve para verificar os *loops* de *sprint*, decisivos para o andamento do jogo no mapa e de condição de final de jogo.

Então foram criados os *scripts* para os objetos de apresentação dos riscos e das prevenções para as etapas de gerenciamento de riscos. Para os riscos, foi necessário três formas diferentes de apresentação, uma que mostra seu número (id) e nome, outra em que um único objeto apresenta mais de um risco, então apenas mostra seu id, e uma última que mostra mais detalhes do risco e que é utilizado quando um risco ocorre durante a fase de exploração. Para as prevenções, apenas um objeto que apresentasse o nome foi o suficiente.

3.6.1 Os Riscos

Os riscos foram implementados como *scriptable objects* para que cada risco contenha as mesmas informações que serão essenciais para o jogo. Essas informações podem ser observadas no Quadro 5.

O valor de *id* é utilizado para identificar nas interfaces de apresentação e principalmente na etapa de avaliação, onde vários riscos devem ser apresentados em uma única interface, então para ter espaço somente o id de cada risco é apresentado.

Quadro 5 – Campos de um Risco

Campo	Descrição
id	Número de identificação do risco
type	Indica o tipo do risco
riskClass	Indica a classe do risco
project	Indica o projeto do risco
riskName	Nome do risco
riskDescription	Descrição do risco
preventions[]	Prevenções que afetam o risco
derivatives[]	Riscos que são decorrentes de um risco quando ocorre
reaction	Reação atribuída ao risco
timesHappened	Quantas vezes o risco aconteceu no jogo
impactLevel	Nível de impacto do risco
probLevel	Nível de probabilidade do risco
evaluatedImpact	Nível de impacto atribuído ao risco pelo jogador
evaluatedProb	Nível de probabilidade atribuído ao risco pelo jogador
baseProbability	Probabilidade inicial do risco
probability	Probabilidade atual do risco
reincidence	Probabilidade que o risco assume após ocorrer pela primeira vez
scopeCost	Custo em escopo
moneyCost	Custo em orçamento
timeCost	Custo em cronograma
VH	Salva o valor fuzzy para probabilidade Muito Alta
H	Salva o valor fuzzy para probabilidade Alta
M	Salva o valor fuzzy para probabilidade Moderada
L	Salva o valor fuzzy para probabilidade Baixa
VL	Salva o valor fuzzy para probabilidade Muito Baixa

Fonte: Elaborado pelo autor

O valor de *type* indica se o risco é de projeto, produto ou negócios. Enquanto *riskClass* é uma *string* com uma classe de risco, utilizada quando o agente de riscos faz a seleção dos riscos que podem ocorrer na etapa do projeto.

Os campos *riskName* e *riskDescription* são utilizados na apresentação dos riscos, sendo este último utilizado apenas na apresentação do risco durante a etapa de exploração.

Os dois campos *preventions* e [derivatives] são listas que indicam, respectivamente, quais prevenções combatem o risco e quais riscos terão a probabilidade aumentada caso o risco ocorra.

O campo *reaction* marca a reação que o jogador atribuiu ao risco na etapa de planejamento, sendo verificado quando um risco ocorre para realizar as penalidades de recursos e as reduções dessas penalidades.

O campo *timesHappened* guarda a contagem de quantas vezes o risco ocorreu durante a fase de exploração e é usado para reduzir a probabilidade do risco caso ele esteja ocorrendo muitas vezes.

Os campos *impactLevel* e *probLevel* são inteiros que vão de 1 a 5, indicando o nível de

impacto e probabilidade do risco, valores usados para verificar a avaliação que o jogador fará dos riscos na matriz de probabilidade/consequência. E os campos *evaluatedImpact* e *evaluatedProb* salvam o nível de impacto e probabilidade que o jogador avaliou o risco na matriz de probabilidade/consequência e são mostrados na fase de planejamento para ajudar o jogador a escolher qual reação atribuir ao risco.

O campo *probability* é o valor que será constantemente alterado e verificado durante o jogo. A probabilidade começa com um valor base, *baseProbability* que é reiniciado sempre que o jogo recomeça ou o jogador volta para o menu principal, e, sempre que o jogador entrar em uma sala, essa probabilidade é fuzzyficada e os valores de cada nível, muito alta, alta, moderada, baixa e muito baixa, são armazenados nos campos *VH, H, M, L* e *VL*, respectivamente. Estes valores são utilizados para definir que risco será sorteado, dependendo da dificuldade escolhida, por exemplo, na dificuldade Difícil, os riscos com maiores graus de pertencimento no campo *VH* e *H*, ou seja, com maior grau de pertencimento à probabilidade muito alta e alta, são os riscos que serão sorteados.

Quando um risco é ativado, ele irá verificar as habilidades da equipe do jogador e aplicar as devidas reduções de impacto baseado no poder de combate do jogador, o qual também pode ser alterado por outras habilidades. Além de verificar as habilidades, ao ser ativado, os riscos também verificam as ações feitas na etapa de planejamento, ou seja, se a prevenção tiver sido corretamente escolhida para o risco e a reação mitigar foi escolhida, então o impacto do risco é reduzido, ou se a ação atribuir tiver sido escolhida, apenas o impacto no orçamento, com maior penalidade, é aplicado.

3.6.2 As Oportunidades

As oportunidades foram também implementadas como *scriptable objects* e as informações essenciais para cada oportunidade podem ser vistas no Quadro 6.

Quadro 6 – Campos de uma Oportunidade

Campo	Descrição
opportunityName	Nome da oportunidade
opportunityEffect	Descrição do efeito da oportunidade
opportunityDescription	Descrição textual da oportunidade
repeatable	Indica se a oportunidade pode ocorrer mais de uma vez
baseScopeBonus	Valor base de bônus de escopo
baseMoneyBonus	Valor base de bônus de orçamento
baseTimeBonus	Valor base de bônus de cronograma
scopeBonus	Valor atual do bônus de escopo
moneyBonus	Valor atual do bônus de orçamento
timeBonus	Valor atual do bônus de cronograma
prevention	Prevenção que a oportunidade aplica
makePrevention	Indica se a oportunidade realiza alguma previsão
increasePower	Indica se o risco aumento o poder de combate a riscos
decreaseCosts	Indica se a oportunidade diminui os custos das oportunidades
diversify	Indica se a oportunidade é a Diversificação de Processos

Fonte: Elaborado pelo autor

Os campos *opportunityName*, *opportunityEffect* e *opportunityDescription* são campos de descrição da oportunidade e são utilizados na tela de apresentação de oportunidade quando uma oportunidade aparece durante a fase de exploração.

O campo *repeatable* é um campo booleano que indica se aquela oportunidade pode aparecer mais de uma vez.

Os campos relativos a escopo, orçamento e cronograma são divididos em base e atuais. Os campos base são utilizados para reiniciar os valores quando o jogo recomeça ou o jogador volta ao menu principal, enquanto os atuais são aqueles que podem ser modificados por outras oportunidades ou habilidades e que são utilizadas para dar o bônus ao jogador e para ser o custo da oportunidade.

O campo *prevention* recebe uma prevenção que a oportunidade pode realizar quando o jogador a aproveita. Este campo pode ser vazio, já que nem todas as oportunidades realizam prevenções.

Em seguida, têm-se os campos booleanos *makePrevention, increasePower, decreaseCosts, diversify*, que indicam diferentes efeitos que uma oportunidade pode realizar. Respectivamente: a oportunidade realiza uma prevenção, a oportunidade aumenta o poder de combate da equipe, a oportunidade diminui os custos das oportunidades, a oportunidade é Diversificação de Processos, que abre uma escolha para 3 opções de efeitos bônus para o jogador escolher.

Uma oportunidade aparece em jogo quando, ao entrar em uma sala, nenhum risco é ativado. A oportunidade é apresentada com uma descrição e com seus bônus e custos para que o jogador decida se será proveitoso investir naquela oportunidade. O jogador pode escolher recusar-lá, não

recebendo nenhum bônus e não pagando nenhum custo. Caso o jogador a aceite, ela aplica os bônus e custos e, caso possua algum efeito adicional, irá aplicá-lo.

As oportunidades com o campo *makePrevention* marcados como verdadeiro possuem o efeito adicional de reduzir a probabilidades dos riscos afetados pela prevenção no campo *prevention*. As oportunidades com o campo *increasePower* verdadeiros possuem o efeito de aumentar o poder de combate do jogador. As oportunidades com o campo *decreaseCosts* indicado como verdadeiro reduz os custos das oportunidades e as oportunidades com o campo *diversify* verdadeiro, ao serem aceitas, abrem uma extensão da interface de apresentação de oportunidade com 3 opções extras de bônus para serem escolhidas.

3.6.3 As Prevenções

As prevenções também foram implementadas como *scriptable objects*, porém são bem mais simples e contém menos informações, implementadas assim a fim de facilitar a organização do projeto no editor da Unity.

Elas possuem apenas um campo para nome e outro para descrição, sendo guardadas em uma lista no *Game Manager*.

3.6.4 As Habilidades

As habilidades, assim como os *scripts* anteriores, foram implementadas como *scriptable objects*, armazenando poucas informações, como o nome da habilidade, uma descrição e uma lista com os riscos que ela combate.

Uma estratégia diferente foi usada com as habilidades para realizar alguns de seus efeitos que não sejam diretamente combater riscos específicos. Com essas habilidades não diretamente combativas, seus efeitos são indicados em campos booleanos do *script* do jogador e então consultados nas etapas apropriadas do jogo.

3.6.5 O Jogador

O *script* gerenciador do jogador foi implementado usando o *singleton pattern*, um padrão onde se cria uma instância estática única que será usada para acessar os campos públicos de qualquer lugar, sem precisar tornar os campos estáticos, além de ser persistente entre as cenas do jogo. O gerenciador do jogador é responsável por gerenciar os recursos que o jogador possui, além de armazenar a equipe selecionada e outras informações sobre quantidade de riscos e oportunidades que ocorreram durante o jogo. Além disso, ele também contabiliza os riscos prevenidos corretamente e campos booleanos indicam quais as habilidades sua equipe. Os campos do gerenciador do jogador podem ser vistos no Quadro 7.

Quadro 7 – Campos do Gerenciador do Jogador

Campo	Descrição
playerInstance	Instância estática do jogador
scope	Valor de escopo atual
money	Valor de orçamento atual
time	Valor de cronograma atual
points	Pontos
team[]	Lista dos membros da equipe escolhidos
icon	Ícone do jogador
combatPower	Valor da redução do impacto dos riscos
entusiasm	Indica se a equipe possuí a habilidade Entusiasmo
disciplin	Indica se a equipe possuí a habilidade Disciplina
organized	Indica se a equipe possuí a habilidade Organização
PreventCorrect	Quantidade de riscos prevenidos corretamente
risksActivated	Quantidade de riscos que aconteceram durante o jogo
opportunitiesTaken	Quantidade de oportunidades aproveitadas durante o jogo

Fonte: Elaborado pelo autor

3.6.6 O Game Manager

O *game manager*, ou gerenciador de jogo, foi implementado também com o *singleton pattern*, já que guardaria informações importantes para vários *scripts* e de forma persistente, não sendo destruído quando o jogo trocar de cena. Ele guarda as listas de riscos, oportunidades, prevenções, habilidades, empregados, além de listas auxiliares para coordenar chamadas aleatórias e remoção de informações durante o jogo, sem perder informações importantes. Os campos do *game manager* são mostrados no Quadro 8.

Os campos *project* e *difficulty* indicam qual o projeto e dificuldade escolhidos no início do jogo.

As listas *allRisks, project1Risks* e *Project2Risks* são as listas que guardam, respectivamente, todos os riscos do jogo, os riscos usados no projeto de implementação de sistema ERP e os riscos do projeto de desenvolvimento de app. A primeira lista é usada para manter controle de todos os riscos, sendo acessada para obtê-los para a decisão dos riscos que podem ocorrer em cada etapa do jogo. Já as outras duas listas, são utilizadas para a fase de identificação, que instancia interfaces de apresentação de riscos apenas para os riscos de cada projeto.

A lista *risks* é utilizada como uma referência para outros *scripts* que guardará os riscos do projeto escolhido, e a lista *risksAux* é utilizada para controlar os riscos que ainda não foram avaliados na fase de avaliação e sua condição de vazia é a condição de finalização dessa fase.

As listas *risksIdentified* e *risksAssigned* guardam, respectivamente, os riscos que foram identificados pelo jogador na fase de Identificação e os riscos aos quais o jogador selecionou a reação Atribuir na etapa de Planejamento. A primeira lista é chamada nas fases subsequentes

para serem realizadas a avaliação e planejamento, enquanto a segunda é consultada sempre que um risco ocorre.

Quadro 8 – Campos do *Game Manager*

Campo	Descrição
project	Projeto escolhido pelo jogador
difficulty	Dificuldade escolhida pelo jogador
allRisks[]	Lista com todos os riscos
project1Risks[]	Lista com os riscos do projeto 1
project2Risks[]	Lista com os riscos do projeto 2
risks[]	Lista dos riscos que estão em jogo
risksAux[]	Lista auxiliar de riscos em jogo
risksIdentified[]	Lista dos riscos identificados pelo jogador
risksAssigned[]	Lista dos riscos com a reação Atribuir
preventionsList[]	Lista com todas as prevenções
preventionsMade[]	Lista com as prevenções planejadas corretamente pelo jogador
opportunitiesList[]	Lista das oportunidades
opportunitiesAux[]	Lista auxiliar das oportunidades
employeesList[]	Lista dos empregados
employeesAux[]	Lista auxiliar dos empregados
currentScene	Cena atual do jogo
nextScene	Próxima cena a ser carregada
currentRoom	Sala em que o jogador se encontra
risksInSequence	Quantidade de riscos que aconteceram em sequência

Fonte: Elaborado pelo autor

As listas *preventionsList* e *preventions* são listas idênticas em conteúdo, porém a segunda é uma instância estática da primeira, usada para ser acessada por outros *scripts* para consultar as prevenções. O mesmo procedimento das listas das prevenções foi utilizado com as listas *opportunitiesList* e *opportunities*, e com *employeesList* e *employees*.

Outra função também importante do *game manager* é controlar as trocas de cenas e ações que acontecem quando uma nova cena é carregada. Esse controle é feito através dos campos *currentScene* e *nextScene*, atualizados sempre que uma nova cena é carregada.

3.6.7 As Salas

As salas são as peças centrais na fase de exploração, o jogador se desloca pelos mapas através delas e é nelas que encontram os riscos e as oportunidades. Elas, por si só, são relativamente simples, controlando o movimento do peão do jogador e do custo de deslocamento, juntamente com quais as próximas salas às quais o jogador pode se mover, porém também ativam o evento *OnEnterRoom*, o qual está sendo escutado por agentes que vão realizar suas ações assim que o jogador entra em uma sala.

O *script* das salas também toma o cuidado de verificar os *loops* dos *sprints* no projeto de desenvolvimento de app, para que o jogo não termine muito cedo e o jogador possa dar um mínimo de 4 *loops* e 2 *sprints* antes de finalizar o jogo.

3.6.8 Os Agentes

Os agentes são peças fundamentais para o funcionamento do jogo, controlando os possíveis riscos, gerenciando suas probabilidades conforme as etapas que completam e gerenciando a moral da equipe.

Eles foram implementados seguindo o modelo descrito na seção 2.2.2, funcionando a partir de gatilhos de eventos de outros *scripts* e trocando informações entre si quando necessário, sendo que nenhum de seus campos seja publicamente acessado. Também foram implementados como persistentes entre as cenas, atrelados um objeto na cena apelidado de Beholder, que não será destruído quando houver troca de cenas.

Também nos agentes é onde é aplicada a lógica *fuzzy*, descrita na seção 2.2.3. Essa lógica foi utilizada com a moral da equipe e com as probabilidades dos riscos possíveis, sendo escolhida uma forma de deffuzificação diferente para cada.

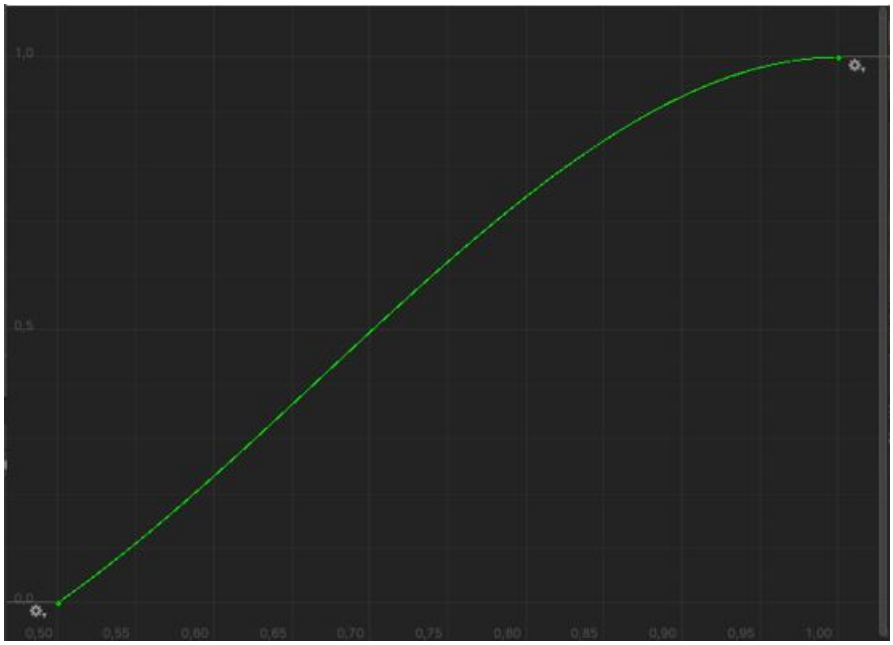
3.6.8.1 Team Agent

O agente da equipe é responsável por ver o estado da moral dos personagens da equipe e perceber o que fazer a seguir. Suas ações são engatilhadas assim que o jogador encontra um risco e a equipe tem que enfrentá-lo. A equipe naturalmente calcula sua própria moral baseada em como enfrentou o risco, então o *team agent* têm seu funcionamento engatilhado para quando esse calculo termina.

Para fazer um calculo mais natural entre os 5 personagens que compõem a equipe do jogador, a lógica *fuzzy* foi utilizada. Assim, obteve-se um cálculo mais abrangente e dinâmico sobre o nível de moral geral a partir de múltiplas morais diferentes.

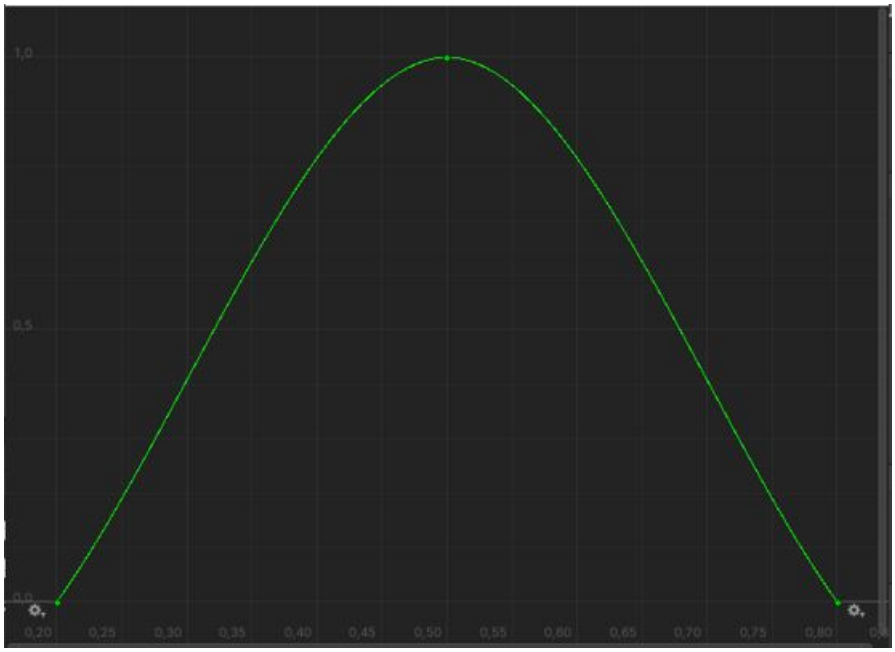
O primeiro passo do *team agente* é ver a moral de cada empregado e fazer a fuzzificação do valor em 3 conjuntos *fuzzy*, Alta, Normal, Baixa. Cada conjunto foi implementado usando *animation curves*, recursos nativos da Unity, e as curvas foram feitas à mão no editor (Figura 18, 19 e 20). Os valores limites para os graus de pertencimento da curva de moral alta são de 0 quando a moral é 0.5 ou menor e 1 quando a moral é 1. Para a curva de moral normal os valores são 0 quando a moral é 0.2 ou menor e quando é 0.8 ou maior e 1 quando a moral for 0.5. Por fim, para a curva de moral baixa, o grau de pertencimento é 0 quando a moral for igual ou maior a 0.5, e 1 quando for igual a 0.

Figura 18 – Curva do Conjunto Fuzzy de Moral Alta



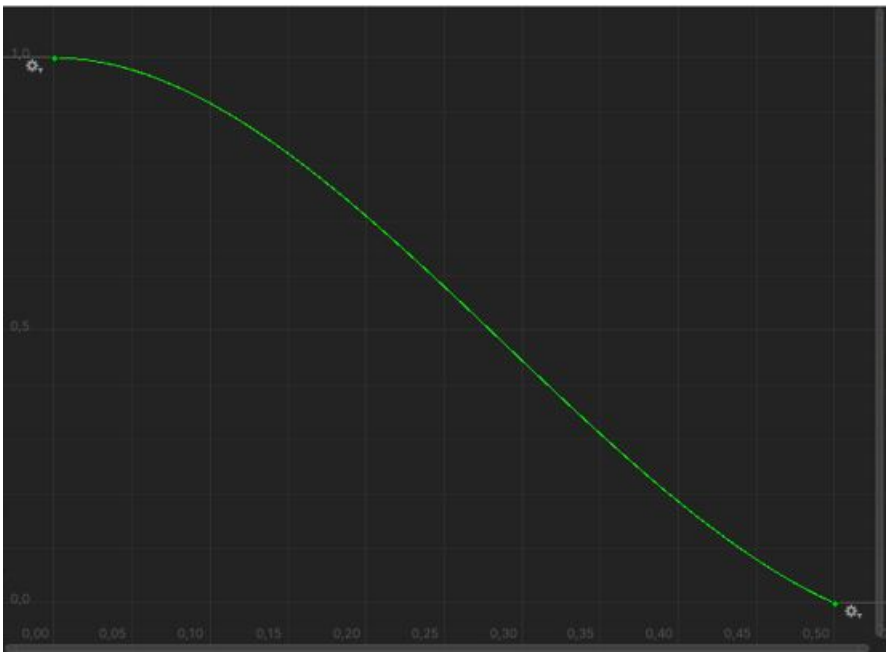
Fonte: Elaborado pelo autor

Figura 19 – Curva do Conjunto Fuzzy de Moral Normal



Fonte: Elaborado pelo autor

Figura 20 – Curva do Conjunto Fuzzy de Moral Baixa



Fonte: Elaborado pelo autor

Cada membro da equipe terá um valor para cada grau e esses valores são guardados em uma lista para cada conjunto *fuzzy*, *highMorale*, *normalMorale* e *lowMorale*. Após os 5 personagens terem sua moral fuzzyficada, são agregados pelas regras fuzzy, onde se escolhe o menor valor para cada conjunto, ou seja, é realizada uma operação "*e*"nos conjuntos, assim determinando se todos os personagens estão com a moral alta *hM*, baixa *lM* ou normal *nM*.

Em seguida, esses valores das regras são utilizados na etapa de deffuzyficação, onde é feita a média:

$$teamMorale = \frac{\sum_0^5 highMorale \times hM + \sum_0^5 normalMorale \times nM + \sum_0^5 lowMorale \times lM}{hM \times 5 + nM \times 5 + lM \times 5} \quad (3.1)$$

Dessa forma obtemos a moral do time e o *team agent* verifica a qual conjunto a moral obtida pertence, classificando em um dos 3 estados de moral. Após a classificação, o agente irá perceber se houve mudança de estado da moral, realizando sua ação em caso positivo. A ação que ele executa é simples, aumentar a probabilidade de riscos da classe de risco "Equipe"caso a moral da equipe esteja baixa, ou diminuir caso seja alta.

3.6.8.2 Rooms Agent

O agente das salas funciona de forma diferente do agente da equipe, não utilizando lógica *fuzzy*, apenas a informação de qual sala o jogador se encontra e quais salas ele já esteve. Essas informações são utilizadas para determinar se o jogador visitou todas as salas da etapa anterior à qual ele se encontra, ou seja, se ele completou a etapa anterior.

O *rooms agent* age assim que o jogador entra em uma sala, vendo a sala atual e então percebendo se ele mudou de etapa. Caso tenha mudado de etapa, a ação do agente é percorrer as salas da etapa anterior e verificar se alguma delas não foi visitada. Em seguida ele vai engatilhar a ativação do *dungeon agent*, o responsável por tomar as ações caso a etapa anterior foi ou não totalmente percorrida.

3.6.8.3 Dungeon Agent

O agente da masmorra é o agente central para controle dos riscos. Ele vê quais etapas já foram completadas e qual a etapa atual que o jogador se encontra e percebe quais riscos são os mais cabíveis para cada etapa, também levendo em conta a classe de cada risco.

Assim que ele decide quais são os riscos possíveis para a etapa atual, ele vai reajustar as probabilidades dos riscos, baseando-se em quais etapas foram totalmente percorridas utilizando um sistema de *flags* booleanas, sendo sempre atualizadas quando o jogador completa uma fase, informação fornecida pelo *rooms agent*.

Além disso, ele realiza um procedimento um pouco diferente para o projeto 2, desenvolvimento de app. Neste projeto, ele não só verifica as etapas percorridas, mas também a quantidade de *loops* e *sprints* completos, reduzindo mais as probabilidades quanto mais ele percorrer a masmorra.

3.6.8.4 Risks Agent

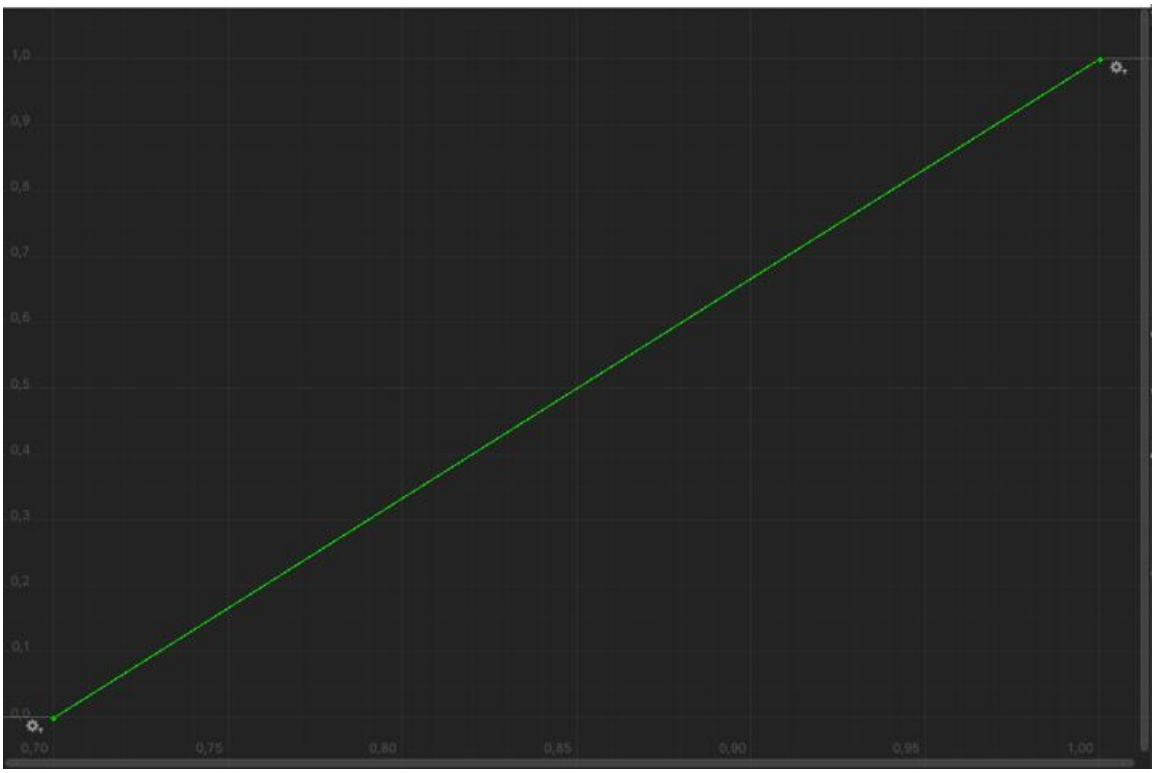
O agente dos riscos é responsável pelo sorteio de riscos entre aqueles determinados como possíveis pelo *dungeon agent*. Ele vai passar as probabilidades de cada risco possível por uma etapa de fuzzyficação com 5 conjuntos *fuzzy* para probabilidade, Muito Alta, Alta, Média, Baixa, Muito Baixa (Figuras 21, 22, 23, 24, 25, respectivamente).

Os valores limites para cada curva foram baseados na escala de probabilidade [0.1, 0.3, 0.5, 0.7, 0.9]. O grau de pertencimento de um conjunto é sempre 0 quando o grau de outro conjunto é 1.

O *risks agent* vê todos os riscos possíveis determinados pelo *dungeon agent* e vai fuzzyficar a probabilidade de cada um deles, obtendo o grau de pertencimento para cada um dos 5 conjuntos anteriormente estabelecidos. Em seguida, vão ser escolhidos 3 riscos aleatoriamente da lista dos riscos possíveis e o agente vai então escolher um deles para ser o risco sorteado. A escolha do risco depende da dificuldade escolhida pelo jogador, se a dificuldade for fácil, o risco com maior grau de pertencimento ao conjunto de probabilidade Baixa será o escolhido, enquanto para a dificuldade normal será escolhido o que tiver maior grau de pertencimento do conjunto de probabilidade Alta ou Normal, e por fim para a dificuldade difícil, o risco com maior grau de pertencimento ao conjunto de probabilidade Muito Alta. Os valores que definem a escolha do risco passam por uma regra *clipped*, considerando os valores fuzzyficados anteriormente.

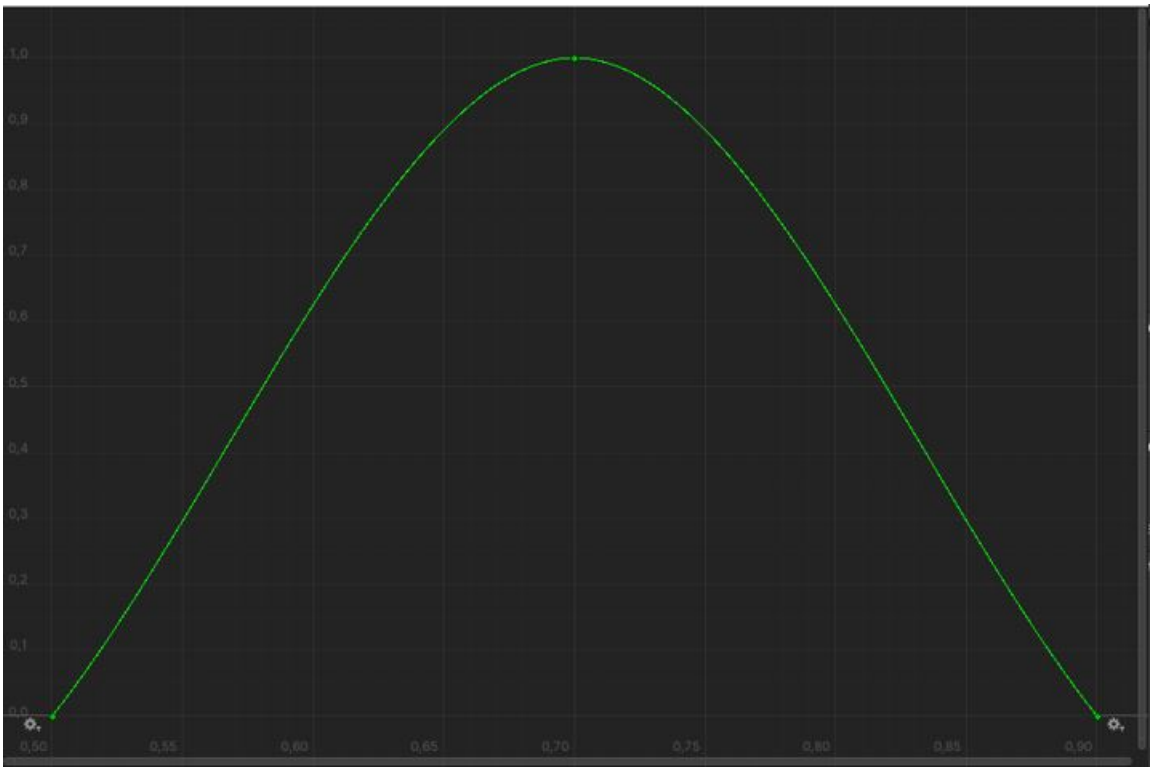
Com o risco determinado, a ação do agente é sortear se o risco irá ocorrer ou não. Um número aleatório, obtido pela função *Random.Range()*, da biblioteca *UnityEngine*, com intervalo *float* de 0 a 1. Se o valor aleatório for inferior ao valor da probabilidade do risco sorteado, o risco acontece, caso contrário, o agente irá sortear aleatoriamente uma oportunidade.

Figura 21 – Curva do Conjunto Fuzzy de Probabilidade Muito Alta



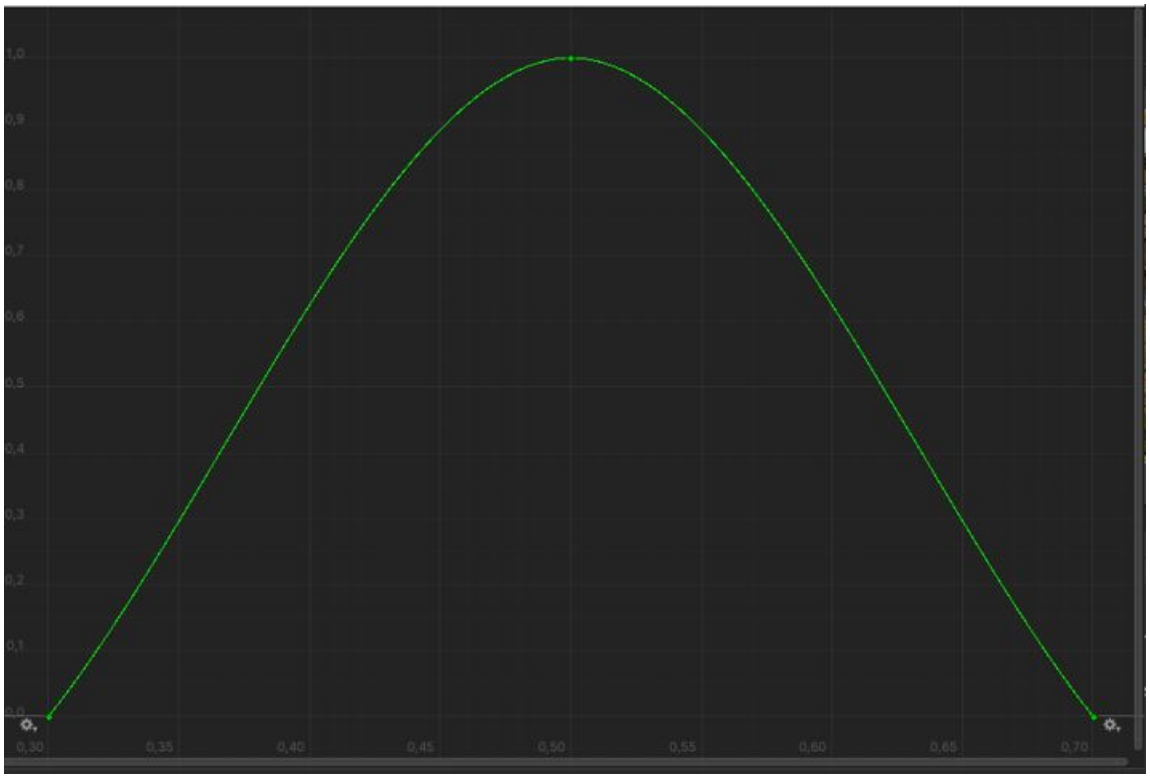
Fonte: Elaborado pelo autor

Figura 22 – Curva do Conjunto Fuzzy de Probabilidade Alta



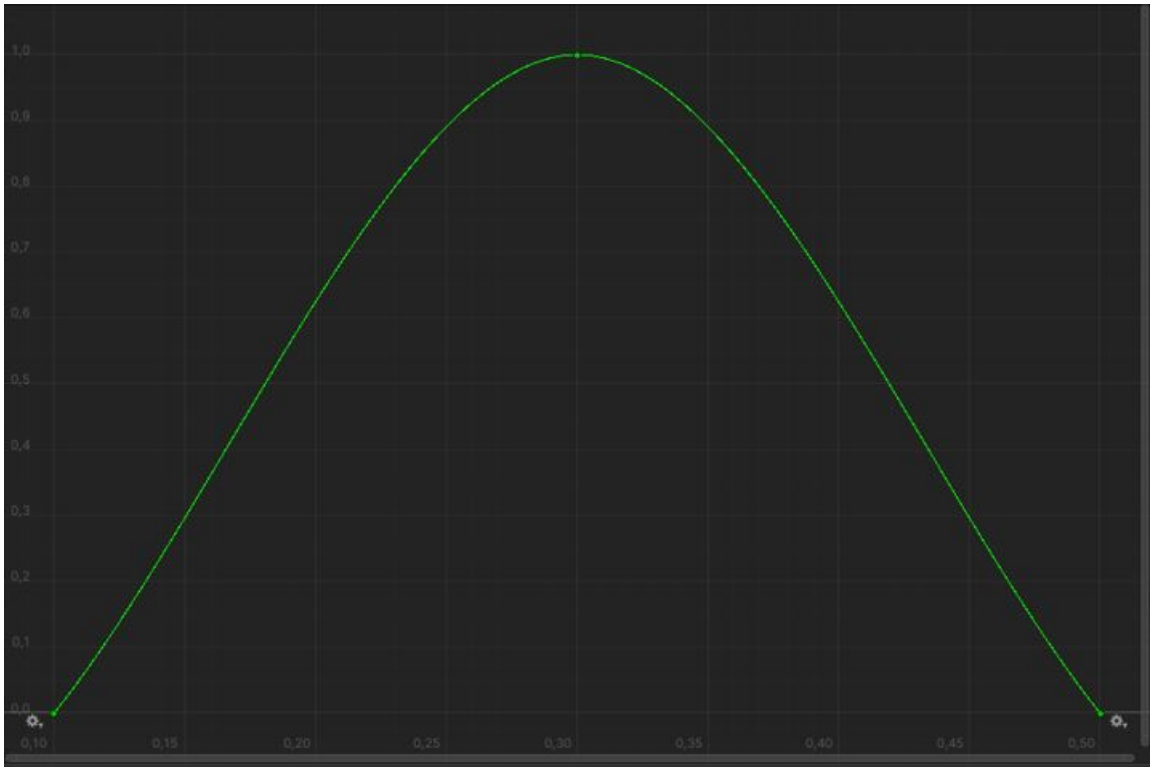
Fonte: Elaborado pelo autor

Figura 23 – Curva do Conjunto Fuzzy de Probabilidade Média



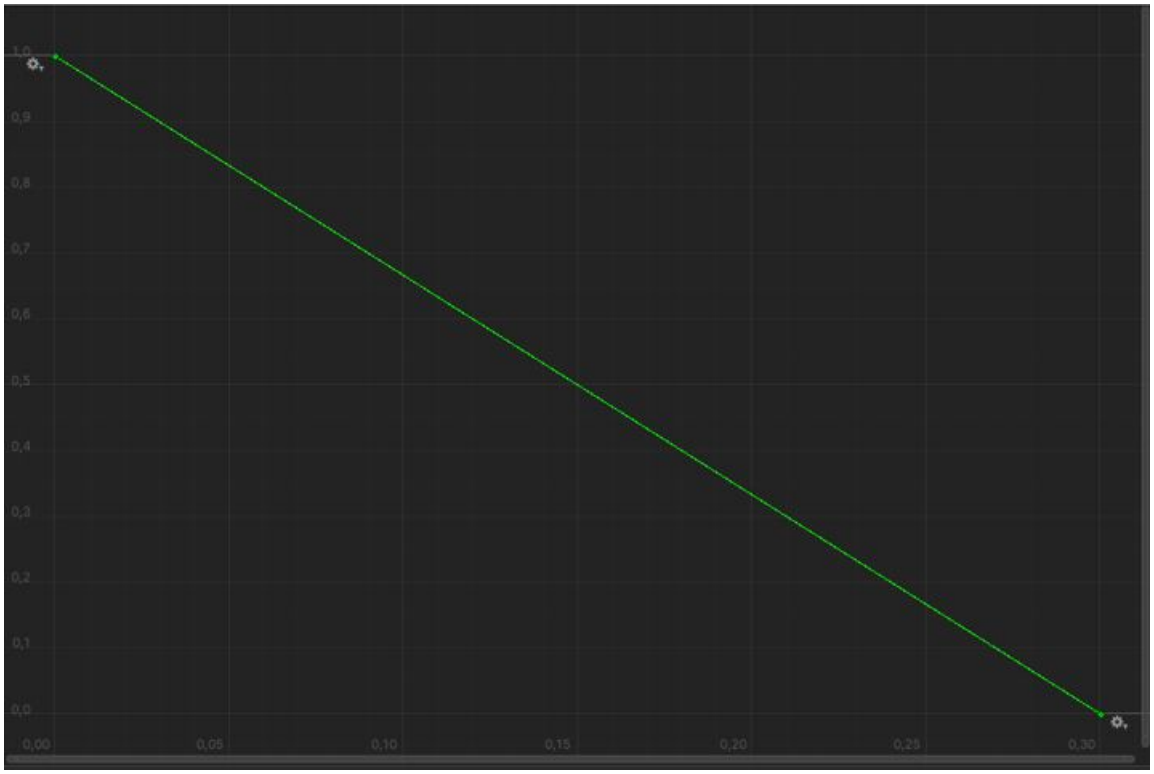
Fonte: Elaborado pelo autor

Figura 24 – Curva do Conjunto Fuzzy de Probabilidade Baixa



Fonte: Elaborado pelo autor

Figura 25 – Curva do Conjunto Fuzzy de Probabilidade Muito Baixa



Fonte: Elaborado pelo autor

4 Execução e Avaliação da Unidade Instrucional

A execução e avaliação foram planejadas para serem feitas através da internet, disponibilizando o jogo no site itch.io, um site que agrega jogos independentes e recursos para serem usados em jogos. O acesso é gratuito e pode ser feito através do endereço <zenndahl.itch.io>. O jogo pode ser baixado para sistemas Linux, Windows e Mac.

O link para o jogo foi distribuído tanto para professores, quanto para alunos do curso de Bacharelado em Ciência da Computação da UNESP do campus de São José do Rio Preto, com um formulário digital, feito através do Google Formulários, contendo avaliações distintas para professores e para alunos.

Para os professores, foi feita uma avaliação quanto a verossimilhança do conteúdo do jogo com a realidade e com como os conteúdos são apresentados em sala de aula. Para os alunos, a avaliação seguiu o modelo MEEGA+, retirando-se as questões envolvendo multijogadores e adicionando avaliações sobre o conteúdo do jogo, como sobre a sensação do impacto e probabilidades dos riscos e o quanto próximos da realidade estariam na opinião deles.

4.1 Resultados

Após três semanas, não foi obtida uma quantia de avaliações satisfatória para gerar dados estatísticos para sustentar uma avaliação concreta do desempenho do jogo como material de apoio ao tema alvo, gerenciamento de riscos.

Porém, resultados preliminares destacam alguns pontos de interesse, como partes que podem melhorar, outras que falta clareza e outras que atingiram seu objetivo. Os principais pontos de destaque que foram bem avaliados foram a interface e o conteúdo do jogo. Quanto a interface, o objetivo era ser agradável e mais próxima do lúdico, utilizando elementos de fantasia, por exemplo. Isso foi bem recebido nos resultados preliminares.

É interessante dar atenção no questionamento se o aluno preferiria aprender com o jogo ao invés de métodos tradicionais. As pesquisas relatadas na seção 2 indicaram que os jogos educativos se encaixam mais como material de suporte que como substitutos e os resultados preliminares corroboram com tais apontamentos.

4.2 Trabalhos Futuros

Aqui são deixadas sugestões de possíveis mecânicas a serem implementadas futuramente para dar maior profundidade ao jogo.

Na etapa de identificação de riscos, foi utilizado um formato bem simples e direto para a seleção dos riscos, porém é possível dar maior profundidade a essa etapa adicionando um esquema de árvore para identificar riscos e seus derivados, recompensando o jogador caso acerte derivados em sequência.

Na seção 2.3.1 foram apresentadas diferentes formas de realizar o gerenciamento de riscos. Pode-se também implementar qualquer daqueles modelos como modelos opcionais, fazendo com que o próprio jogador escolha a forma que prefere realizar o gerenciamento e também diversificando e ampliando o conteúdo do jogo.

Uma última sugestão para futura implementação é a elaboração de mais opções de projeto, com riscos e oportunidades específicos, ou novos modelos de desenvolvimento, podendo também implementar de forma que o jogador possa combinar projetos com os modelos, criando experiências únicas para cada um, também refletindo em maior engajamento e ampliando o conteúdo do jogo.

5 Conclusão

O jogo desenvolvido conseguiu implementar com sucesso os requisitos levantados, criando uma experiência de conteúdo integrada com jogabilidade e mecânicas, aproximando o lúdico e o aprender. Elementos de conceitos e estratégias abordados na seção 2.3 foram inseridos, como a matriz de probabilidade/consequência e o modelo *bow-tie*, além de permitir espaço para diferentes recursos serem implementados.

Com poucas avaliações, não é possível tirar um resultado conclusivo sobre o desempenho do jogo, mas ele se mostra promissor quanto a apresentação e reafirmação de conteúdos estudados e com algumas mudanças pode ser um material de apoio flexível e atrativo, principalmente pelo visual mais lúdico, que ajuda a separar o jogo do ambiente de sala de aula.

Com ajustes em relação à claridade de tarefas e ações, *feedback* e em relação à diversificação de conteúdo, o jogo tem potencial para trazer mais atenção e familiaridade com o tema de gerenciamento de riscos. Adicionalmente, o formato do jogo é um tópico que pode ser ainda melhorado quanto a proximidade da realidade e de como é apresentado na sala de aula.

Assim pode-se dizer que o objetivo do trabalho, desenvolver um jogo educativo sobre gerenciamento de riscos, enfatizando o aprendizado lúdico e integrando os conceitos a serem ensinados com mecânicas de jogo, foi atingido. Todavia, o mesmo carece de uma ampla validação por seu público-alvo, o que não foi possível de ser obtido na presente proposta. Sendo, portanto, uma indicação de trabalho futuro.

Sendo um formato simples e que permite flexibilidade para encaixar diferentes conteúdos que podem dar tanto maior abrangência, quanto profundidade ao tema, além de poder realizar uma comunicação com outros temas de engenharia de *software* ou outros modelos de gerenciamento de riscos, o jogo desenvolvido fornece um potencial material interdisciplinar.

Referências

ANASTASIADIS, T.; LAMPROPOULOS, G.; SIAKAS, K. Digital game-based learning and serious games in education. *International Journal of Advances in Scientific Research and Engineering (ijasre)*, v. 4, n. 12, p. 139–144, 2018.

ANDERSON, L. W.; KRATHWOHL, D. R. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. [S.l.]: Longman,, 2001.

AUDI, G. Emergência e incerteza em jogos de videogame. *Novos Olhares*, p. 72–86, 2014.

BACKLUND, P.; HENDRIX, M. Educational games-are they worth the effort? a literature survey of the effectiveness of serious games. In: IEEE. *2013 5th international conference on games and virtual worlds for serious applications (VS-GAMES)*. [S.l.], 2013. p. 1–8.

BATTISTELLA, P. E.; WANGENHEIM, C. G. von. Engaged: Um processo de desenvolvimento de jogos para ensinar computaçao. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2016. v. 27, n. 1, p. 380.

BLOOM, B. S.; KRATHWOHL, D. R.; MASIA, B. B. Bloom taxonomy of educational objectives. In: *Allyn and Bacon*. [S.l.]: Pearson Education, 1984.

CAMPANHA, C.; CAMPOS, A. P. S. de. Panorama do uso de games, serious games e gamificação na educação. *REVISTA PLURI*, v. 1, n. 2, p. 27–45, 2019.

CARD, A. J.; WARD, J. R.; CLARKSON, P. J. Beyond fmea: The structured what-if technique (swift). *Journal of Healthcare Risk Management*, Wiley Online Library, v. 31, n. 4, p. 23–29, 2012.

CARDOSO, C. Z. Serious games como estratégia motivacional para adesão ao tratamento em saúde mental: revisão sistemática. 2019.

CAULFIELD, C.; XIA, J. C.; VEAL, D.; MAJ, S. A systematic survey of games used for software engineering education. *Modern Applied Science*, Canadian Centre of Science and Education, v. 5, n. 6, p. 28–43, 2011.

DALKEY, N.; HELMER, O. An experimental application of the delphi method to the use of experts. *Management science*, INFORMS, v. 9, n. 3, p. 458–467, 1963.

D'ÁVILA, C. M. Didática lúdica: saberes pedagógicos e ludicidade no contexto da educação superior. *Revista Entreideias: educação, cultura e sociedade*, v. 3, n. 2, 2014.

DEMASI, P.; CRUZ, A. Modelagem fuzzy para um jogo de naves espaciais. In: *I Workshop Brasileiro de Jogos e Entretenimento Digital*. [S.l.: s.n.], 2002.

DICTIONARIES, O. *Oxford English Dictionary*. [S.l.]: Oxford University Press, 2013. v. 7.

DZIUK, A.; MIIKKULAINEN, R. Creating intelligent agents through shaping of coevolution. In: IEEE. *2011 IEEE Congress of Evolutionary Computation (CEC)*. [S.l.], 2011. p. 1077–1083.

FOUNDATION, K. *Krita*. 1998. Disponível em: <https://krita.org>.

FRANKLIN, S.; GRAESSER, A. Is it an agent, or just a program?: A taxonomy for autonomous agents. In: SPRINGER. *International workshop on agent theories, architectures, and languages*. [S.l.], 1996. p. 21–35.

FREIRE, M.; SERRANO-LAGUNA, Á.; MANERO, B.; MARTÍNEZ-ORTIZ, I.; MORENO-GER, P.; FERNÁNDEZ-MANJÓN, B. Game learning analytics: learning analytics for serious games. In: *Learning, design, and technology*. [S.l.]: Springer Nature Switzerland AG, 2016. p. 1–29.

FU, F.-L.; SU, R.-C.; YU, S.-C. Egameflow: A scale to measure learners' enjoyment of e-learning games. *Computers & Education*, Elsevier, v. 52, n. 1, p. 101–112, 2009.

GALVÃO, T. A. B.; NETO, F. M. M.; BONATES, M. F.; CAMPOS, M. T. A serious game for supporting training in risk management through project-based learning. In: SPRINGER. *International Conference on Virtual and Networked Organizations, Emergent Technologies, and Tools*. [S.l.], 2011. p. 52–61.

GARRIS, R.; AHLERS, R.; DRISKELL, J. E. Games, motivation, and learning: A research and practice model. In: *Simulation in Aviation Training*. [S.l.]: Routledge, 2017. p. 475–501.

GHELLI, G. M. A construção do saber no ensino superior. *Cadernos da FUCAMP*, v. 3, n. 3, p. 79–96, 2004.

HOPPE, L.; KROEFF, A. M. S. Educação lúdica no cenário do ensino superior. *Veras*, v. 4, n. 2, p. 164–181, 2014.

HUITT, W. Bloom et al.'s taxonomy of the cognitive domain. *Educational psychology interactive*, v. 22, 2011.

HUIZINGA, J. *Homo ludens*. [S.l.]: Editora Perspectiva SA, 2020.

"INKARNATE", I. E. L. *INKARNATE*. 2012. Disponível em: <https://inkarnate.com>.

INSTITUTE, P. P. M. *Guia PMBOK®: Um Guia para o Conjunto de Conhecimentos em Gerenciamento de Projetos*. [S.l.: s.n.], 2017. Sexta edição.

ISO/IEC. 31010: Risk management—risk assessment techniques. 2019.

KOSA, M.; YILMAZ, M.; O'CONNOR, R.; CLARKE, P. Software engineering education and games: a systematic literature review. *Journal of Universal Computer Science*, Technische Universitaet Graz* Institut fuer Informationssysteme und Computer ..., v. 22, n. 12, p. 1558–1574, 2016.

LADEWIG, I. A importância da atenção na aprendizagem de habilidades motoras. *Revista paulista de educação física*, v. 3, p. 62–71, 2000.

LEIGH, D. Swot analysis. *Handbook of Improving Performance in the Workplace: Volumes 1-3*, Wiley Online Library, p. 115–140, 2009.

LEOPOLDINO, C. B. Avaliação de riscos em desenvolvimento de software. 2004.

LIMA, L. A. L. Gestão de riscos em projetos de software. 2009.

LUCKESI, C. C. *Educação, Ludicidade e Prevenção das Neuroses Futuras: uma proposta pedagógica a partir da Biossíntese. 2005*. 2017.

MACHADO, V.; JÚNIOR, P. A.; COSTA, H. *Catálogos de Riscos em Gerência de Projetos de Software*. Tese (Doutorado) — Universidade Federal de Lavras, 2014.

MARRO, A. A.; SOUZA, A. M. d. C.; CAVALCANTE, E. R. d. S.; BEZERRA, G. S.; , R. d. O. N. . Lógica fuzzy: Conceitos e aplicações. 2010.

OLIVEIRA, C.; CINTRA, M.; NETO, F. M. Learning risk management in software projects with a serious game based on intelligent agents and fuzzy systems. In: ATLANTIS PRESS. *8th conference of the European Society for Fuzzy Logic and Technology (EUSFLAT-13)*. [S.l.], 2013. p. 874–879.

OLIVEIRA, L. R. de; GOMES, G. S.; LIMA, F. P. de. Análise de riscos pelo uso de métodos ágeis na gestão de projetos de desenvolvimento de software. *Revista de Gestão e Projetos*, v. 5, n. 2, p. 90–101, 2014.

OLIVEIRA, R. N. R. de; CARDOSO, R. P.; BRAGA, J. C. B.; ROCHA, R. V. da. Frameworks para desenvolvimento de jogos educacionais: uma revisão e comparação de pesquisas recentes. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2018. v. 29, n. 1, p. 854.

PALUDO, L.; RAABE, A. L. A.; BENITTI, F. B. V. Rskmanager–um jogo para apoiar o ensino de gerência de riscos em projetos de software. *RENOTE*, v. 11, n. 3, 2013.

PETRI, G.; BATTISTELLA, P. E.; CASSETTARI, F.; WANGENHEIM, C. G. von; HAUCK, J. Um quiz game para a revisão de conhecimentos em gerenciamento de projetos. In: *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*. [S.l.: s.n.], 2016. v. 27, n. 1, p. 320.

PETRI, G.; WANGENHEIM, C. G. von. How games for computing education are evaluated? a systematic literature review. *Computers & education*, Elsevier, v. 107, p. 68–90, 2017.

PETRI, G.; WANGENHEIM, C. G. von; BORGATTO, A. F. Meega+: Um modelo para a avaliação de jogos educacionais para o ensino de computação. *Revista Brasileira de Informática na Educação*, v. 27, n. 03, p. 52–81, 2019.

SANTOS, H. M. dos; DURELLI, V. H.; SOUZA, M.; FIGUEIREDO, E.; SILVA, L. T. da; DURELLI, R. S. Cleangame: Gamifying the identification of code smells. In: *Proceedings of the XXXIII Brazilian Symposium on Software Engineering*. [S.l.: s.n.], 2019. p. 437–446.

SANTOS, S. H. N.; COSTA, Y. d. J. S.; SANTOS, D. V. dos; FILHO, A. O. B.; JUNIOR, J. B. B.; CABREJOS, L. J. E. R. Identificando jogos sérios para o ensino de engenharia de software no brasil através de um mapeamento sistemático. *Research, Society and Development*, v. 9, n. 7, p. e329973702–e329973702, 2020.

SAVI, R.; WANGENHEIM, C. G. von; BORGATTO, A. F. A model for the evaluation of educational games for teaching software engineering. In: IEEE. *2011 25th Brazilian Symposium on Software Engineering*. [S.l.], 2011. p. 194–203.

SERRANO-LAGUNA, Á.; MANERO, B.; FREIRE, M.; FERNÁNDEZ-MANJÓN, B. A methodology for assessing the effectiveness of serious games and for inferring player learning outcomes. *Multimedia Tools and applications*, Springer, v. 77, n. 2, p. 2849–2871, 2018.

SINDRE, G.; NATVIG, L.; JAHRE, M. Experimental validation of the learning effect for a pedagogical game on computer fundamentals. *IEEE Transactions on Education*, IEEE, v. 52, n. 1, p. 10–18, 2008.

SOMMERVILLE, I. *Engenharia de Software*. [S.l.]: Editora Pearson, 2016. 768 p. ISBN 9788543024974.

SOUZA, M. R. D. A.; VEADO, L. F.; MOREIRA, R. T.; FIGUEIREDO, E. M. L.; COSTA, H. A. X. Games for learning: Bridging game-related education methods to software engineering knowledge areas. In: IEEE. *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*. [S.l.], 2017. p. 170–179.

SQUIRE, K. Video games in education. *Int. J. Intell. Games & Simulation*, Citeseer, v. 2, n. 1, p. 49–62, 2003.

UMA Lista de Riscos na Implantação de Sistemas ERP.

WANGENHEIM, C. G. von; SAVI, R.; BORGATTO, A. F. Deliver!–an educational game for teaching earned value management in computing courses. *Information and software Technology*, Elsevier, v. 54, n. 3, p. 286–298, 2012.

WILSON, L. Bloom's taxonomy revised. *Understanding the Revised Version of*, 2001.

WILSON, L. O. Anderson and krathwohl–bloom's taxonomy revised. *Understanding the New Version of Bloom's Taxonomy*, 2016.

WOOLDRIDGE, M. Intelligent agents. *Multiagent systems*, MIT Press Cambridge, v. 6, 1999.

WOOLDRIDGE, M.; JENNINGS, N. R. Intelligent agents: Theory and practice. *The knowledge engineering review*, Cambridge University Press, v. 10, n. 2, p. 115–152, 1995.

XIE, T.; TILLMANN, N.; HALLEUX, J. D. Educational software engineering: Where software engineering, education, and gaming meet. In: IEEE. *2013 3rd International Workshop on Games and Software Engineering: Engineering Computer Games to Enable Positive, Progressive Change (GAS)*. [S.l.], 2013. p. 36–39.

ZADEH, L. Zadeh, fzzy sets. *J. Klir and B. Yuan, Editors, Fuzzy sets, fuzzy logic, and fuzzy systems*, p. 19–34, 1965.

ZHONGGEN, Y. A meta-analysis of use of serious games in education over a decade. *International Journal of Computer Games Technology*, Hindawi, v. 2019, 2019. Disponível em: <https://doi.org/10.1155/2019/4797032>.