UNIVERSIDADE FEDERAL DE SANTA CATARINA
SISTEMAS DE INFORMAÇÃO

RAFAEL EUCLIDES DAMASCO

RETRAINING: UMA SOLUÇÃO COMPUTACIONAL PARA APOIAR O ENSINO DE ESPECIFICAÇÃO DE REQUISITOS

FLORIANÓPOLIS
2021

RAFAEL EUCLIDES DAMASCO

RETRAINING: UMA SOLUÇÃO COMPUTACIONAL PARA APOIAR O ENSINO DE ESPECIFICAÇÃO DE REQUISITOS

Trabalho Conclusão do Curso de Graduação em Sistemas de Informação do Centro de Tecnologia da Universidade Federal de Santa Catarina como requisito para a obtenção do Título de Bacharel em Sistemas de Informação
Orientador: Prof. Fabiane Barreto Vavassori Benitti

Universidade Federal de Santa Catarina

2021

RESUMO

A Engenharia de Requisitos é responsável em identificar, analisar, documentar e gerenciar os requisitos de um software. Por esses motivos, esta se trata de uma fase importante no processo de criação de um software. Diversas técnicas e métodos de levantamento que facilitam a realizar a obtenção e gerenciamento desses requisitos. Porém, atualmente boa parte dos projetos de software que falham com o cumprimento de prazos e orçamentos, isso muitas vezes são ocasionados por falhas nos requisitos, já que esses são custosos de corrigir. Devido ao difícil aprendizado dessas técnicas que vêm sendo feitas normalmente através somente de maneira teórica, a elicitação e análise de requisitos acabam sendo mal feitas ou deixadas de lado durante o processo de desenvolvimento, é difícil ocorrer métodos mais práticos e dinâmicos para a transmissão do conhecimento. Este trabalho propõe criar um software para apoiar o ensinamento de técnicas de levantamento e análise de requisitos de forma mais ativa.

Palavras chaves: Tcc; Ine; Ufsc; Engenharia; Engenharia de Software; Especificação; Documentação; Elicitação; Requisitos; Levantamento; Ensino-aprendizagem.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

Índice

1 INTRODUÇÃO

O termo Engenharia de Software surgiu em 1968, tendo sido utilizado pela primeira vez por uma engenheira da NASA, Margaret Hamilton, na NATO Conference on Software Engineering (ERDOGMUS; MEDVIDOVIĆ; PAULISCH, 2008), tratando o desenvolvimento de software de uma forma mais sistemática, se preocupando com todos os aspectos da produção de software e visando a empresa como um todo. “Engenharia de software é uma abordagem sistemática e disciplinada para o desenvolvimento de software” (PRESSMAN, 2006). A área de estudos dos softwares surgiu com a necessidade de lidar com o aumento de diversidade, demandas pela diminuição do tempo para entrega e desenvolvimento de software confiável, e para sanar esse problema, são utilizados processos, ou melhor uma sequência de métodos com a finalidade de desenvolver um software. Os processos de software são basicamente divididos em 4 etapas, especificação, projeto, implementação, validação e evolução (PRESSMAN, 2006). Magela (2006) entende como Engenharia de Software

> “O conjunto de técnicas, métodos, ferramentas e processos utilizados na especificação, construção, implantação e manutenção de um software que visa a garantir a gerência, o controle e a qualidade dos artefatos gerados através de recursos humanos.”

Uma área importante da Engenharia de Software é a Engenharia de requisitos, que dispõe-se a facilitar a compreensão de um problema pelo engenheiro de software, e com a finalidade de sanar os problemas do desenvolvimento de aplicações.

> “Especificação de software ou engenharia de requisitos é o processo de compreensão e definição dos serviços requisitados do sistema e identificação de restrições relativas à operação e ao desenvolvimento do sistema. (SOMMERVILLE, 2011)”.

É através da Engenharia de requisitos que pode-se viabilizar os insumos necessários para a elaboração do projeto, o que o cliente deseja do projeto e como os usuários finais vão interagir com o projeto. A principal necessidade da Engenharia de requisitos, como o próprio nome diz, é definir os requisitos do sistema, estabelecendo o que o sistema deve fazer e quais são suas restrições.

Sommerville (2005) descreve Requisitos como as descrições das suas funções e restrições, sendo o processo de descobrir, analisar, documentar e verificar essas funções e restrições denominadas de Engenharia de requisitos.

Os requisitos elicitados e analisados são escritos em um documento chamado documento de requisitos de software, ou também chamado de Especificação de Requisitos de Software, que é o produto final do processo de descobrimento de requisitos que reúne necessidades e propósitos demandados pelos stakeholders, neste documento consta todos os requisitos que o sistema deve ter. Este documento pode ser considerado “uma declaração formal de requisitos de clientes, usuários finais e desenvolvedores de software” (SOMMERVILLE & SAWYER, 1997). Um documento de requisitos, segundo a norma (IEEE, 1996), deverá conter “declarações não ambíguas e ser verificável, consistente, modificável, rastreável e usável durante todas as fases do ciclo de vida do requisito”.

Para Sommerville (2011) uma das maiores dificuldades dos engenheiros de software é compreender as informações que o cliente transmite e elaborar um documento de requisito que esteja de acordo com a realidade do software requisitado. Para que os requisitos sejam bem compreendidos é importante que não haja margem para má interpretações, este é um dos grandes motivos para os atrasos no projeto, erro na elaboração do escopo e no orçamento.

1.1 PROBLEMA

De acordo com pesquisa realizada pelo Standish Group (2015), apenas 29% de grandes projetos foram entregues com sucesso, ou seja, realizados no tempo e orçamento planejado e com uma implementação satisfatória, 19% foram cancelados e os outros 52% foram projetos em que se ultrapassou o orçamento, atrasou o prazo e/ou não teve uma implementação satisfatória.

É de praxe que erros ocorram em, pelo menos, alguma das etapas de análise e desenvolvimento. Porém, um momento crítico para se encontrar um erro de software é após a fase de requisitos, quando o software já está em produção ou já foi entregue. Estudos indicam que, erros encontrados em requisitos de software, descobertos após a implementação do software, tendem a custar até 20 vezes mais para se corrigir do qualquer outro tipo de erro, e até 200 vezes mais custosos do que seriam se corrigidos durante a fase de engenharia de requisitos (BOEHM, 1989).

Para minimizar estes problemas o profissional da área deve possuir uma comunicação interpessoal, sólidos conhecimentos técnicos, aprofundar-se nas regras de negócio de cada cliente. Para então definir os requisitos, analisá-los, determinar se estes estão incompletos, ambíguos, contraditórios e torná-los úteis ou excluí-los; e por fim, evidenciar esses requisitos em um documento, que pode ser tanto um documento escrito em linguagem natural, casos de uso; estórias do usuário, dentre outros (IEEE, 1996).

É importante que sejam identificadas possíveis funções necessárias, adicionais ou diferentes, representando todos os requisitos e as restrições destes requisitos pretendidas pelo usuário e que estas restrições não sejam conflitantes (SOMMERVILLE, 2011).

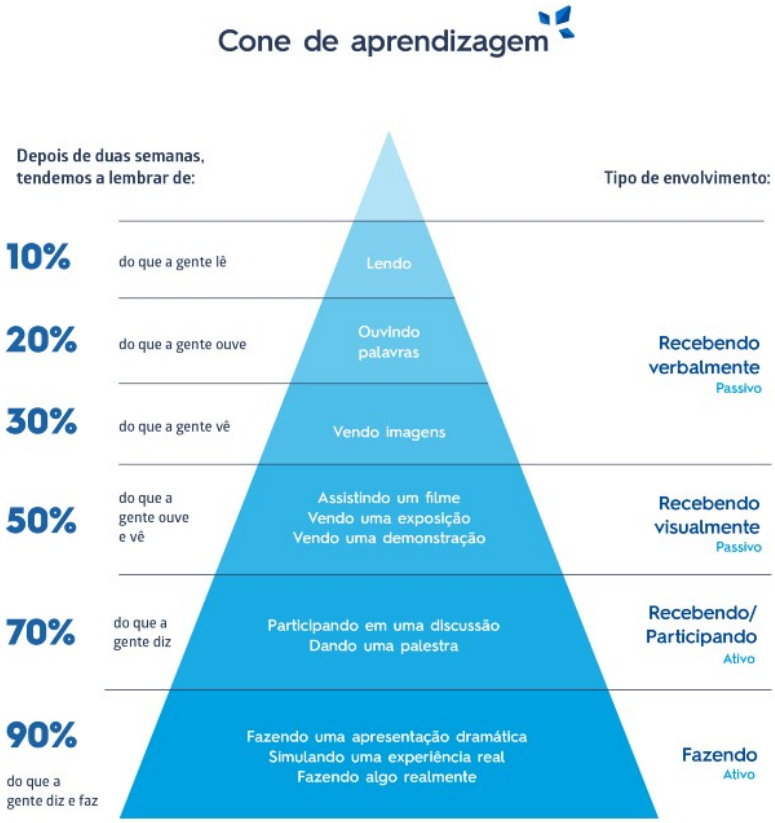
As empresas de desenvolvimento de software se queixam de que os profissionais recém-graduados não possuem as habilidades necessárias para atuar na área, causado muitas vezes pela falta de prática e experiência dos recém-formados. Esse contato prévio com a problemática de análise e documentação de requisitos é de extrema importância na área, haja vista, que a função de um analista de requisitos não é exata, podendo ser interpretada de forma diferente por um analista com mais experiência (VIZCAÍNO et al., 2010).

A má formação de profissionais pode estar ligada à forma de aprendizagem comumente usada nas universidades para o ensino de engenharia de requisito, chamada de aprendizagem passiva, onde os alunos são submetidos a sessões incessantes de conteúdo teórico e o professor apenas transmite a temática para o aluno, e por sua vez o aluno recebe essa temática sendo apenas um receptor do conteúdo e não interagindo com ele. “A boa educação é aquela em que o professor pede para que seus alunos pensem e se dediquem a promover um diálogo para promover a compreensão e o crescimento dos estudantes” (GLASSER, 2001).

O psiquiatra americano William Glasser formulou, a este respeito, o princípio da pirâmide do conhecimento (Figura 1). Sua teoria parte da ideia de que há similaridades importantes nas formas de aprender para todas as pessoas, e que os métodos tradicionais de ensinos não são os mais eficazes (GLASSER, 2001). Glasser (2001) “Não se deve trabalhar apenas com memorização, porque a maioria dos alunos simplesmente esquecem os conceitos após a aula”, de acordo com sua pesquisa, ao passar o período de 2 semanas, lembramos apenas de 30% quando sua forma de transmissão é verbalmente e que não tenha nenhuma

interação do aluno com esse conteúdo, como debater com os colegas, anotar ou resumir o assunto (Figura 1).

Figura 1 – Pirâmide de aprendizagem.



Fonte: (GLASSER,2001)

Apreender é o processo de transformar experiência em conhecimento (BLOOM 1956). A taxonomia de Bloom defini três domínios possíveis de aprendizagem, são eles: domínio cognitivo, domínio afetivo e domínio psicomotor. Abordaremos o domínio cognitivo como foco do presente trabalho.

Para Bloom o conhecimento é gerado a partir da organização das capacidades em níveis cognitivos e que há uma hierarquia de objetivos de aprendizagem a ser respeitada, partindo do nível mais simples para o mais complexo. Os níveis são descritos através de

categorias, e são: lembrar, compreender, aplicar, analisar, avaliar e criar. É possível visualizar similaridades na forma de aprendizagem proposta por Glasser e Bloom, ambos enfatizam a importância do ato de participar, aplicar e executar para gerar uma melhor assimilação dos conceitos

1.2 SOLUÇÃO PROPOSTA

A grande maioria das pessoas consegue absorver melhor um conteúdo através da aprendizagem ativa. (Figura 1) A aprendizagem ativa são práticas pedagógicas nas quais o aluno não é apenas um receptor de informações, ele deve interagir com um problema de maneira ativa para solucioná-lo, gerando experiências e melhorando a aprendizagem.

Assim, aprendizagem ativa ocorre quando o aluno interage com o assunto em estudo – ouvindo, falando, perguntando, discutindo, fazendo e ensinando – sendo estimulado a construir o conhecimento em vez de recebê-lo de forma passiva do professor. Em um ambiente de aprendizagem ativa, o professor atua como orientador, supervisor, facilitador do processo de aprendizagem, e não apenas como fonte única de informação e conhecimento (BARBOSA; MOURA, 2013, p.55).

A aprendizagem ativa pode ser viabilizada por meio de jogos e ferramentas de simulação, nas quais o usuário interage com um determinado problema assiduamente para tentar resolvê-lo, as pessoas são capacitadas pela vivência com situações realísticas que podem ser encontradas na prática. Tal abordagem permite o desenvolvimento de experiências sem os riscos existentes no mundo real. Além disso, utilizar abordagens de jogos e simulação permite ao estudante aprender fazendo, reduzindo assim a lacuna existente entre teoria e prática. Os jogos estimulam e motivam, tornando o processo de aprendizagem muito mais rápido e proporcionam aos estudantes uma realidade onde ele terá que enfrentar desafios, fases, dificuldades, limites, a enfrentar fracassos e correr riscos, com segurança. (MORAN, 2012). Ao colocar essas técnicas de aprendizagem em prática com foco na documentação de requisitos, espera-se contribuir com a formação dos alunos na área de Requisitos, pois o aluno adquirirá um conhecimento prático sobre o assunto e estará mais preparado para formular um documento que condiz com as necessidades do cliente.

Este trabalho vem por meio de um jogo computacional, no formato RPG, permitir aos alunos exercitarem os conceitos de Engenharia de Requisitos, através da simulação de um ambiente organizacional de modo que os alunos possam “se aproximar” da prática de especificação de requisitos.

1.3 OBJETIVOS

Nas seções abaixo estão descritos o objetivo geral e os objetivos específicos deste trabalho.

Objetivo Geral

Desenvolver um jogo computacional no formato RPG para apoio ao ensino de especificação de Requisitos.

Objetivos Específicos

- Analisar abordagens de aprendizagem ativa para o ensino de Engenharia de Requisitos, com ênfase em jogos.

- Conceber e implementar um jogo para apoiar o ensino de documentação/especificação de requisitos de software.

- Avaliar o jogo desenvolvido.

1.4 METODOLOGIA

Para promover um melhor entendimento dos métodos abordados neste trabalho, esta seção foi dividida em metodologia de pesquisa e procedimento metodológico.

Metodologia de pesquisa

Para a elaboração deste trabalho, utilizou-se a metodologia de pesquisa indutiva, segundo (GIL, 2008) “De acordo com o raciocínio indutivo, a generalização não deve ser buscada aprioristicamente, mas constatada a partir da observação de casos concretos suficientemente confirmadores dessa realidade.” O método indutivo neste trabalho tem como finalidade a avaliação do jogo aqui proposto, buscando averiguar a qualidade do jogo e se ele contribui de alguma forma para o ensino de engenharia de requisitos, com enfoque na documentação dos requisitos.

Do ponto de vista dos objetivos, este trabalho classifica-se como uma pesquisa exploratória, para Gil (2008) “Pesquisas exploratórias são desenvolvidas com o objetivo de proporcionar visão geral, de tipo aproximativo, acerca de determinado fato”. Neste trabalho a pesquisa exploratória tem como finalidade a busca de jogos para o ensino de engenharia de software.

Quanto a natureza desta pesquisa, foi classificada como aplicada, tendo em vista que o jogo será desenvolvido para auxiliar o processo de ensino aprendizagem. Para Appolinário (2011), a pesquisa aplicada é realizada com o intuito de resolver problemas ou necessidades concretas e imediatas, ainda em relação a pesquisa aplicada, para Silva e Menezes (2001), uma pesquisa aplicada “objetiva gerar conhecimentos para aplicação prática dirigidos à solução de problemas específicos”.

Quanto à abordagem, este trabalho pode ser classificado como pesquisa qualitativa, tendo em vista que há busca de aperfeiçoamento do processo de ensino-aprendizagem de assuntos abordados.

Os procedimentos envolvidos neste trabalho são classificados como bibliográfico, visando um melhor entendimento dos assuntos abordados, procurando explicar um problema a partir de referências teóricas publicadas em documentos, “Utilizam-se dados de categorias teóricas já trabalhadas por outros pesquisadores e devidamente registrados”.(SEVERINO, 2007).

Procedimentos Metodológicos

Este trabalho é composto pelas seguintes etapas:

1. Gerência do projeto: Esta etapa visa a análise da viabilidade do projeto bem como todo seu planejamento.

a) Planejamento: Visa organizar o plano de execução do trabalho proposto.

b) Monitoramento e controle: Possui o objetivo garantir que o trabalho ocorre dentro do plano previsto.

2. Fundamentação bibliográfica: Esta etapa tem a função de captar insumos que serão utilizados para os objetivos específicos.

a) Engenharia de requisitos: levantamento da bibliografia acerca do tema proposto no trabalho.

3. Estado da arte: Esta etapa tem o objetivo de identificar e analisar recursos para ensino de Engenharia de Requisitos, focando em aprendizagem ativa:

a) Planejar o mapeamento sistemático: etapa de identificação das questões de pesquisa e elaboração do protocolo de revisão.

b) Condução: Seleção dos estudos, seguindo o protocolo.

c) Análise dos resultados: Análise e normalização dos dados captados, realização de uma síntese e descrição das evidências.

4. Desenvolvimento: Compreende o desenvolvimento do jogo, onde serão disponibilizados os resultados obtidos com os objetivos 1, 2 e 3

a) Planejamento: definição dos requisitos do jogo e elementos gráficos, e também todo o conteúdo abordado.

b) Desenvolvimento: Desenvolver aplicação.

c) Testes: Analisar os resultados da aplicação.

5. Aplicação do jogo: o jogo será avaliado por alunos que cursam áreas relacionadas aos cursos de tecnologia.

a) Planejamento: Definição do processo a ser realizado para avaliação dos objetivos expostos.

b) Execução: Executar o processo de avaliação definido.

c) Análise dos resultados: Sintetizar os resultados da avaliação.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda os aspectos de Engenharia de requisitos, evidenciando o seu processo, bem como conceitos e classificação, e por fim, é aprofundada a fase de documentação, foco do presente projeto.

2.1 ENGENHARIA DE REQUISITOS

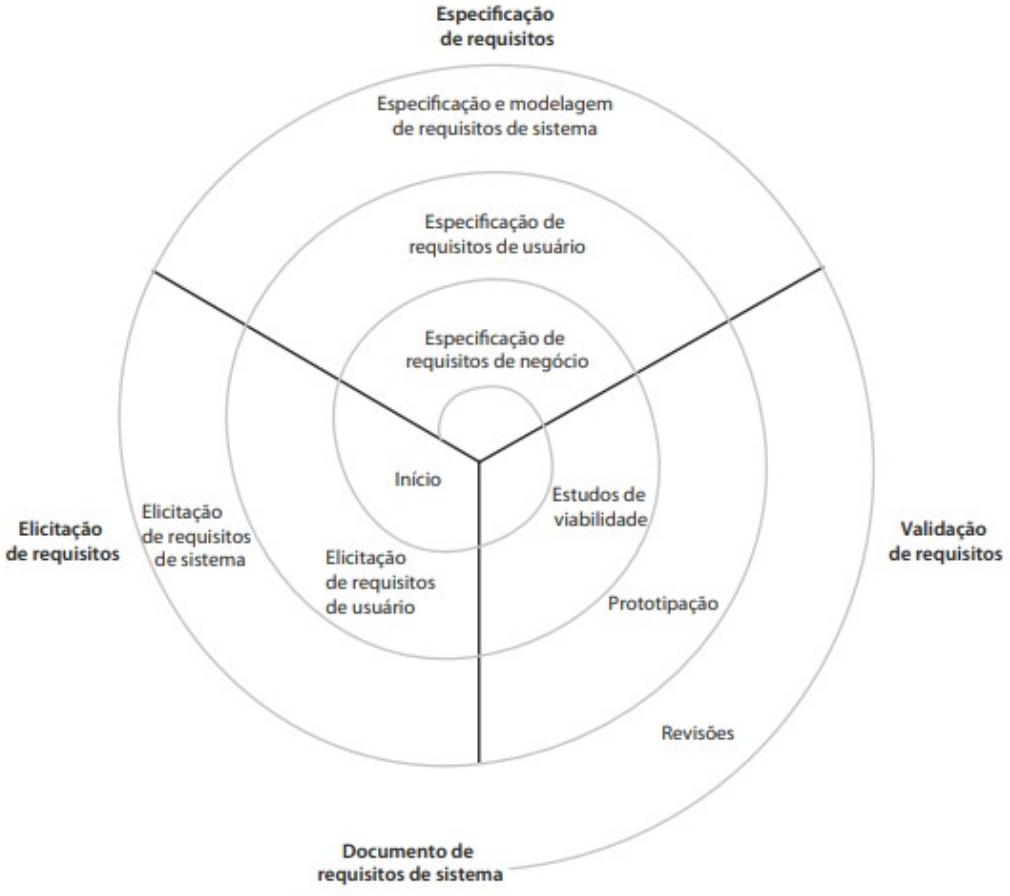
Boehm (1989) caracteriza a Engenharia de requisitos como uma área de conhecimento que se objetiva a desenvolver uma especificação íntegra, consistente e não ambígua, que funcionará como um contrato entre o desenvolvedor e o contratante para definir o que o produto fará.

De acordo com o IEEE (29148:2018), o procedimento que define a obtenção, verificação das necessidades do usuário e o aprimoramento destas necessidades é definido como engenharia de requisitos. O objetivo principal desta disciplina é compor métodos para que o processo de definição de requisitos seja feito de forma correta e completa, tornando o processo de desenvolvimento de software mais eficaz

2.2 PROCESSO DE ENGENHARIA DE REQUISITOS

Para Sommerville (2011) o processo de Engenharia de Requisitos se divide em 4 fases ou atividades de alto nível (Figura 2), conforme segue:

Figura 2 – Processo de engenharia de requisitos.



Fonte: (SOMMERVILLE, 2011)

- Estudo de viabilidade: Nessa fase do processo é realizada uma pesquisa com o intuito de esclarecer algumas dúvidas, dentre elas: (i) avaliar se a rentabilidade do sistema para a empresa será positiva; (ii) se o desenvolvimento do sistema está dentro dos limites orçamentais da empresa; e (iii) se o sistema que será desenvolvido pode ser integrado a algum outro sistema operante. “Este estudo visa verificar se o software é viável técnica e economicamente, e se os benefícios trazidos serão compensadores. O estudo de viabilidade requer que tenham sido definidos alguns requisitos para que se possa ter ideia do que será o sistema” (LEITE, 2000). Ao final do estudo de viabilidade, o resultado deverá informar se o projeto tem condições de continuar.

- Elicitação e análise de Requisitos: Por subsequente à fase do estudo de viabilidade, a elicitação e análise de requisitos tem como objetivo o processo de obtenção dos requisitos, que pode ser através da observação de sistemas já existentes ou da pesquisa junto ao requerente e usuários alvo do sistema. Nesta etapa é imprescindível a participação de todos os stakeholders do sistema, a fim de, amparar a descoberta de requisitos, para posteriormente classificá-los, priorizar os mais importantes e por fim especifica-los. Há algumas técnicas para identificar as fontes de informações, na criação de requisitos, como por exemplo a etnografia e entrevistas. Para um melhor entendimento do problema, pode ser necessário a criação de protótipos (SOMMERVILLE, 2011).

- Especificação de requisitos: Após as informações serem coletadas na etapa de elicitação e análise de requisitos, é necessário traduzir tais informações em um documento de requisitos. Há algumas técnicas para facilitar a visualização e compreensão destes requisitos, dentre as principais estão linguagem natural e os diagramas UML (NEIL; LAPLANTE, 2003). A linguagem natural muitas vezes pode levar a ambiguidade acarretando em diversos problemas, mas ainda sim é a mais utilizada segundo Sommerville. A UML por sua vez, é uma linguagem de modelagem (do inglês, UML - Unified Modeling Language) mundialmente utilizada para a elaboração da estrutura de projetos de software, podendo ser amplamente utilizada na visualização, especificação, construção e na documentação de requisitos (BOOCH, 2000). A seção 2.4 aprofundará aspectos da etapa de especificação de requisitos pois é o foco do presente projeto.

- Validação de Requisitos: Como última etapa do processo, a validação de requisitos tem o intuito de informar se o produto desenvolvido realmente é o que o cliente necessita (NEIL; LAPLANTE, 2003). Essa etapa tem como função verificar os requisitos quanto ao seu realismo, consistência e completude. No decorrer desse processo, os erros encontrados no documento de requisitos são inevitavelmente

descobertos. Posteriormente, o documento deve ser modificado para corrigir esses problemas (SOMMERVILLE, 2011).

2.3 O QUE SÃO REQUISITOS E TIPOS

A norma IEEE Std 610.12-1990 (IEEE, 1990, p. 62), define requisitos como:

- Capacidade necessária para o usuário resolver um problema ou atingir um objetivo; ou

- Capacidade que precisa ser atendida ou estar presente em um sistema ou componente, para satisfazer um contrato, uma norma, uma especificação ou outro documento imposto formalmente.

> "Requisitos são uma especificação do que deve ser implementado. Eles são descrições de como o sistema deve se comportar ou de uma propriedade ou atributo do sistema. Eles podem ser uma restrição no processo de desenvolvimento do sistema." (SOMMERVILLE; SAWYER, 1997).

Para Davis (1993), o termo requisito não é usado de forma consistente pela indústria de software, pois o termo "requisito" é usado de forma diferente dependendo da situação. Davis chega nesta afirmação através do exemplo da compra de software por parte de uma empresa, onde o requisitante define as necessidades do software de forma abstrata, para que a solução que este software deve resolver não fique pré-definida. Os requisitos são escritos de forma que vários contratantes possam oferecer maneiras diferentes de resolver um problema apresentado pela empresa. Após chegar a um acordo, este requisito que antes fora definido de forma ambígua, precisa ser refinado, tornando-se uma definição mais detalhada do sistema. Sommerville (2011) conclui que a falta de clareza na definição dos níveis de detalhamento dos requisitos pode gerar problemas durando o processo de engenharia de requisitos. Portanto há necessidade de diferenciar estes requisitos em relação ao seu nível de detalhamento, utilizando-se os termos:

- Requisitos de usuário

São declarações, em uma linguagem natural com diagramas, de quais serviços o sistema deverá fornecer a seus usuários e as restrições com as quais este deve operar, são voltados para os usuários que não possuem conhecimento técnico na área de engenharia de software (SOMMERVILLE, 2011).

- Requisitos de sistema

São descrições mais detalhadas dos requisitos de usuário, acrescentam especificidades quanto as funções, serviços e restrições operacionais do sistema de software. O documento de requisitos do sistema é utilizado pelos engenheiros de software como fonte inicial de informação, deve definir exatamente o que deve ser implementado, e pode ser parte do contrato entre o comprador do sistema e desenvolvedores de software.

Em relação à abrangência dos requisitos de sistema, podemos distinguir três tipos de requisitos (POHL, 2010). São eles:

- Requisitos Funcionais

“Um requisito funcional é um requisito relacionado ao resultado de algum comportamento a ser fornecido por uma função do sistema.” (POHL, 2010).

São as declarações das funções que o sistema deve oferecer, estão ligadas diretamente com a funcionalidade do software (YOUNG, 2004; PFLEEGER, 2004; SOMMERVILLE, 2005).

Requisitos funcionais são os requisitos que determinam funcionalidades, particularidades e ações que o sistema deve se comprometer a fazer. São facilmente visualizados com a ajuda de diagramas, principalmente diagramas de caso de uso. (ENGHOLM JUNIOR, 2010).

Lima (2007) conclui que os requisitos funcionais, descrevem ações que o sistema precisa executar.

- Requisitos Não Funcionais

Eles servem para restringir os serviços ou funções do sistema, e não tem relação com as funcionalidades que o sistema deve oferecer, preocupa-se com as propriedades fundamentais do sistema. Os requisitos não funcionais surgem com o intuito de solucionar os problemas dos usuários em relação as restrições de orçamento, políticas internas e externas ou compatibilidade com outro sistema ou hardware, podendo ser em relação ao tempo que uma função demora para ser finalizada, restrições no processo de desenvolvimento, como por exemplo a linguagem que o sistema será desenvolvido. Os requisitos não funcionais se preocupam com o sistema por completo (SOMMERVILLE, 2011).

Segundo Engholm Júnior (2010), “requisitos não funcionais descrevem atributos do sistema ou do ambiente do sistema”, tais como:

- Extensibilidade
- Usabilidade
- Confiabilidade
- Desempenho
- Escalabilidade
- Reusabilidade
- Capacidade de manutenção
- Reutilização de código
- Performance

- Eficiência no desenvolvimento

- Confiabilidade nos dados apresentados

• Regra de Negócio

Pohl (2010) ressalta que há um outro tipo de requisitos, que trata das restrições acerca das funcionalidades, conhecido também como requisitos de domínio, pois estão diretamente ligados ao domínio da aplicação. Leite e Leonardi (1998) entendem regras de negócio como:

> "Regras do Negócio são declarações sobre a forma da empresa fazer negócio. Elas refletem políticas do negócio. Organizações têm políticas para satisfazer os objetivos do negócio, satisfazer clientes, fazer bom uso dos recursos, e obedecer às leis ou convenções gerais do negócio. Regras do Negócio tornam-se requisitos, ou seja, podem ser implementados em um sistema de software como uma forma de requisitos de software desse sistema"

Leite e Leonardi (1998) enfatizam que as regras de negócio refletem às necessidades do negócio e modelam como o sistema deve responder há eventos ocorridos no mundo real.

2.4 DOCUMENTAÇÃO DE REQUISITOS

Após as informações serem extraídas nas etapas anteriores da engenharia de requisitos, é indispensável que haja uma documentação para que as partes envolvidas saibam como o software será feito. Serão explicitadas neste documento informações que dizem respeito aos protocolos de entrevistas, relatórios de atividades de validação e acordo, bem como solicitações de mudanças no software, porém a principal função do documento é documentar os requisitos do sistema de forma apropriada (POHL, 2010).

> "A especificação de requisitos é o processo de escrever os requisitos de usuário e de sistema em um documento de requisitos. Idealmente, os requisitos de usuário e de sistema devem ser claros, inequívocos, de fácil compreensão, completos e consistentes. (SOMMERVILLE, 2011)"

Segundo a ISO 29148, para que um requisito seja especificado de forma correta é necessário que:

- O requisito atenda as necessidades de um stakeholders, resolvendo um problema e/ou alcançando um objetivo.

- É qualificado por condições mensuráveis.

- É limitado por restrições.

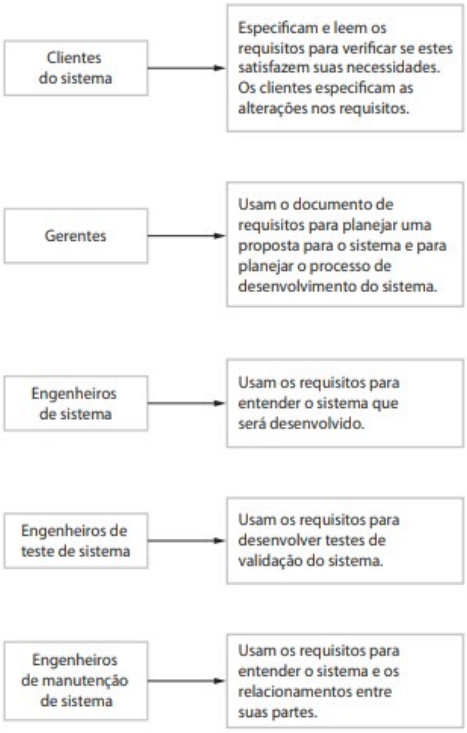
- Define o desempenho do sistema quando utilizado por um stakeholder específico ou a capacidade do sistema

- Pode ser verificado.

O documento de requisitos tem o objetivo de proporcionar conhecimento a um grupo de pessoas que vão interagir com o sistema, ele atende desde a alta administração da organização, que está arcando com os custos do sistema até os engenheiros responsáveis pelo desenvolvimento do software (KOTONYA, 1998; SOMMERVILLE, 1998).

Devido ao grande número de pessoas que atende (Figura 3), o documento de requisitos é um meio de comunicação de extrema importância, e a forma como ele é expressado deve ser de claro entendimento para todos, por isso a utilização da linguagem natural pode ser a mais indicada. Entretanto, apesar da linguagem natural ser de fácil compreensão a todos, ela também apresenta um alto grau de ambiguidade, deixando o documento inconsistente. (SOMMERVILLE, 2011; DAVIS, 1993)

Figura 3 – Usuários do documento de requisitos.



Fonte: (SOMMERVILLE, 2011)

Uma forma de contornar os problemas citados no parágrafo anterior e compreender sistemas complexos é a utilização de diagramas UML, através de um conjunto de diagramas que possibilita restringir nosso foco a um único aspecto por vez, auxiliando a equipe na visualização do problema, permitindo que o problema seja resolvido da forma mais rápida e correta (BOOCH, 2000). Na seção seguinte será apresenta a modelagem utilizando diagrama de casos de uso.

2.4.1 Modelagem de casos de uso

Esta modelagem surgiu na década de 1970, idealizada pelo engenheiro de software, Ivar Jacobson. Devido a sua notação gráfica simples e descrição em linguagem natural, facilita a comunicação entre o time de desenvolvedores e os stakeholders, a modelagem utilizando casos de uso vem se tornando cada vez mais popular para a elaboração de um documento de requisito de um sistema (BEZERRA, 2002).

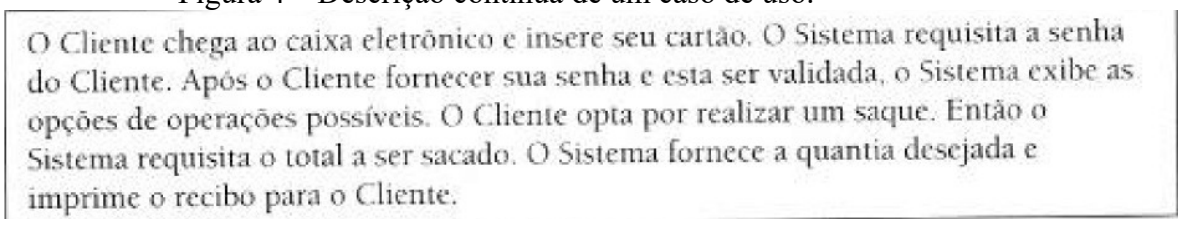
Os casos de uso são uma representação das funcionalidades do sistema e das entidades externas ao sistema que interagem com ele. Por se tratar de uma representação das funcionalidades do sistema, significa que estão ligados diretamente a um requisito funcional. Como citado brevemente na seção 2.4, a utilização desse modelo é parte fundamental da documentação de requisitos (SOMMERVILLE, 2011; BEZERRA, 2002).

2.4.1.1 Formato

Para Bezerra (2002) os formatos usualmente utilizados para representar casos de uso são a descrição contínua, a descrição enumerada e a descrição particionada. Segue exemplo do caso de uso realizar saque em um caixa eletrônico, utilizando as descrições mencionadas por Bezerra (2002):

- Descrição contínua: a descrição desse formato é feita utilizando-se um texto livre ou casual (Figura 4).

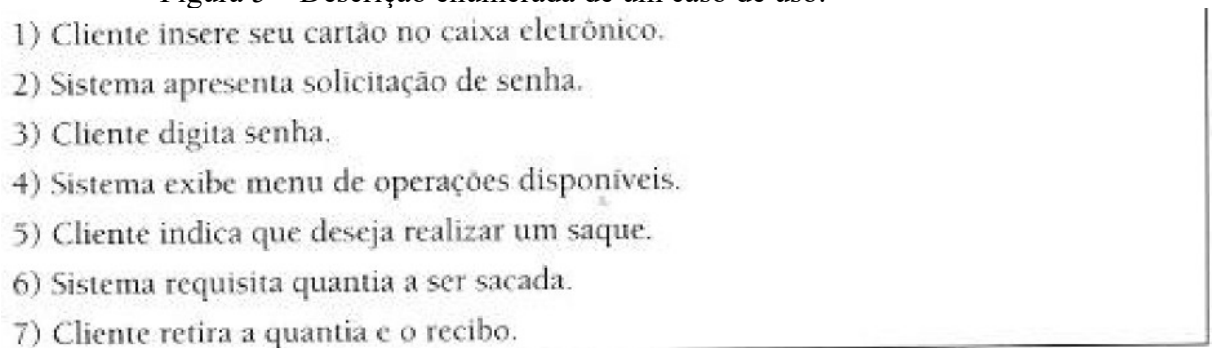
Figura 4 – Descrição contínua de um caso de uso.


O Cliente chega ao caixa eletrônico e insere seu cartão. O Sistema requisita a senha do Cliente. Após o Cliente fornecer sua senha e esta ser validada, o Sistema exibe as opções de operações possíveis. O Cliente opta por realizar um saque. Então o Sistema requisita o total a ser sacado. O Sistema fornece a quantia desejada e imprime o recibo para o Cliente.

Fonte: (BEZERRA, 2002)

- Descrição enumerada: esse formato é descrito por uma sequência de passos enumerados (Figura 5).

Figura 5 – Descrição enumerada de um caso de uso.


1) Cliente insere seu cartão no caixa eletrônico.
2) Sistema apresenta solicitação de senha.
3) Cliente digita senha.
4) Sistema exibe menu de operações disponíveis.
5) Cliente indica que deseja realizar um saque.
6) Sistema requisita quantia a ser sacada.
7) Cliente retira a quantia e o recibo.

Fonte: (BEZERRA, 2002)

- Descrição particionada: essa representação se difere dos formatos citados a cima pois divide-se em duas colunas, uma coluna representa a sequência de interações dos atores e a outra as interações do sistema (Figura 6).

Figura 6 – Descrição particionada de um caso de uso.

Cliente	Sistema
Insere seu cartão no caixa eletrônico.	
	Apresenta solicitação de senha.
Digita senha.	
	Exibe menu de operações disponíveis.
Solicita realização de saque.	
	Requisita quantia a ser sacada.
Retira a quantia e o recibo.	

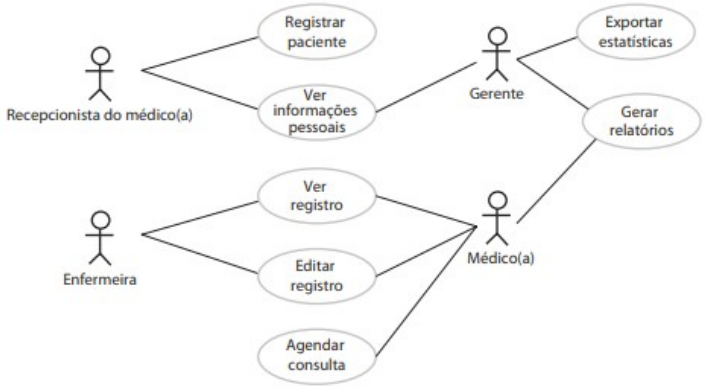
Fonte: (BEZERRA, 2002)

2.4.2 Diagrama de casos de uso

Esta é uma das muitas técnicas que podem ser aplicadas no processo de engenharia de requisitos, este método identifica os atores, que podem ser usuários, software ou hardware que por sua vez interagem com a aplicação e normalmente são representados por bonecos “palitos”. As interações são nomeadas caso de uso, que são as ações que o sistema deve executar, normalmente representados por uma elipse e conectada ao ator que executa esta ação. (SOMMERVILLE, 2011).

Nenhum sistema existe isoladamente. Todo e qualquer sistema interage com atores humanos ou autômatos. Os casos de uso normalmente são aplicados para identificar o comportamento e as relações entre sistemas, sem necessidade de mostrar como esse comportamento é implementado (BOOCH, 2000). Como pode ser visto na figura 7.

Figura 7 – Diagrama de casos de uso.



Fonte: (SOMMERVILLE, 2011)

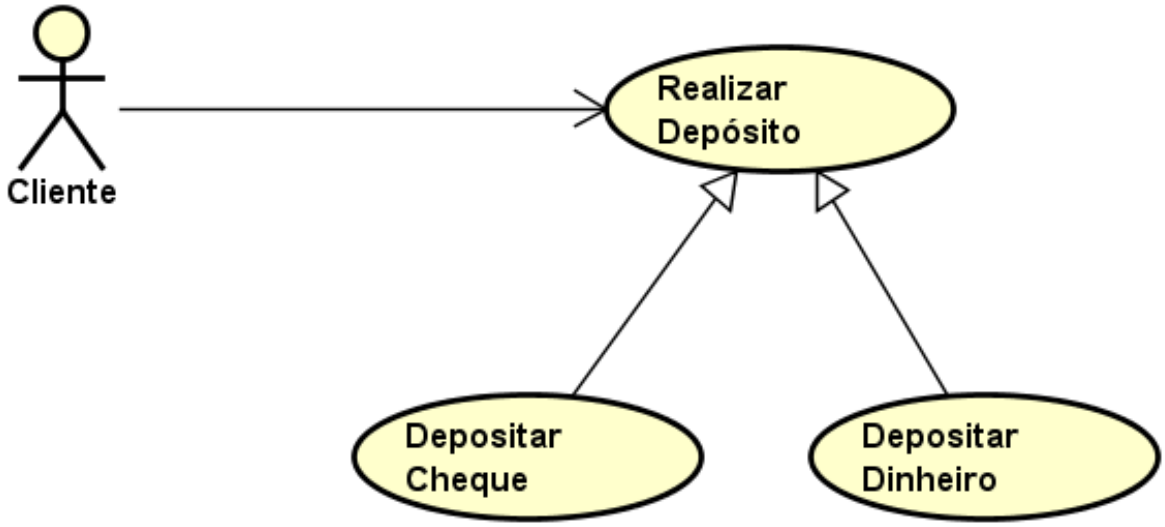
2.4.2.1 Relacionamentos de caso de uso

As principais relações entre os casos de uso são a generalização ou especialização, include e o extend.

- Generalização: A generalização tem a função de estabelecer uma relação de especialização tanto de casos de uso como de atores, sendo muito similar às encontradas na programação orientada a objetos. Na figura 8 o comportamento “depositar cheque” é uma especialização do caso de uso genérico “realizar depósito”,

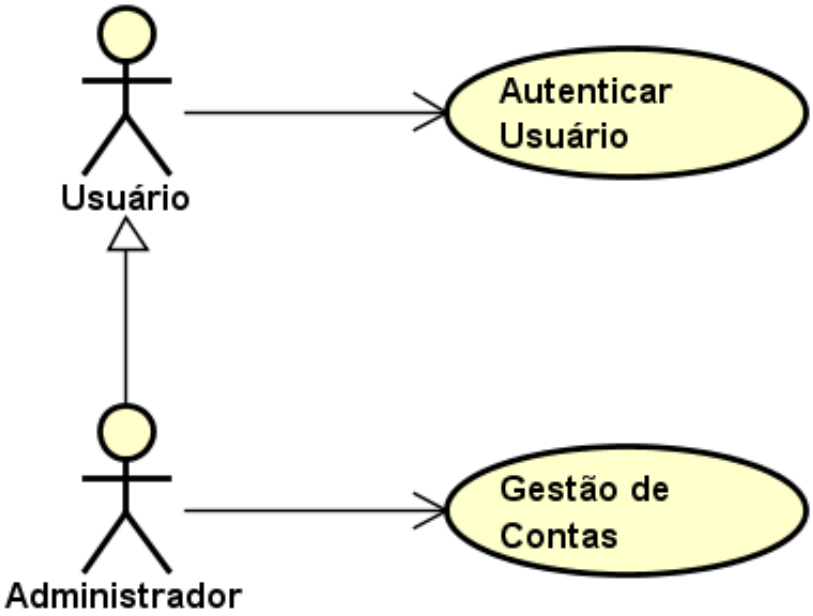
da mesma forma que na figura 9 o ator “administrador” é a especialização do ator genérico “usuário”.(VAZQUEZ, 2016; SIMÕES, 2016).

Figura 8 – Generalização de caso de uso



Fonte: LUQUE, 2017

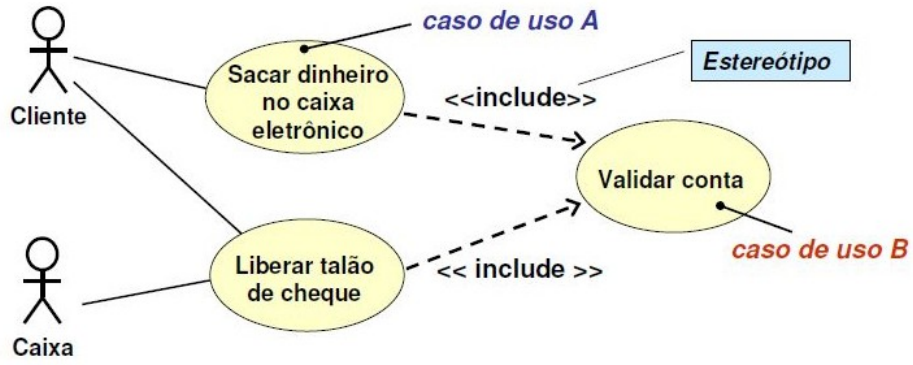
Figura 9 – Generalização de atores



Fonte: LUQUE, 2017

- Include: É a inclusão de um caso de uso pelo outro, implicando que o comportamento do caso de uso base é incorporado pelo caso de uso incluído. É uma relação de obrigatoriedade, significa que sempre que a atividade do caso de uso “Saque” é executada consequentemente a atividade do caso de uso “Registro Movimento” também deve ser executada. Para representar esta relação é utilizado uma seta pontilhada que se origina do caso de uso base em direção ao caso de uso incluído junto de um estereótipo «include», ver figura 10. (VAZQUEZ, 2016; SIMÕES, 2016).
-

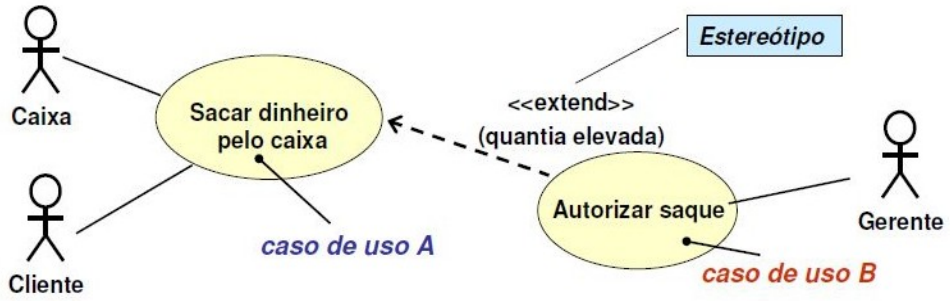
Figura 10 – Dependência entre de casos de uso <<include>>



Fonte: KOLB, 2014

- Extend: Um caso de uso pode estender outro, isto é, um caso de uso extensor pode ou não ser inserido no caso de uso estendido, o caso de uso “Saque” e “Deposito” não necessariamente será executado quando o caso de uso “Encerrar Conta” for executado. Esta relação também é representada por uma seta pontilhada, mas dessa vez ela se origina do caso de uso estendido para o caso de uso base junto de um estereótipo «extend», ver figura 11. (VAZQUEZ, 2016; SIMÕES, 2016)

Figura 11 – Dependência entre de casos de uso <<extend>>

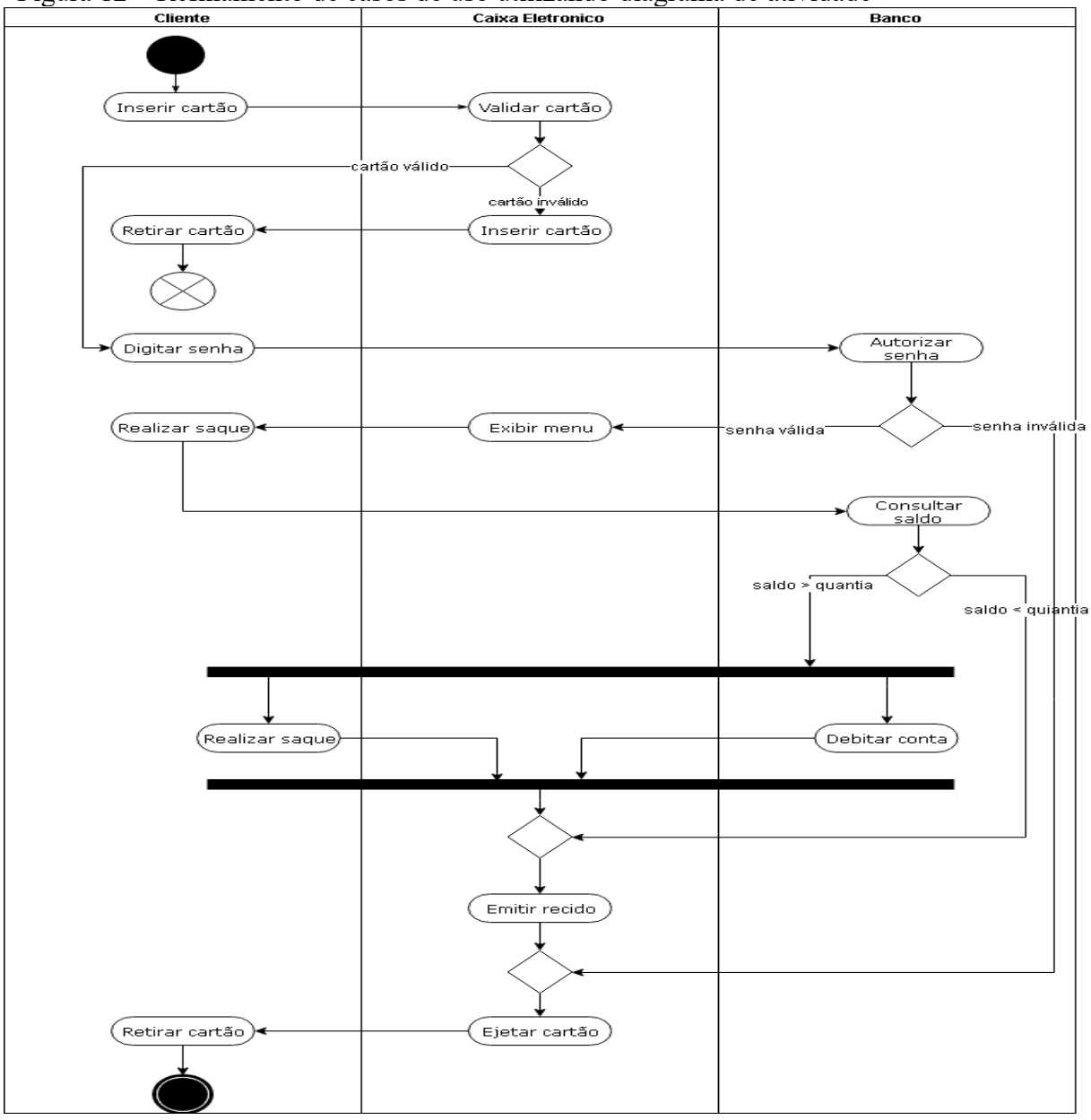


Fonte: KOLB, 2014

2.4.2.2 Diagrama de atividades no refinamento de casos de uso

A realização de um caso de uso requer que alguma lógica computacional seja aplicada e é através do diagrama de atividades que podemos detalhar esta lógica de forma clara e sucinta. O diagrama de atividade pode ser utilizado com o objetivo de refinar um caso de uso, sua função é de representar os estados de uma atividade, tendo em vista que a utilização do diagrama de casos de uso não se dispõem a detalhar o comportamento da atividade, simultaneamente com a carência de uma sintaxe para detalhar decisões, iterações e passos executados em paralelo (BEZERRA, 2002). Como pode ser visto na figura 12.

Figura 12 – Refinamento de casos de uso utilizando diagrama de atividade



Fonte: (Própria, 2019)

Bezerra (2002) afirma que é comum se deparar na documentação de casos de uso frases como “O passo P ocorre até que a condição C seja verdadeira”, ou “Se C ocorrer, vá para o passo P”. E são nessas situações que o refinamento do caso de uso com o diagrama de atividades é interessante. Bezerra (2002) ainda salienta que o diagrama de atividades deve ser utilizado para complementar a descrição de um caso de uso, e nunca substitui-lo.

O intuito deste capítulo foi evidenciar as formas de especificação de requisitos que são abordadas no presente projeto. As formas de especificação abordadas são:

- Documentação de requisitos utilizando linguagem natural, que compreende os requisitos funcionais, não funcionais e regra de negócio.
- Documentação de requisitos utilizando linguagem de modelagem unificada (UML), os modelos conceituais utilizados foram: diagrama casos de uso, e o seu refinamento através de cenários e o diagrama de atividade também utilizado para refinar casos de uso.

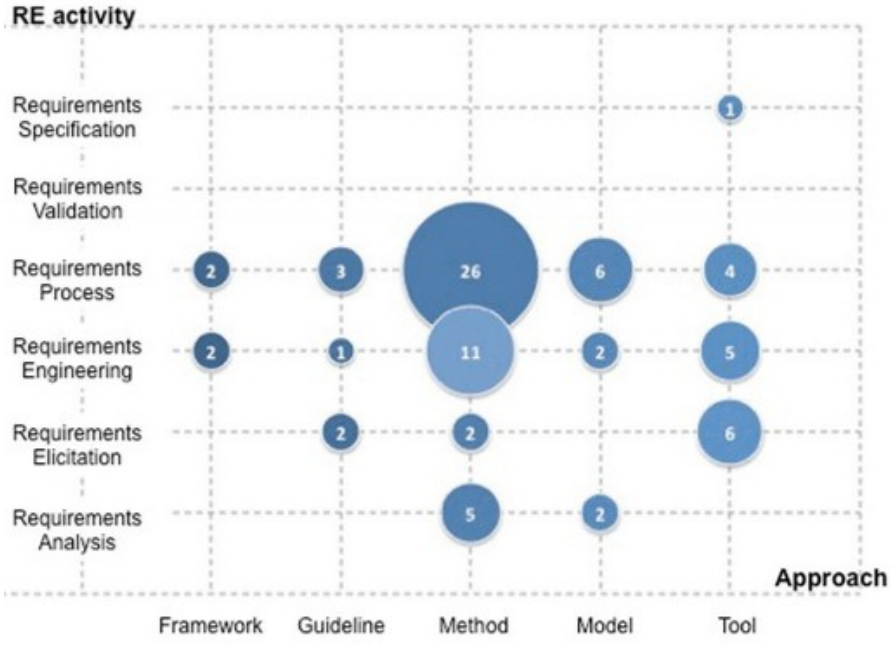
3 ESTADO DA ARTE

Este capítulo tem a função de evidenciar estudos sobre trabalhos relacionados ao tema proposto, baseando-se em práticas recomendadas para o mapeamento sistemático. Como primeiro passo para o mapeamento sistemático, buscou-se verificar a existência de mapeamento na área. Neste processo, foi identificado o artigo de Ouhdi (et al. 2015) intitulado “Requirements engineering education: a systematic mapping study”, o qual realizou um mapeamento sistemático nas seguintes bases de dados: IEEE, ACM, ScienceDirect e o Google Scholar, utilizando a seguinte string de busca:

‘‘Requirements’’ AND (‘‘Engineering’’ OR ‘‘Elicita-tion’’ OR ‘‘Specification’’ OR ‘‘Analysis’’ OR ‘‘Valida-tion’’ OR ‘‘Process’’) AND (‘‘Educat*’’ OR ‘‘Train*’’ OR‘‘Teach*’’ OR ‘‘Course’’ OR ‘‘Learn*’’ OR ‘‘Curricul*’’OR ‘‘Guide’’ OR ‘‘Coach*’’ OR ‘‘Explain*’’).

A pesquisa de Ouhdi (et al. 2015) identificou 1359 estudos, até 2015, dentre estes foram selecionados 79 estudos. Entre os artigos identificados, 52% apresentam abordagem para processos de requisitos, 27% dos trabalhos selecionados não definiram nenhuma atividade da engenharia de requisitos, portanto, não foram enquadrados na categoria engenharia de requisitos, 11% foram definidos como estudos sobre a elicitação de requisitos, 9% discutiram a análise de requisitos e apenas 1% dos trabalhos encontrados tratavam sobre a documentação de requisitos (Figura 13).

Figura 13 – Artigos identificados



Fonte: (OUHBI; IDRI; FERNÁNDEZ-ALEMÁN; JOSÉ; TOVAL, 2013)

Através de um recente mapeamento, o trabalho aqui presente buscou apenas identificar novos trabalhos a partir de 2015, baseado no protocolo já proposto por Ouhdi (et al. 2015).

3.1 PROTOCOLO

O objetivo desta pesquisa é identificar e analisar trabalhos correlacionados ao tema do trabalho aqui proposto, visando identificar soluções com o compromisso de ensinar engenharia de requisitos. As perguntas de pesquisa que se busca responder são:

- Quais são as abordagens que foram relatadas para o ensino de engenharia de requisitos? Para descobrir as existentes abordagem de ensino de engenharia de requisitos.

- Quais são as atividades de engenharia de requisitos que foram abordadas na literatura com a intenção de ensinar engenharia de requisitos? Para identificas qual o foco dos pesquisadores no ensino de engenharia de requisitos.

A string da busca utilizada é baseada na string original de Ouhdi (et al. 2015), limitada a busca na base de dados Scopus, compreendendo o período de 2015 até maio 2019. Os critérios de inclusão e exclusão estão definidos na tabela 1.

Tabela 1 – Artigos identificados

Critérios de inclusão	**Critérios de exclusão**
-Artigo que apresenta uma solução para o ensino na área de engenharia de requisitos.	-Artigos que não são focados na educação. -Artigos que apresentam software educacional, mas não são destinados a área de Engenharia de Requisitos. -Artigos resumidos (menos de 5 páginas) e que não tenham sido publicados em eventos ou periódicos da área de computação.

3.2 EXECUÇÃO DO PROTOCOLO

A execução da busca foi efetuada em maio de 2019, utilizando a string de busca formatada para a base Scopus, conforme segue:

(requirements AND (engineering OR elicitation OR specification OR analysis OR validation OR process) AND (education OR training OR teaching OR learning)) AND (LIMIT-TO (PUBYEAR , 2019) OR LIMIT-TO (PUBYEAR , 2018) OR LIMIT-TO (PUBYEAR , 2017) OR LIMIT-TO (PUBYEAR , 2016)) AND (LIMITTO (SUBJAREA , "COMP")) AND (LIMIT-TO (EXACTKEYWORD , "Software Engineering"))

Foram obtidos 326 resultados, de 2015 em diante, dos quais foram avaliados os 100 primeiros, ordenados por relevância. Após aplicação dos critérios de inclusão e exclusão, 9 estudos foram identificados.

3.3 ANÁLISE DOS ESTUDOS RELACIONADOS

Utilizando as strings e ferramentas de busca definidas, foram encontrados trabalhos correlacionados que se enquadrassem nos critérios de aceitação. Os dados foram coletados focando nos seguintes aspectos mostrados na tabela 2:

Tabela 2 – Trabalhos correlacionados

Nome do artigo	**Id do artigo**	**Abordagem de ensino**	**Atividades da ER**	**Foi aplicado na prática?**	**Quantos alunos?**
Active Learning with LEGO for Software Requirements	1	Jogo	Elicitação e análise de requisitos	Curso de Ciências da computação, em outono de 2017 na disciplina de Engenharia de software	21
Software Visual Specification for Requirement Engineering Education	2	Ferramenta	Especificação de requisitos	Engenharia de software	55
Experiences of using a game for improving learning in software	3	Jogo	Elicitação de requisitos	Alunos do quarto ano da Engenharia de computação, cursando a	94

requirements elicitation				disciplina Engenharia de Software	
TBL as an active learningteaching methodology for Software Engineering courses	4	Metodologia	Engenharia de requisitos	Curso de Engenharia de software, realizado em 3 semestres a partir de 2017	500
Teaching Requirement s Engineering Concepts using CaseBased Learning	5	Metodologia	Engenharia de requisitos	Alunos da pós-graduação em engenharia de software em outono de 2017	112
The use of Comic Strips in the teaching of Software Engineering Especificaca o de requisitos	6	Metodologia	Especificação de requisitos	Alunos do primeiro e do quarto ano de Ciências da computação	101

A tabela 2 possibilitou responder às perguntas da pesquisa elaboradas na sessão 3.1, evidenciando que as abordagens de ensino encontradas na pesquisa foram dois jogos, onde ambos os jogos tratavam de elicitação de requisitos como foco de ensino. Foram encontradas três metodologias onde duas delas não tinham uma área da engenharia de requisitos específica, portanto foi enquadrada na categoria “Engenharia de Requisitos” e a outra metodologia encontrada tem como foco ensinar a especificação de requisitos, que é o tema abordado neste trabalho. Observou-se também a utilização de uma ferramenta para ensinar a especificação de requisitos, como mostra a figura 14.

Figura 14 – Abordagens para o ensino de Engenharia de requisitos

Fonte: (Própria, 2019)

O presente trabalho, baseando-se nos trabalhos citados acima, aborda conceitos de especificação de requisitos através de um jogo computacional. Portanto, trata-se de uma forma de abordagem para o ensino de Engenharia de requisitos ainda não explorada pela literatura.

Foi através da análise dos trabalhos relacionados, que funcionalidades como: tempo para finalizar os projetos, dicas durante o desenvolvimento do jogo e feedback a respeito dos resultados obtidos pelo usuário, tornaram-se partes fundamentais acerca do jogo desenvolvido.

4 DESENVOLVIMENTO

Este projeto propõe um jogo computacional com o intuito de auxiliar o ensino de especificação de requisitos, tem como enfoque apresentar situações onde o usuário tenha contato com técnicas de especificação de requisitos por intermédio de um ambiente controlado. O jogo apresenta aos usuários documentos ou partes de documentos de requisitos, e o jogador/aluno deverá exercitar a habilidade de interpretar e complementar aspectos desta documentação.

O jogo foi projetado para ser jogado por um único jogador, a interação online, na qual a ação de um jogador interfere em outro jogador não faz parte do escopo deste trabalho, porém o progresso de todos que utilizaram o jogo poderá ser comparado e ranqueado para uma possível questão de competitividade.

As formas de especificação a serem abordadas no jogo serão: linguagem natural, cenários, casos de uso e refinamento de casos de uso com diagrama de atividade.

Não será englobado neste trabalho técnicas de estudo de viabilidade nem qualquer outro subprocesso de engenharia de requisitos.

4.1 VISÃO GERAL DO JOGO

Assim como todo projeto de sistemas, um jogo surge de uma ideia, que posteriormente recebe pequenas melhorias ou mudanças. Há algumas técnicas para fomentar o surgimento de ideias, seja para o jogo ou qualquer outro segmento que necessite ser alimentados pela criatividade, tais como o Brainstorm, que pode ser feito individualmente ou em grupo, com a intenção de elicitar o maior número de ideias para posteriormente selecionar o que realmente será usado no projeto (ESTEVES, 2020). Para que a ideia crie corpo e torne-se de fato um jogo que proporcione uma experiência cativante ao jogador, é importante conhecer o público-alvo e colocar-se no lugar do player, tornando assim o jogo mais desafiador e atrativo (COPLE, 2019).

No que diz respeito a Design de games, quando a ideia central do jogo já está amadurecida, o ideal é seguir para a próxima etapa, que é a criação do Storyline, o qual exerce

a função de um roteiro preliminar. Para a criação do Storyline é importante compreender quem são os personagens do jogo, suas interações e a participação de cada um deles no enredo. (COPLE, 2019).

O próximo passo para o desenvolvimento do jogo é a criação do GDD (Game design document), que se trata de um conjunto de todos os elementos acerca do jogo reunidos em um único documento, ele serve como base para a elaboração da maioria dos projetos abrangendo a criação de jogos digitais (COPLE, 2019). O GDD serve para apoiar a equipe de desenvolvimento caso haja alguma dúvida sobre o que deverá ser desenvolvido e as delimitações acerca do projeto, portanto, o que não estiver especificado no GDD não faz parte do sistema desenvolvido (COPLE, 2019; FLYNT, 2005; PERUCIO, 2007).

Os tópicos abaixo foram adaptados das obras de Scott Rogers (2014) e Denis Cople (2019) para formular o GDD do presente trabalho.

Conceitos do Jogo: O jogo será voltado aos estudantes que pretendem revisar ou assimilar de forma mais descontraída o conteúdo abordado nas aulas de Engenharia de Software. O jogador estará exposto a práticas pedagógicas nas quais ele deve agir de forma ativa para resolver determinadas situações utilizando os conceitos de Engenharia de requisitos para resolver os projetos.

Nome: Retraining.

Número da versão, Autor, Data: Primeira versão do jogo Retraining; de autoria de Rafael Euclides Damasco; começou a ser projetado 1/04/2021.

Gênero: RPG

Público Alvo: O público-alvo do jogo serão os alunos de graduação dos cursos da área de tecnologia da informação.

Descrição do estilo de jogo: O jogo se passa em uma sala de escritório, com programadores e o chefe da empresa.

Síntese da história: A maioria das pessoas quando iniciam seus estudos buscam aplicá-los na prática, e o meio mais fácil e explorado pelos estudantes é iniciar em uma empresa como estagiário, e este será o ponto de partida do jogo. Como todo bom estagiário você será responsável por fazer e servir cafezinho para os outros integrantes da equipe de desenvolvimento, aprender os conteúdos que você não prestou atenção na sala de aula e agora precisa pôr em prática, participar das reuniões diárias, ser "bombardeado" de documentos de requisitos que você deve interpretar de forma correta. Quanto melhor o seu desempenho nos projetos, maior será a confiança do chefe em você, portanto, suas chances de efetivação também serão maiores.

Esboço do Jogo: Retraining é um jogo educacional em terceira pessoa, que transporta o usuário ao cotidiano de uma empresa de tecnologia iniciando sua carreira como estagiário, e que ao final de seu contrato almeja ser efetivado. E como todo início de carreira, passamos por um período de adaptação, falta de experiência e novos relacionamentos no âmbito profissional.

Como estagiário do projeto, de início você não participará dos projetos da equipe, por isso seu trabalho será um pouco mais fácil, você passará por um período de treinamento na empresa a fim de obter os conhecimentos necessários. O treinamento é composto por algumas fases, que abordarão perguntas de conhecimento geral e você terá a tarefa de analisar e completar essas perguntas. Com o passar do tempo e ao adquirir mais experiência você passará a ter contato direto com os projetos, aumentando sua responsabilidade na empresa, tornando-se responsável por auxiliar nos projeto presentes no contexto do jogo.

As formas de especificação a serem abordadas no jogo serão: linguagem natural, cenários, casos de uso e refinamento de caso de uso com diagrama de atividade.

Fluxo do jogo: Ao iniciar o jogo o usuário poderá dar nome ao seu personagem e escolher algumas características para o mesmo, como por exemplo: comunicativo, focado ou ponderado. Depois de escolher seus traços de personalidade seu personagem aparecerá na

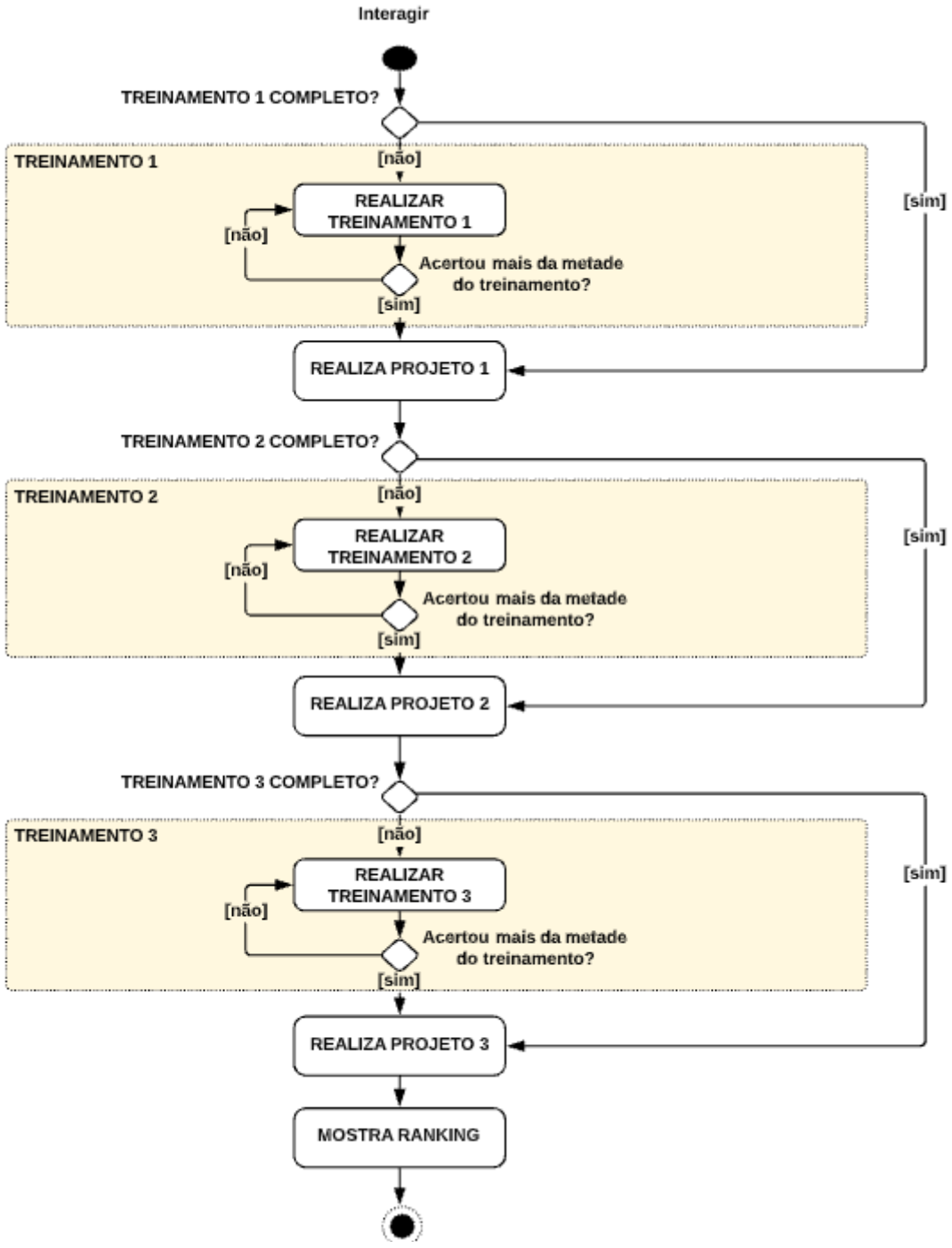
tela apresentando de forma visual qual personalidade foi escolhida.

No menu principal do jogo o usuário poderá ver a quantidade de experiência total que possui.

O jogo foi dividido em 3 fases, sendo que em cada fase o jogador passará por dois períodos distintos. Primeiro haverá um treinamento, e o usuário receberá informação a respeito de um determinado conteúdo, para posteriormente responder perguntas, com o intuito reforçar o conteúdo e alcançar o maior número de experiência possível. Caso o jogador tenha acertado mais da metade do treinamento, ele poderá em seguida colaborar com a equipe da empresa em um projeto. Vale lembrar que cada fase será composta por temáticas diferentes (Figura 15).

Ao terminar todos os projetos presentes nas 3 fases do jogo, o jogo é encerrado e disponibilizará uma tabela de ranking (Figura 15).

Figura 15 – Diagrama de atividade Retraining



Fonte: (Própria, 2021)

Fases de treinamento: No período de treinamento o usuário terá que responder à perguntas de conhecimento geral, para obter experiência o usuário poderá se deparar com questões de múltipla escolha, verdadeiro ou falso, completar e etc. Para completar cada fase de treinamento o usuário deverá acertar mais da metade das perguntas, possibilitando ao usuário contribuir para os projetos da empresa.

Fase 1: O treinamento dessa fase será relacionado com o tema de requisitos em linguagem natural.

Objetivos de aprendizagem:

- Compreender o que é um requisito de software.
- Diferenciar requisitos funcionais e não funcionais.
- Avaliar a qualidade de um requisito

Fase 2: O treinamento dessa fase será relacionado ao tema das relações entre os casos de uso generalização ou especialização, include e extend.

Objetivos de aprendizagem:

- Compreender os elementos do diagrama de casos de uso.
- Diferenciar os relacionamentos que podem existir no diagrama de casos de uso.

Fase 3: O treinamento dessa fase será relacionado ao tema de descrição de cenários e diagrama de atividade como forma de detalhar o comportamento de um caso de uso.

Objetivos de aprendizagem:

- Compreender os elementos do diagrama de atividade.
- Compreender o que são cenários e como descrevê-los.
- Identificar cenários principais, alternativos e de exceção.

Projetos: Os projetos farão com que o usuário tenha contato direto com o documento de especificação. Os projetos são formados por questões de múltipla escolha, verdadeiro ou falso, completar e etc. Os projetos se dividem em:

Projeto da fase: O projeto da fase será introduzido ao fim do período de treinamento da fase 1 e complementado ao final dos períodos de treinamento seguintes. Os projetos de fase serão relacionados aos assuntos apresentados em cada período de treinamento, para avançar nos projetos de fase o usuário apenas precisa responder as perguntas do projeto. Ao final de cada projeto de fase o jogador receberá uma quantidade de experiência que será computada no ranking.

Progressão:

A única forma de finalizar o jogo é passar pelo período de treinamento que lhe concederá acesso aos projetos de cada fase.

Personagens jogáveis: O personagem principal será um estagiário (Usuário), e será possível escolher entre alguns tipos de personalidades, estas serão: Comunicativo, Focado ou Ponderado.

Comunicativo: Possui a habilidade de ter um melhor desempenho nas reuniões e nas relações interpessoais, levando-o a ganhar mais tempo para realizar cada projeto, porém acaba se distraindo demais e ganhando menos experiência.

TEMPO PARA ATIVIDADE 3 MIN

EXPERIENCIA GANHA 90

Focado: Por ser muito esforçado em aprender, possui a habilidade de ganhar mais experiência ao realizar um projeto, porém não se da bem com a equipe e acaba não ganhando muito tempo para realizar um projeto.

TEMPO PARA ATIVIDADE 1 MIN

EXPERIENCIA GANHAR 110

Ponderado: Por ser muito equilibrado, possui um relacionamento mediano com a equipe e esforço para aprender.

TEMPO PARA ATIVIDADE 2 MIN

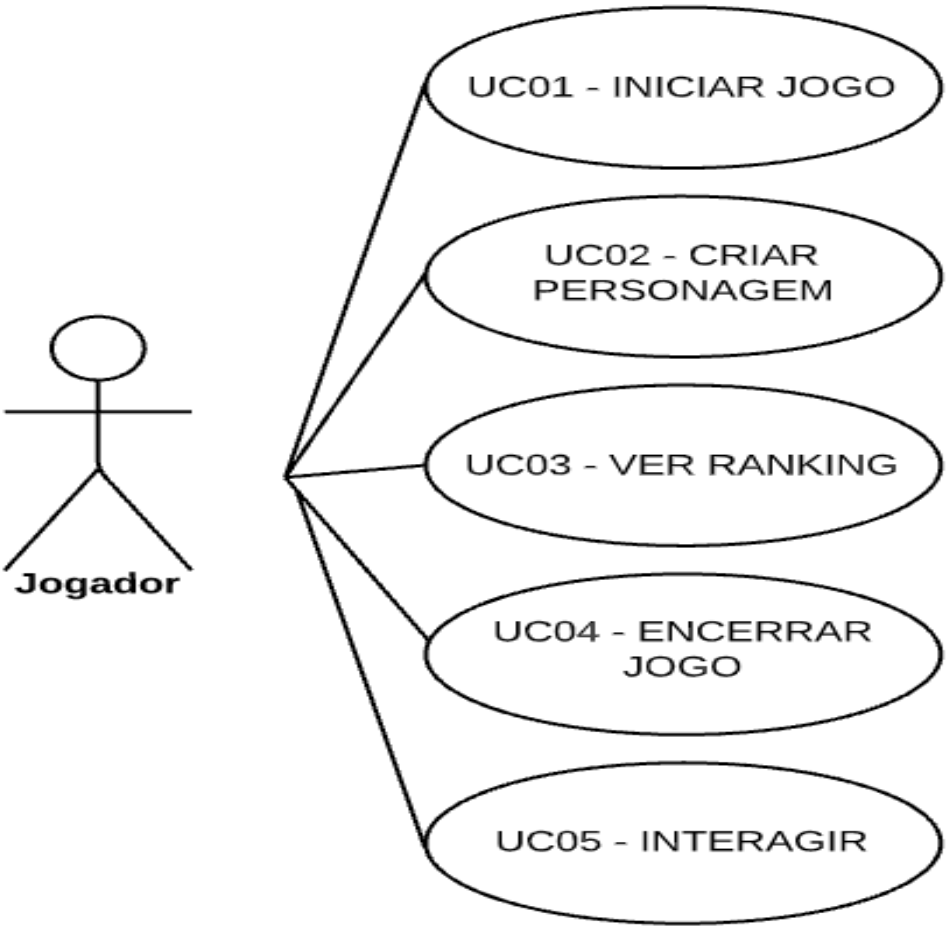
EXPERIENCIA GANHA 100

Personagens não jogáveis: Os personagens não jogáveis estarão inseridos no mundo do jogo e serão apresentados ao usuário conforme o seu andamento. Eles serão os programadores e o chefe da empresa.

4.2 DIAGRAMA DE CASOS DE USO DO JOGO

Nessa sessão será apresentado o diagrama de caso de uso do jogo conforme imagem abaixo (Figura 16). O detalhamento dos casos de uso abaixo através do diagrama de atividade podem ser encontrados no apêndice C.

Figura 16 – Diagrama de caso de uso Retraining



Fonte: (Própria, 2021)

UC01 – Iniciar jogo

- **Objetivo:** Inicia o jogo, e disponibiliza algumas orientações para o usuário

UC02 – Criar Personagem

- **Objetivo:** Escolher o nome do personagem e sua personalidade, bem como os atributos ligados a ela.

UC03 – Ver Ranking

- **Objetivo:** Mostrar o nome dos personagens que obtiveram o maior número de experiência.

UC04 – Encerrar Jogo

- **Objetivo:** Encerra o jogo.

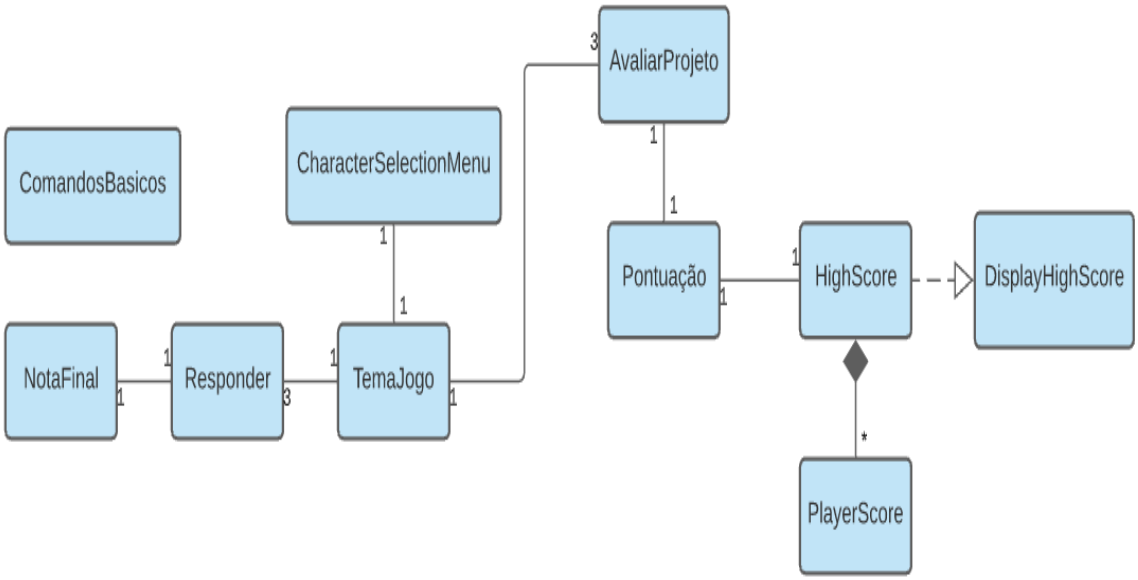
UC05 – Interagir

- **Objetivo:** Interagir com um NPC para receber dicas, realizar treinamento ou realizar o projeto.

4.3 DIAGRAMA DE CLASSE

A seguir é descrito o diagrama de classe do jogo (Figura 17). É importante salientar que muitas funcionalidades são feitas através da interface a Unity, e não são descritas neste diagrama.

Figura 17 – Diagrama de classe Retraining



Fonte: (Própria, 2021)

As classes contidas no diagrama serão descritas abaixo.

ComandosBasicos: Trata-se de uma classe responsável pela navegação entre as telas do jogo e para encerrar o jogo.

CharacterSelecitionMenu: Classe responsável por exibir os personagens do jogo, salvar os atributos de cada personagem e iniciar o jogo,

TemaJogo: Classe encarregada de apresentar ao usuário em que parte do jogo ele está, bem como quais elementos são gerados na tela de início do jogo, como por exemplo: nome do usuário, experiência e etc. É através dele que o usuário navega entre as fases de treinamento, projetos e resultados que obteve no jogo.

Responder: Está classe contem todas as questões relacionadas ao período de treinamento do jogo, é através desta classe que o sistema apresenta as questões ao jogador, quais ele acertou e se estará apto a avançar.

NotaFinal: Apresenta para o usuário informações sobre o seu desempenho ao responder as questões contidas no script responder.

AvaliarProjeto: Classe muito semelhante à classe responder, porém trata das questões relacionadas ao projeto do jogo. Quais questões serão apresentadas aos usuários, quantas ele acertou, quanto de experiência o usuário obteve. Também é responsável por iniciar a contagem regressiva presente nas questões do projeto.

Pontuação: Apenas contem as informações de experiência do usuário, para ser preenchida ou não na classe de HighScores.

HighScore: Classe que armazena as variáveis: nome e pontuação de cada jogador, além de fazer toda a comunicação com o servidor da dreamlo.

PlayerScore: Classe incumbida de armazenar as informações de nome e experiência final do usuário.

DisplayHighScore: Classe responsável por baixar os dados contidos no servidor da dreamlo e exibir ao usuário.

4.4 TECNOLOGIAS

Esta sessão é dedicada a apresentação das ferramentas usadas no desenvolvimento deste trabalho.

Unity

A Unity é um ambiente para o desenvolvimento de jogos, possui um conjunto de ferramentas integrado que permite criar jogos tanto em 3D quanto em 2D. Com a Unity o desenvolvedor tem a capacidade de construir seus jogos para diversas plataformas e a facilidade de utilizar objetos prontos através da loja oficial da Unity ou da comunidade desenvolvedora.

O presente trabalho foi desenvolvido utilizado a versão 2021.1.12f1 da Unity, disponível em https://unity3d.com/pt/get-unity/download. Como a Unity permite que o desenvolvedor possa importar pacotes gráficos através da Asset Store, alguns dos componentes gráficos deste jogo foram importados da própria loja oficial da Unity, mas também foram utilizados pacotes vindos do site https://opengameart.org.

A Unity utiliza C# como sua principal linguagem de programação, esta linguagem utiliza paradigmas orientada a objetos, entretanto ela disponibiliza uma API que refina a estrutura básica contida na linguagem C#, possibilitando o uso do paradigma orientado a componentes.

Dentro da plataforma da Unity todos os componentes devem herdar de um MonoBehaviour, sendo ele o responsável por implementar as comunicações entre os componentes ou métodos e inserir o componente dentro do ciclo de vida de Unity. Para cada um dos métodos a Unity percorre uma lista de GameObjects e realiza a chama dos respectivos métodos, por padrão, ao criar um Script, a Unity já cria os métodos bases, que são: Start e Update. O primeiro deles é chamado apenas uma vez, ao iniciar o Script, já o segundo é chamado uma vez a cada ciclo.

A escolha da Unity como plataforma de desenvolvimento do jogo Retraining foi feita pois a interface é muito fácil de usar, organizada, intuitiva e além de que o usuário pode customizar da forma que preferir. A utilização do C# como linguagem de programação foi outro fator que pesou a favor de escolher a Unity como plataforma, já que se assemelha muito ao Java, linguagem que o presente autor domina. A comunidade da Unity foi um fator que foi levado muito em consideração, já que possui muitos fóruns, documentações e tutoriais feito por terceiros, além de tutoriais criados pela própria Unity Technologies.

Dreamlo

Para o sistema de ranking, foi utilizado o Plug-in Dreamlo, que usa uma solicitação HTTP GET para armazenar o nome e a pontuação dos usuários que obtiveram maior quantidade de experiência. Com o Dreamlo foi possível mostrar na interface da Unity o ranking atual de players e atualizá-los. http://dreamlo.com.

Itch.io

É um site para que desenvolvedores "Indie" possam hospedar seus jogos e assim disponibilizar para um maior número de jogadores. Foi feita a build do jogo e exportado para o site Itch.io, e foi através da Itch.io que o jogo Retraining estará disponível para o público geral.

4.5 OPERACIONALIDADE

Nesta sessão é apresentado o funcionamento do jogo Retraining: Uma Solução computacional para apoiar o ensino de especificação de requisitos.

O jogo pode ser acessado em: https://rafaeldamasco.itch.io/retraining. Código fonte: https://github.com/RafaelDamasco/Retraining

Para uma melhor compreensão da aplicação, o jogo foi dividido em duas partes, a primeira contém informações sobre a navegação antes do início do jogo e a segunda apresenta o funcionamento do jogo.

4.5.1 Retraining – Navegação

Ao abrir o jogo, o usuário é apresentado a tela de menu inicial (Figura 18). Ao selecionar a opção "Sair" o jogo é encerrado. Ao selecionar a opção "Iniciar Jogo" a aplicação exibirá informação a respeito do funcionamento do jogo, bem como uma pequena história do dia a dia de um estagiário em uma empresa qualquer (Figura 19). Caso o não deseje avançar no jogo ele pode voltar para o menu inicial e encerrar a aplicação.

Figura 18 – Menu Inicial

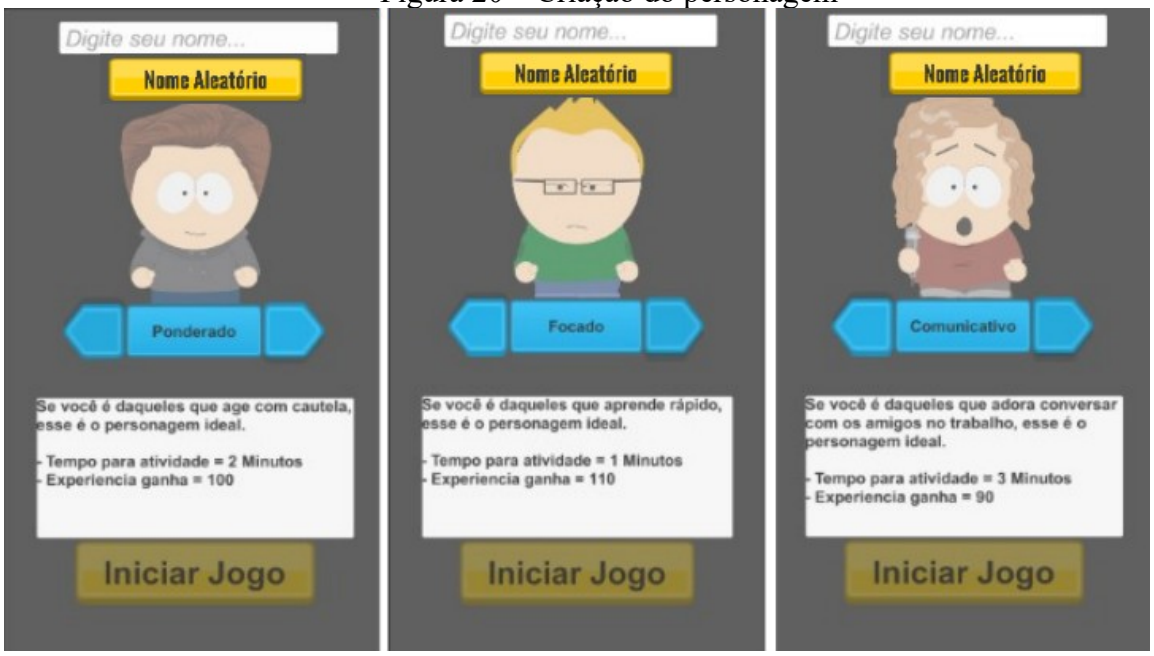
Fonte: (Própria, 2021)

Figura 19 – Apresentação Jogo

Fonte: (Própria, 2021)

Caso o usuário selecione a opção “Criar Personagem” ele poderá navegar entre três opções de personagem, cada um deles contará com uma aparência diferente e que graficamente se assemelha a uma personalidade. As personalidades disponíveis são: Comunicativo, Focado e Ponderado, cada uma delas trará um bônus e um ônus diferente ao usuário. Para habilitar o botão de “Iniciar Jogo” o usuário primeiramente deverá digitar o nome que deseja dar para o personagem ou clicar na opção de nome aleatório. A funcionalidade de gerar nome aleatório foi inserida no jogo, para que seja possível jogar em dispositivos mobile, já que não estava sendo possível usar o teclado dos aparelhos como ferramenta de entrada de informação (Figura 20).

Figura 20 – Criação do personagem



Fonte: (Própria, 2021)

4.5.2 Retraining – Jogo

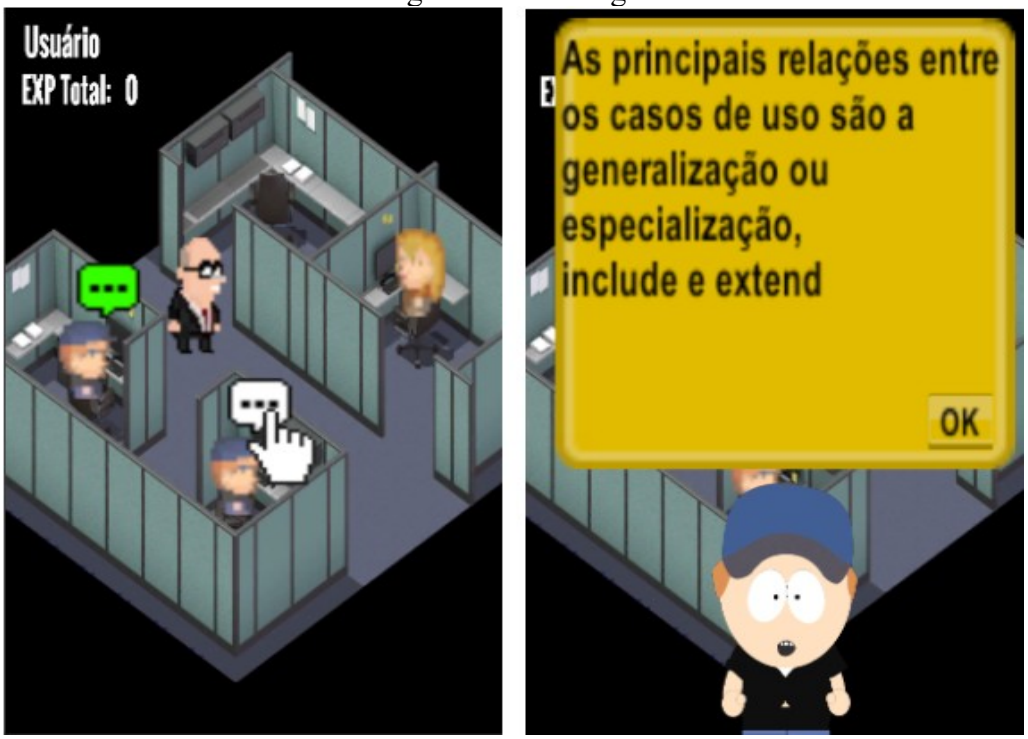
Antes de entrar em detalhes do jogo, é descrito um pouco da parte gráfica do início do jogo (Figura 21)

Figura 21 – Início Jogo

Fonte: (Própria, 2021)

O jogo se passa em um escritório, nele pode-se observar quatro personagens, sendo eles: o personagem escolhido anteriormente e três npcs (non-player character ou personagem não jogável). O primeiro deles da direita pra esquerda é o personagem que o usuário escolheu. O segundo é um npc que possui um balão branco acima da cabeça, se o usuário leu as informações a respeito do funcionamento do jogo nas telas passadas saberá que aquele npc não é obrigatório para o fluxo do jogo (Figura 22). Logo ao lado observa-se um npc muito semelhante ao anterior, porém com um balão verde acima da cabeça, como descrito anteriormente esse npc será responsável por aplicar o treinamento ao usuário. Mais a cima temos um outro npc um pouco diferente dos demais, ele parece um pouco mais velho, um pouco mais experiente e é ele o responsável por apresentar ao usuário um projeto em que a empresa precisará se dedicar.

Figura 22 – Interagir



Fonte: (Própria, 2021)

O fluxo do jogo é contínuo e feito de forma sequencial. Isso quer dizer que o usuário primeiramente passa por um período de treinamento sobre um conteúdo (Figura 23), e caso obtenha acertos maior que a metade das questões, ele estará apto a executar o projeto daquele treinamento.

Figura 23 – Realizar Treinamento

Fonte: (Própria, 2021)

O projeto será introduzido logo após o usuário completar o primeiro período de treinamento. A problemática abordada no projeto trará a tona a realidade de uma empresa que pretende gerenciar um estacionamento através de um software. Como dito anteriormente o npc “chefe” que disponibilizará o projeto para o usuário, surgirá um balão amarelo contendo um “ ! “ (Figura 24).

Será apresentada uma tela ao jogador mostrando o projeto que ele deverá cumprir. Nesse momento o chefe informa que o usuário será avaliado, quanto tempo ele tem para responder todas as perguntas e quanto de experiência ele ganha ao responder de forma correta uma pergunta, Todas essas variáveis foram definidas na hora em que o usuário escolheu a personalidade para o seu personagem. Ao responder todas as perguntas ou se o tempo se

esgotar são exibidos a porcentagem de acerto das questões e quanto de experiência o jogador obteve naquele projeto (Figura 24).

Figura 24 – Projeto

Fonte: (Própria, 2021)

Este ciclo de treinamento seguido de projetos se repete até o projeto três, quando é exibida uma tela de ranking contendo o nome e a experiência dos cinco jogadores que mais obtiveram experiência (Figura 25).

Figura 25 – Ranking

Fonte: (Própria, 2021)

Todas as questões, tanto as do período de treinamento, quanto as do projeto estão contidas no apêndice A e B, respectivamente.

5 AVALIAÇÃO

Este capítulo tem a finalidade de descrever como foi feita a avaliação do jogo proposto.

5.1 PLANEJAMENTO

Após a finalização do jogo Retraining, foi feita uma avaliação entre alunos selecionados nas áreas de tecnologia, para tal, decidiu-se pelo modelo de avaliação MEEGA+.

O MEEGA+ é um modelo sistemático para a avaliação de jogos educacionais voltados para o ensino de computação, este modelo permite avaliar a qualidade, eficiência e eficácia do presente jogo (SAVI, 2011). Segundo as análises do instrumento de medição do modelo chegou-se a um nível de confiabilidade aceitável para a avaliação da qualidade de jogos usados para o ensino de computação (PETRI, 2020).

O modelo utiliza a abordagem GQM (Goal-Question-Metric) para obter resultados das seguintes variáveis: atenção focada, diversão, desafio, interação social, confiança, relevância, satisfação e usabilidade (SAVI, 2011). O modelo inicia com a coleta de informações demográficas do usuário e segue para o questionário a respeito do jogo (PETRI, 2020).

O objetivo da avaliação foi de obter e atender os vieses anteriormente destacados em avaliar o jogo em questão utilizando o modelo MEEGA+. Bem como avaliar a eficácia do jogo como ferramenta complementar de ensino.

5.2 EXECUÇÃO

Em virtude da pandemia do Covid-19 o questionário de avaliação foi aplicado de maneira virtual, em que foram selecionados cinco alunos que já cursaram a disciplina de Engenharia de Software e aplicado o questionário do modelo de avaliação de jogos educativos MEEGA+ (Anexo A).

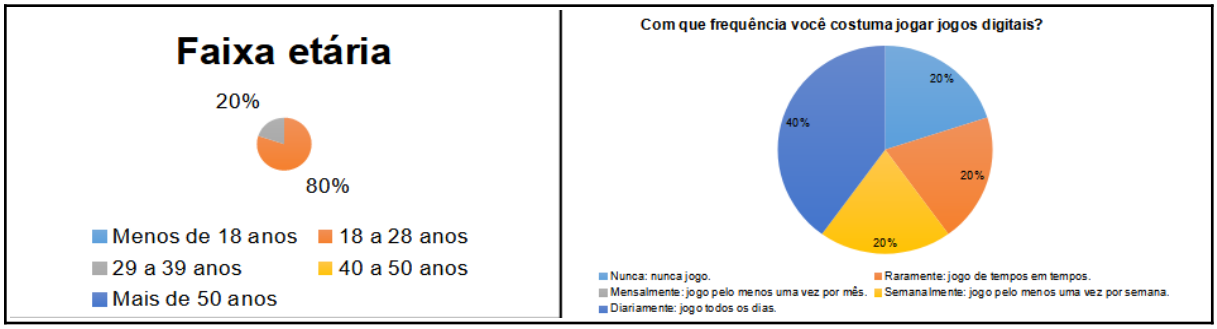
Ao avaliar o jogo a intenção foi medir se o método de ensino proposto seria complementar ao método de ensino tradicional ou não, analisando se de fato o jogo computacional proposto agregaria conhecimento para o jogador de forma lúdica, melhorando assim o aprendizado do aluno.

5.3 RESULTADOS

Esta seção destina-se a apresentar os resultados das avaliações citadas na seção anterior utilizando-se do modelo MEEGA+. O questionário é dividido em 3 partes: informações demográficas, experiência do jogador e percepção da aprendizagem.

No que diz respeito a informações demográficas, a amostra é predominantemente masculina, composta por 80% de homens. A faixa etária entre os intervalos de 18 a 28 anos corresponde a 80% da amostra. Quanto a frequência em que a amostra costuma jogar jogos digitais, verificou-se que 40% jogam todos os dias, 20% da amostra nunca, raramente ou semanalmente tem contato com jogos eletrônicos e ninguém respondeu jogar mensalmente. No que diz respeito a jogos não digitais, 40% dos avaliados raramente os jogam, essa mesma porcentagem respondeu ter contato com jogos não digitais mensalmente e 20% responderam ter contato com este tipo de jogo semanalmente.

Figura 26 – Informações Demográficas



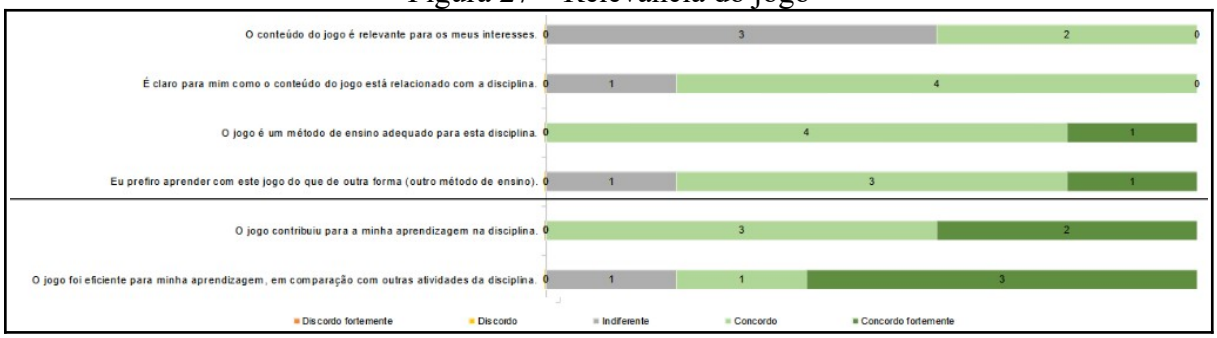
Fonte: (Própria, 2021)

Dividiu-se os aspectos avaliados no questionário em dois grupos, o primeiro informa sobre a relevância do jogo para o ensino, já o segundo aborda temas como satisfação, diversão, interação social, desafio, atenção focada e usabilidade do jogo.

Sobre a relevância do jogo como método de ensino obteve-se resultados positivos, todos os entrevistados afirmaram que o jogo é um método de ensino adequado para a disciplina de engenharia de requisitos (Figura 26). A percepção de aprendizagem do jogo foi outro aspecto que agradou aos usuários, a grande maioria respondeu que o jogo contribui para a aprendizagem na disciplina.

Analisando-se as outras temáticas abordadas no questionário pode-se observar que em relação a variável desafio abordado no jogo, os avaliados informaram em sua maioria que o jogo era adequado e desafiador para eles. A satisfação dos jogadores ao jogar o jogo foi descrita como boa pela ampla maioria, e, a interação social foi o aspecto que menos agradou aos jogadores, haja vista que o jogo não tinha nenhuma interação social. O elemento abordado no questionário relacionado a diversão foi o que mais agradou aos usuários, todos eles responderam que o jogo os fez sorrir. A atenção focada aplicada ao jogo pelos jogadores foi algo que não obteve bom resultado, sendo irrelevante pela maioria. Sobre sua usabilidade, pode-se dizer que foi satisfatória, menos em relação aos tamanhos de fonte e estilo das letras nos textos do jogo.

Figura 27 – Relevância do jogo



Fonte: (Própria, 2021)

Vale ressaltar que o questionário foi aplicado em um universo restrito de apenas cinco alunos, portanto trata-se de uma avaliação ainda inicial acerca do jogo e que não comprova a sua real eficiência para o ensino. Porém, foi possível observar através das

respostas do questionário que há indícios de que o jogo pode ser uma opção como ferramenta complementar no ensino de especificação de requisitos.

6 CONSIDERAÇÕES FINAIS

A Engenharia de Requisitos é uma etapa fundamental para auxiliar na identificação, análise, documentação e gerenciamento de requisitos de um software, tendo em vista sua importância significativa para a elaboração de um projeto, o presente trabalho propôs uma sugestão de auxílio no processo de criação de um software.

A escolha do tema foi pautada na dificuldade de cumprimento de prazos e orçamentos dos projetos de software apresentados atualmente, pois mesmo com diversas técnicas e métodos de levantamento para melhorias na aplicação as falhas nos requisitos podem tornar sua execução difícil e cara.

Com base nisso, observa-se como a dificuldade apresentada pelos alunos no processo de aprendizagem de Engenharia de Requisitos reflete na ausência ou errônea aplicação em projetos. Segundo Glasser (2001), parte da defasagem do estudo de Engenharia de Requisitos provém do método de ensino – pautado apenas no teórico, e, devido à falta de compreensão de sua real aplicação durante o processo de desenvolvimento, os alunos não dão a devida importância à elicitação e análise de requisitos. Logo, a intenção do presente trabalho propôs a criação de um software para apoiar o ensinamento de técnicas de levantamento e análise de requisitos de forma mais ativa, visando atingir principalmente o público em âmbito universitário e em processo de aprendizagem de Engenharia de Requisitos.

A antelação do tema, provém de sua importância no meio acadêmico, pois através da aprendizagem ativa por meio do jogo computacional proposto com a análise de abordagens para o ensino de Engenharia de Requisitos, visa auxiliar e ensinar a documentação e especificação de requisitos de software, e, assim, diminuir as falhas e dificuldades de execução em futuros projetos.

A ferramenta de ensino computacional Retraining tem como finalidade ajudar os alunos a entender a matéria de Elicitação de Requisitos, e através da aplicação do questionário MEEGA+, foi possível identificar a partir das respostas, indícios de cumprimento dos objetivos propostos pelo jogo Retraining. A partir deste modelo sistemático de avaliação no universo em que foi aplicado, foi possível identificar que as respostas indicaram haver eficácia do jogo como ferramenta complementar de ensino, e através dos dados obtidos entre

os alunos selecionados nas áreas de tecnologia o resultado foi satisfatório, e o jogo cumpriu assim, os objetivos propostos neste presente trabalho.

REFERÊNCIAS

APPOLINÁRIO, Fabio. **Dicionário de Metodologia Científica**. 2. ed. São Paulo: Atlas, 2011. 295p.

BARBOSA, E. F. & MOURA, D. G. **Metodologias ativas de aprendizagem na Educação Profissional e Tecnológica**. B. Tec. Senac, Rio de Janeiro, v. 39, n.2, p.48-67, maio/ago. 2013.

BEZERRA, Eduardo. **Princípios de Análise e Projeto de Sistemas com UML.** Rio de Janeiro: Campus, 2002.

BOEHM, B.W. **Software risk management.** IEEE Computer Society Press: Washington, 1989.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. UML **Guia do Usuário**. Rio de Janeiro: Campus, 2000.

C. J. Neill and P. A. Laplante. **Requirements engineering: The state of the practice.** IEEE Software, vol 20(6):40–45, 2003.

COPLE, Denis. **Desenvolvimento De Jogos II**. 1ª Edição Rio de Janeiro: SESES, 2019.

DAVIS, A. M. **Software Requirements: Objects, Functions and States.** Englewood Cliffs, NJ: Prentice Hall, 1993.

ENGHOLM JÚNIOR, H. **Engenharia de Software na prática**. São Paulo: Novatec, 2010.

ESTEVES, Rodrigo. Brainstorm: **Como gerar ideias com mais eficiência**. Dash Editora, 2020.

FLYNT, John P. **Software Engineering for Game Developers.** Boston: Thomson Course Technology, 2005.

G. PETRI, C. G. VON WANGENHEIM, AND A. F. BORGATTO. **Meega+: Um modelo para a avaliac¸ao de jogos educacionais para o ensino de computação** Revista Brasileira de Informática na Educação27(03):52, 2020.

GIL, A. C**. Métodos e técnicas de pesquisa social**. 6 ed. São Paulo. 2008.

GLASSER, W. **Teoria da Escolha: uma nova psicologia de liberdade pessoal.** São Paulo: Mercuryo, 2001.

GUEDES, Gilleanes T. A. UML: **Uma abordagem prática**. São Paulo: Novatec, 2006.

H. Erdogmus, N. Medvidović and F. Paulisch, "**50 Years of Software Engineering,**" in *IEEE Software*, vol. 35, no. 5, pp. 20-24, September/October 2018. doi: 10.1109/MS.2018.3571240

IEEE Std. 12333-1996 IEEE **Guide for Developing System Requirements Specification.** The Institute of Electrical and Electronics Engineers. Piscataway, NJ. Guide, EUA, 1996.

ISO/IEC/IEEE 29148:2018. **Systems and Software Engineering Life Cycle Processes Requirements Engineering.** The International Organization for Standardization (ISO)/The International Electrotechnical Commission (IEC)/The Institute of Electrical and Electronics Engineers (IEEE) Computer Society. 2018.

LEITE, J. C. **Notas de Aula de Engenharia de software.** Natal, RN, 2000.

LEITE, J.C.S.P.; LEONARDI, M.C. (1998). **Business Rules as organizational policies.** In: Proceedings of the 9th International Workshop on Software Specification & Design. ISE-Shima, Japan. 1ed. USA: IEEE CSP, Los Alamitos, CA. P. 68-76, Apr.

LIMA, A. D. S. **UML 2.0: do requisito à solução.** 2. ed. São Paulo: Érica, 2007.

LUQUE, L. **Engenharia de Software II**, 2017.

MAGELA, Rogério. **Engenharia de Software Aplicada – Princípios –** 1º. Ed. – Alta Books, 2006.

MORAN, José Manuel. **Tecnologias digitais para uma aprendizagem ativa e inovadora**, Campinas, SP

MORAN, José Manuel. **A educação que desejamos: novos desafios e como chegar lá**. 2. ed. Campinas, SP: Papirus, 2007. 174p.

PERUCIO, A. S.; BERTHÊM, A. C. de; BERTSCHINGER, G. L.; MENEZES, R. R. C. **Desenvolvimento de jogos eletrônicos: teoria e prática.** 2.ª ed. São Paulo: Novatec, 2007.

PFLEEGER, S. L. **Engenharia de software: teoria e prática.** 2.ª ed. São Paulo: Prentice Hall, 2004.

PRESSMAN, R.S. **Engenharia de software.** Pearson Makron Books, 2006.

POHL, Klaus. **Requirements engineering: fundamentals, principles, and techniques.** Springer Publishing Company, Incorporated, 2010.

ROGERS, Scott. **Level Up! The guide to great video game design.** John Wiley & Sons, 2014.

SAVI, R., GRESSE VON WANGENHEIM, C., E BORGATTO, A. (2011). **Um Modelo de Avaliação de Jogos Educacionais na Engenharia de Software**. Anais do XXV Simpósio Brasileiro de Engenharia de Software (SBES 2011), pp. 194-203.

SEVERINO, Antonio Joaquim. **Metodologia do Trabalho Científico.** São Paulo: Cortez, 2007

SHANE H, S.W. **Standish group 2015 Chaos Report - Q&A**. October 2015.

SILVA, E. L., MENEZES, E. M. **Metodologia da pesquisa e elaboração de dissertação**. 3.

ed. rev. atual., Laboratório de Ensino a Distância. da UFSC, Florianópolis, Santa Catarina. 2001.

SOMMERVILLE, I. **Engenharia de Software**. 6ª. Ed. São Paulo: PEARSON Addison Wesley, 2005.

SOMMERVILLE, I. **Engenharia de Software.** 9ª. Ed. São Paulo: PEARSON Addison Wesley, 2011.

SOMMERVILLE, I. and KOTONYA, G. (1998) **Requirements Engineering: Processes and Techniques.** John Wiley & Sons, Inc., Hoboken.

VIZCAÍNO, A; PIATTINI, M.; CABALLERO, I.; MONASOR, M. J. **Preparing Students and Engineers for Global Software Development**: A Systematic Review. IEEE International Conference On Global Software Engineering. Annals of..., Princeton, 2010.

VAZQUEZ, Carlos Eduardo; SIMÕES, Guilherme Siqueira. **Engenharia de Requisitos: Software Orientado ao Negócio.** Rio de Janeiro: Brasport, 2016.

YOUNG, R. **The Requirements Engineering handbook**. Artech House, 2004.

APÊNDICE A – PERGUNTAS E RESPOSTAS TREINAMENTO

Treinamento parte 1

O treinamento desta fase será relacionado com o tema de requisitos em linguagem natural.

Objetivos de aprendizagem:

- Compreender o que é um requisito de software.
- Diferenciar requisitos funcionais e não funcionais.
- Avaliar a qualidade de um requisito.

1. Sobre a definição de um requisito marque verdadeiro ou falso.

a) Requisito é o que o sistema tem que ter para atender plenamente ao propósito para o qual foi criado.
b) Uma especificação do que deve ser construído. São descrições de como o sistema deve se comportar, ou uma propriedade ou atributo do sistema.
c) Uma necessidade do cliente ou usuário.
d) Objetivos ou restrições estabelecidas por clientes e usuários do sistema que definem as diversas propriedades do sistema.

Alternativas:
A) VVFV B) FFVV **C) VVVV** D)FVVV

2. Quais são os benefícios da Engenharia de Requisitos?

a) Estabelecer e manter concordância com os clientes e outros investidores sobre o que o sistema deve fazer.
b) Definir os limites do sistema.

c) Fornecer uma base para estimar o custo e o tempo de desenvolvimento do sistema.
d) Fornecer uma base para planejar o conteúdo técnico das iterações.

Alternativas:
A) VVFV B) FFVV **C) VVVV** D)FVVV

3. Assinale a alternativa incorreta em relação aos requisitos funcionais

a) São as declarações das funções que o sistema deve oferecer.
b) Estão ligado diretamente com a funcionalidade do software.
c) Um requisito funcional é um requisito relacionado ao resultado de algum comportamento a ser fornecido por uma função do sistema.
d) São facilmente visualizados com a ajuda de diagramas, principalmente diagramas de atividade.

4. Esses são exemplos de requisitos funcionais, exceto:

a) RF01 O sistema deve fazer a persistência de dados através do banco de dados loca SQLite.
b) RF02 O sistema deve possibilitar o logout dos usuários que estão logados.
c) RF03 O sistema deve permitir ao usuário cadastrar produtos.
d) RF04 O sistema deve permitir ao usuário emitir um relatório de produtos.

5. Assinale a alternativa incorreta em relação aos requisitos não funcionais.

a) Eles servem para restringir os serviços ou funções do sistema.
b) Os requisitos não funcionais se preocupam com o sistema por completo.
c) Tem relação com as funcionalidades que o sistema deve oferecer.
d) Podem ser relacionados a usabilidade, desempenho e segurança.

6. Quais são os atributos do sistema que são englobados pelos requisitos não funcionais.

   a) Usabilidade.
   b) Desempenho.
   c) Segurança.
   d) Hardware.

   Alternativas:
   A) VVFV B) FFVV **C) VVVV** D)FVVV

7. São exemplos de requisitos não funcionais, exceto:

   a) RNF01 A interface gráfica será a disponibilizada pelo Java-Swing.
   b) RNF02 O sistema deve fazer a persistência de dados através do banco de dados local SQLite.
   c) RNF03 O sistema deve ser implementado na linguagem de programação Java na versão 8.
   d) RNF04 O sistema deve possibilitar o logout dos usuários que estão logados.

8. Em relação à regra de negócio marque verdadeiro ou falso.

   a) As regras de negócio são restrições/premissas necessárias para o negócio.
   b) Requisitos que se originam do domínio da aplicação do sistema e que refletem características desse domínio.
   c) Não possuem dependência com um requisito funcional.
   d) Refere-se à o que o sistema deverá fazer.

Alternativas:

A) VVFV B) FFVV **C) VVFF** D)FVVV

9. São exemplos de regras de negócio aplicadas a um sistema de uma universidade:

   a) O aluno não pode ultrapassar 35 horas-aula semanal
   b) A aprovação em uma disciplina é obtida com média final maior ou igual a 6.0 e 75% de presença.
   c) Para se matricular em uma disciplina é necessário ter feito todos os pré requisitos.
   d) Não pode haver choque de horário com outra disciplina.

   Alternativas:

   A) VVFV B) FFVV **C) VVVV** D)FVVV

10. Durante a validação de um requisito, devemos verificar sua:

    a) Verificações de validade: O sistema deve fornecer as funções que melhor atende às necessidades do usuário.
    b) Verificações de consistência: Requisitos no documento não devem entrar em conflito. Não deve haver restrições contraditórias
    c) Verificações de completude: O documento de requisitos deve incluir requisitos que definem todas as funções e restrições.
    d) Verificações de realismo: Os requisitos devem ser verificados para assegurar que podem ser implementados.

    Alternativas:

    A) VVFV B) FFVV **C) VVVV** D)FVVV

Treinamento parte 2

O treinamento desta fase será relacionado ao tema das relações entre os casos de uso generalização ou especialização, include e extend.

Objetivos de aprendizagem:

- Compreender os elementos do diagrama de casos de uso.
- Diferenciar os relacionamentos que podem existir no diagrama de casos de uso.

1. O diagrama de casos de uso não entra em detalhes sobre:

a) Descrever quem faz o quê no sistema..
b) As sequências de passos necessárias ou possíveis para prover uma resposta ao evento que o motivou.
c) As regras de negócio que se aplicam.
d) A sequência de casos de uso que deve ser realizada para se cumprir um processo de negócio.

Alternativas:

A) VVFV B) FFVV C) VVVV **D)FVVV**

2. O diagrama de casos de uso pode ser composto por:

a) Atores.
b) Casos de uso.
c) Associação do tipo Include, Extend e Generalização.
d) Nodo Decisão.

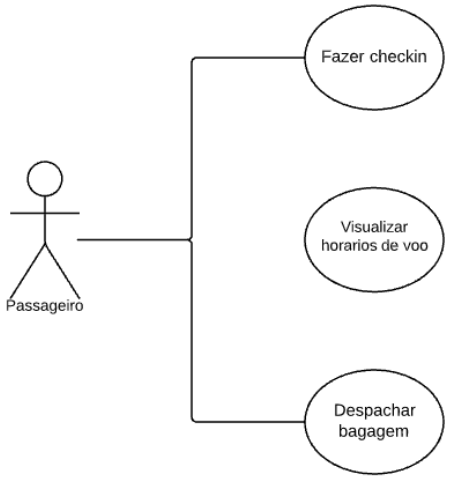
Alternativas:

A) VVVF B) FFVV C) VVVV D)FVVV

3. Um ator pode corresponder há:

   a) Hardware.
   b) Organização.
   c) Outro sistema.
   d) Todas as anteriores inconsistência.

4. A representação do diagrama de casos de uso abaixo está:



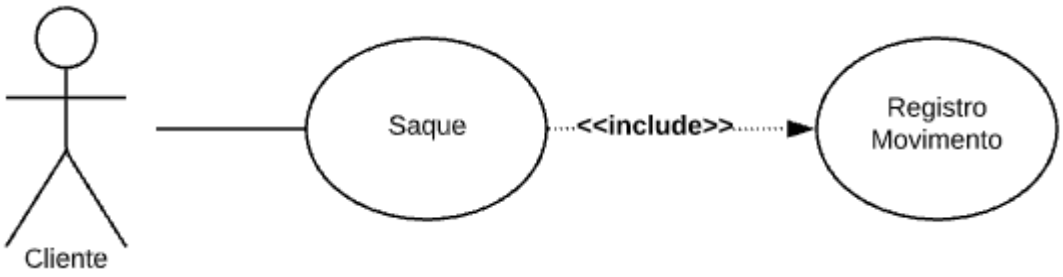
   A) Correto **B) Errado**

5. Sobre as relações entre os casos de uso marque a alternativa correta.

   a) A generalização tem a função de estabelecer uma relação de especialização apenas entre atores.
   b) A inclusão de um caso de uso pelo outro, implica que o comportamento do caso de uso base é incorporado pelo caso de uso incluído. É uma relação de não-obrigatoriedade.
   c) **O extend significa que um caso de uso pode estender outro**.

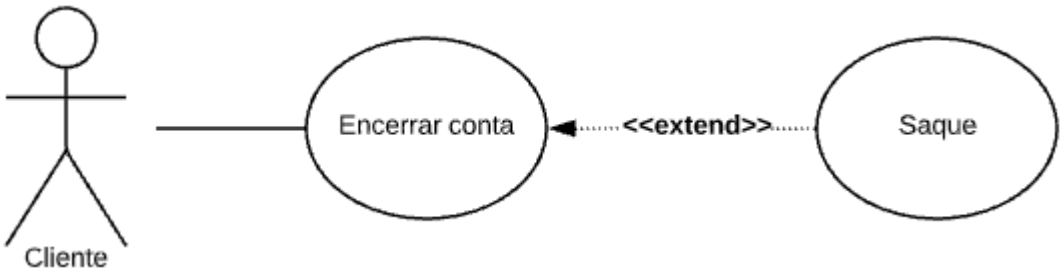
d) A generalização tem a função de estabelecer uma relação de especialização apenas entre casos de uso.

6. No exemplo abaixo o caso de uso “Registro de Movimento” será executado sempre que o caso de uso “Saque” for executado?



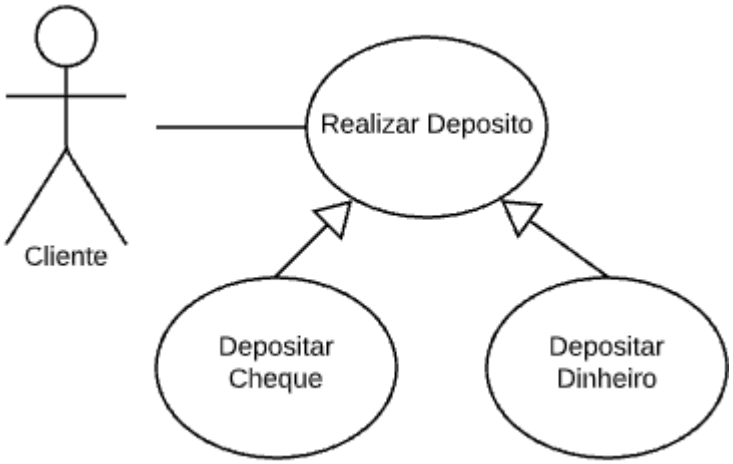
A) Correto B) Errado

7. No exemplo abaixo o caso de uso “Saque” será executado sempre que o caso de uso “Encerrar Conta” for executado?



A) Correto **B) Errado**

8. A imagem abaixo representa:



a) Generalização de atores.
b) Generalização de casos de uso.
c) Include.
d) Extend.

Treinamento parte 3

O treinamento desta fase será relacionado ao tema de descrição de cenários e diagrama de atividade como forma de detalhar o comportamento de um caso de uso.

Objetivos de aprendizagem:

- Compreender os elementos do diagrama de atividade.
- Compreender o que são cenários e como descrevê-los.
- Identificar cenários principais, alternativos e de exceção.

1. Detalhamento de casos de uso através de cenários podem ser descritos das seguintes formas:

a) Descrição contínua: a descrição desse formato é feita utilizando-se um texto livre ou casual.
b) Descrição enumerada: esse formato é descrito por uma sequência de passos enumerados.
c) Descrição particionada: essa representação divide-se em duas colunas, uma coluna representa a sequência de interações dos atores e a outra as interações do sistema.
d) Uma descrição pode ser composta pelos seguintes cenários: Cenário Principal, Cenário Alternativo e Cenário de Exceção.

Alternativas:
A) VVFV B) VFVV C) VVVF **D)VVVV**

2. Sobre a descrição de um cenário assinale a alternativa incorreta:

a) A descrição de um cenário explora como e quando um caso de uso começa.
b) A descrição de um cenário explora quando o caso de uso interage com os atores e quais dados eles trocam entre si.
c) Aborda aspectos sobre um requisito não funcional.
d) Aborda aspectos sobre um requisito funcional.

3. A realização de um caso de uso requer que alguma lógica computacional seja aplicada, qual diagrama da UML detalha esta lógica de forma clara e sucinta?

a) Diagrama de casos de uso.
b) Diagrama de classe.
c) Diagrama de atividade.
d) Diagrama de estado.

4. O diagrama de atividade pode ser utilizado com o objetivo de refinar um caso de uso, sua função é de representar os estados de uma atividade, tendo em vista que a

utilização do diagrama de casos de uso não se dispõem a detalhar o comportamento da atividade

A) Correto B) Errado

APÊNDICE B – PERGUNTAS E RESPOSTAS PROJETO

Apresentação do projeto

Nossa equipe foi contratada para criar um software que gerencia um estacionamento, vamos começar com a especificação do sistema para posteriormente desenvolvermos o produto.

O software será responsável por controlar a entrada e saída de automóveis, fornecerá um gerenciamento personalizado das vagas do estacionamento, desde a disposição das vagas até os diferentes preços oferecidos.

- O manobrista será o único responsável por incluir os dados do automóvel no sistema, além de estacionar e retirar os automóveis.
- Não é necessário manter um cadastro dos clientes, tampouco dos automóveis
- O sistema deverá permitir que os preços das horas e lavação sejam alterados pelo manobrista. Além disso, é necessário que seja gerado uma lista contendo as informações dos automóveis estacionados (placa, hora de entrada e saída, se utilizou o serviço de lavação), a quantidade de automóveis e o lucro total. Esta lista deve ser gerada com as informações do dia.
- O sistema deverá funcionar sem a necessidade de instalação de um SGBD, pois o computador disponibilizado para o sistema possuirá apenas recursos básicos de hardware, e o sistema deve abrir rapidamente.
- O sistema deverá apresentar uma interface simples, com no máximo nove telas, onde as janelas de aviso não são contabilizadas. Além disso, as vagas devem ser diferenciadas através de cores, onde a cor vermelha indicará que a vaga está ocupada e verde, livre.
- O estacionamento possui vinte vagas, abre às 08:00 horas e fecha às 18:00 horas. Após as 18:00 horas nenhum automóvel pode entrar, assim como se o estacionamento estiver lotado. O preço do estacionamento é um pacote mínimo de 3 horas e para cada

hora adicional, é acrescentado um valor. Após às 18:00 horas o valor acrescido é mais alto.

- Os valores que virão pré cadastrados serão: 3 horas R$ 5,00; hora adicional antes das 18:00 horas R$ 2,00 e hora adicional depois das 18:00 horas R$ 6,00, pois são os preços cobrados até o dia da entrevista.
- O preço da lavação difere conforme o tipo de automóvel, bem como se é comum ou completa. Os tipos possíveis são hatch, sedan, SUV/pick up e van/furgão. Os preços pré cadastrados serão: Hatch Comum: R$ 10,00 / Completa: R$ 15,00; Sedan Comum: R$ 15,00 / Completa: R$ 22,00, SUV/Pickup Comum: R$ 20,00 / Completa: R$ 27,00 e Van/Furgão Comum: R$ 25,00 / Completa: R$ 37,00, pois também são os preços cobrados até o dia da entrevista.
- Até que o último veículo seja retirado, o manobrista deverá permanecer em seu posto. Portanto, o sistema não poderá ser fechado se algum automóvel estiver estacionado.

Projeto parte 1

1. O sistema deve permitir ao usuário manobrista registrar a entrada de um automóvel no estacionamento.

    a) Requisito Funcional
    b) Requisito Não Funcional
    c) Regra de Negócio
    d) Nenhuma das anteriores

2. O sistema deve permitir ao usuário manobrista registrar a saída de um automóvel do estacionamento.

    a) Requisito Funcional
    b) Requisito Não Funcional
    c) Regra de Negócio

d) Nenhuma das anteriores

3. O sistema deve permitir ao usuário manobrista registrar a lavação de um automóvel do estacionamento.

a) Requisito Funcional
b) Requisito Não Funcional
c) Regra de Negócio
d) Nenhuma das anteriores

4. O sistema deve permitir ao usuário manobrista alterar os preços de estacionamento e lavação dos automóveis.

a) Requisito Funcional
b) Requisito Não Funcional
c) Regra de Negócio
d) Nenhuma das anteriores

5. O sistema deve permitir ao usuário manobrista gerar uma lista diária com os automóveis estacionados e o total arrecadado.

a) Requisito Funcional
b) Requisito Não Funcional
c) Regra de Negócio
d) Nenhuma das anteriores

6. O sistema deve permitir ao usuário manobrista fechar o sistema.

a) Requisito Funcional
b) Requisito Não Funcional

c) Regra de Negócio
d) Nenhuma das anteriores

7. O sistema deve permitir ao usuário manobrista registrar o pagamento do estacionamento.

a) Requisito Funcional
b) Requisito Não Funcional
c) Regra de Negócio
d) Nenhuma das anteriores

8. O sistema deve ser implementado na linguagem de programação Java

a) Requisito Funcional
b) Requisito Não Funcional
c) Regra de Negócio
d) Nenhuma das anteriores

9. O sistema deve apresentar as vagas livres na cor verde e as vagas ocupadas na cor vermelha

a) Requisito Funcional
b) Requisito Não Funcional
c) Regra de Negócio
d) Nenhuma das anteriores

10. O sistema deverá ser compatível com a versão do Java 7.0.800

a) Requisito Funcional
b) Requisito Não Funcional

c) Regra de Negócio
d) Nenhuma das anteriores

11. O sistema deverá persistir os dados necessários através do recurso de serialização da linguagem Java.

a) Requisito Funcional
b) Requisito Não Funcional
c) Regra de Negócio
d) Nenhuma das anteriores

12. O sistema deverá apresentar no máximo nove telas de interface

a) Requisito Funcional
b) Requisito Não Funcional
c) Regra de Negócio
d) Nenhuma das anteriores

13. O sistema deverá iniciar em no máximo 5 segundos

a) Requisito Funcional
b) Requisito Não Funcional
c) Regra de Negócio
d) Nenhuma das anteriores

14. O estacionamento possui 20 vagas para automóveis.

a) Requisito Funcional
b) Requisito Não Funcional
c) Regra de Negócio

d) Nenhuma das anteriores

15. O estacionamento abre às 08:00 horas e fecha às 18:00 horas, após as 18:00 horas nenhum automóvel pode entrar.

a) Requisito Funcional
b) Requisito Não Funcional
c) Regra de Negócio
d) Nenhuma das anteriores

16. Se todas as vagas estiverem ocupadas, nenhum novo automóvel poderá entrar no estacionamento.

a) Requisito Funcional
b) Requisito Não Funcional
c) Regra de Negócio
d) Nenhuma das anteriores

17. O manobrista só poderá fechar o estacionamento se não houver nenhum automóvel estacionado.

a) Requisito Funcional
b) Requisito Não Funcional
c) Regra de Negócio
d) Nenhuma das anteriores

18. O preço do estacionamento é um pacote mínimo de 3 horas e para cada hora adicional, é acrescentado um valor.

a) Requisito Funcional

b) Requisito Não Funcional
c) Regra de Negócio
d) Nenhuma das anteriores

19. O preço da lavação difere conforme o tipo de automóvel, bem como se é comum ou completa.

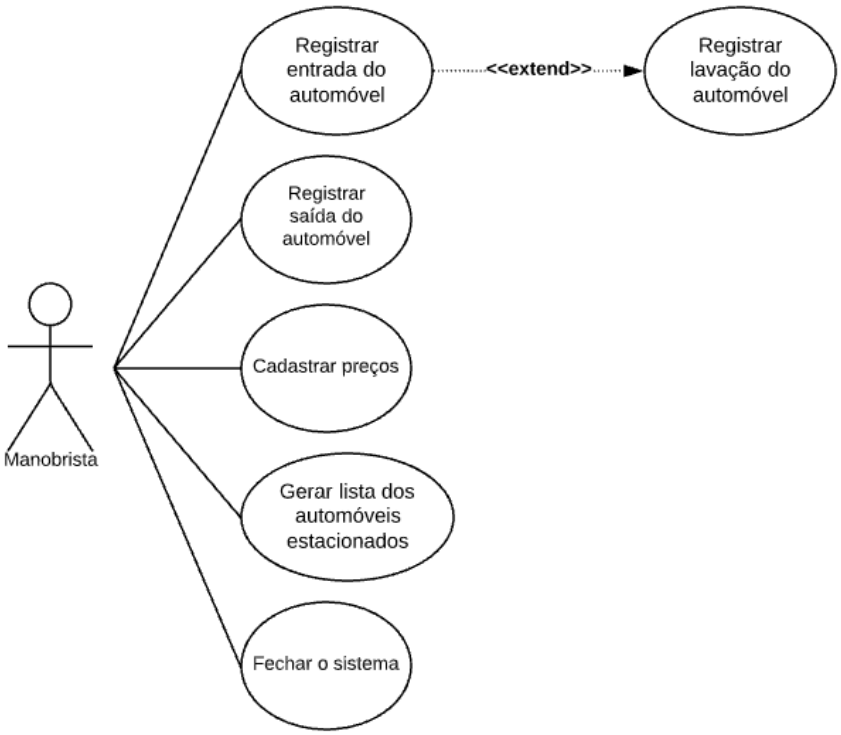
a) Requisito Funcional
b) Requisito Não Funcional
c) Regra de Negócio
d) Nenhuma das anteriores

20. Os automóveis são separados em quatro tipos: Hatch; Sedan; SUV/Pick up; Van/Furgão.

a) Requisito Funcional
b) Requisito Não Funcional
c) Regra de Negócio
d) Nenhuma das anteriores

Projeto parte 2

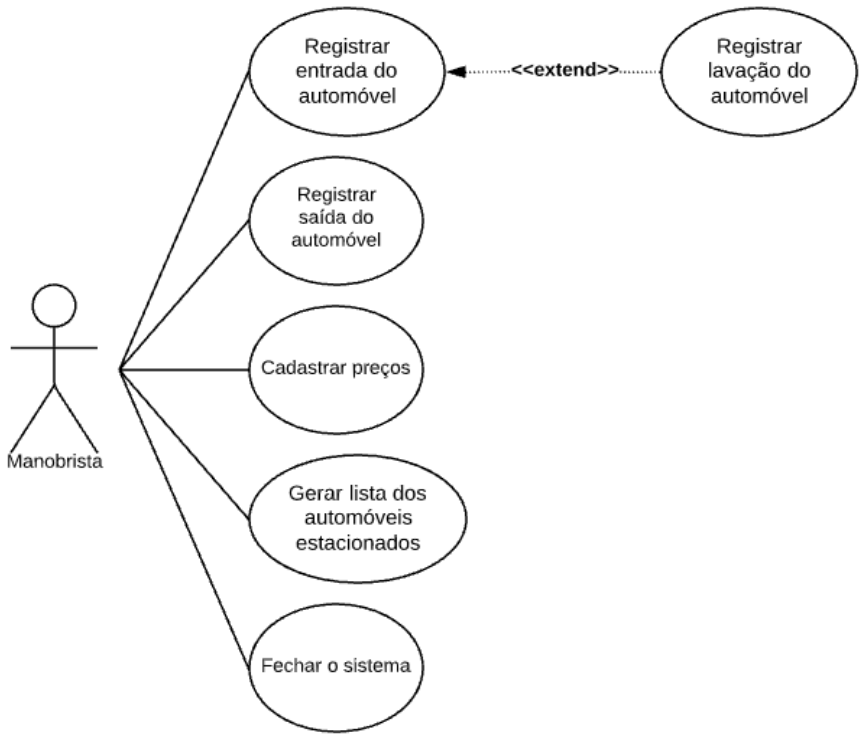
1. O diagrama de caso de uso do sistema de gerenciamento de carros abaixo está:



Alternativas:

A) Correto **B) Errado**

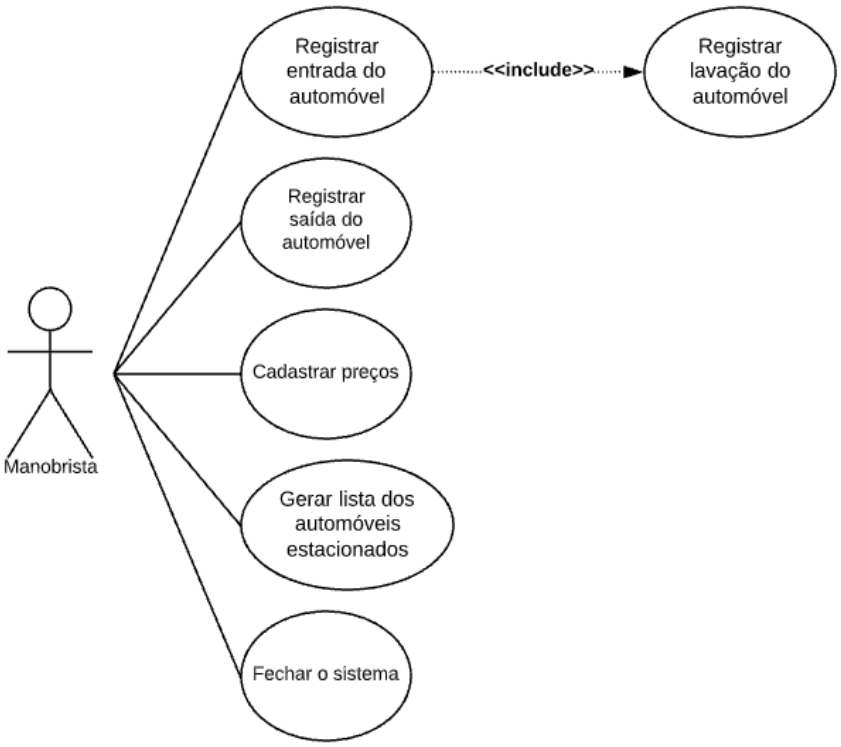
2. O diagrama de caso de uso do sistema de gerenciamento de carros abaixo está:



Alternativas:

A) Correto B) Errado

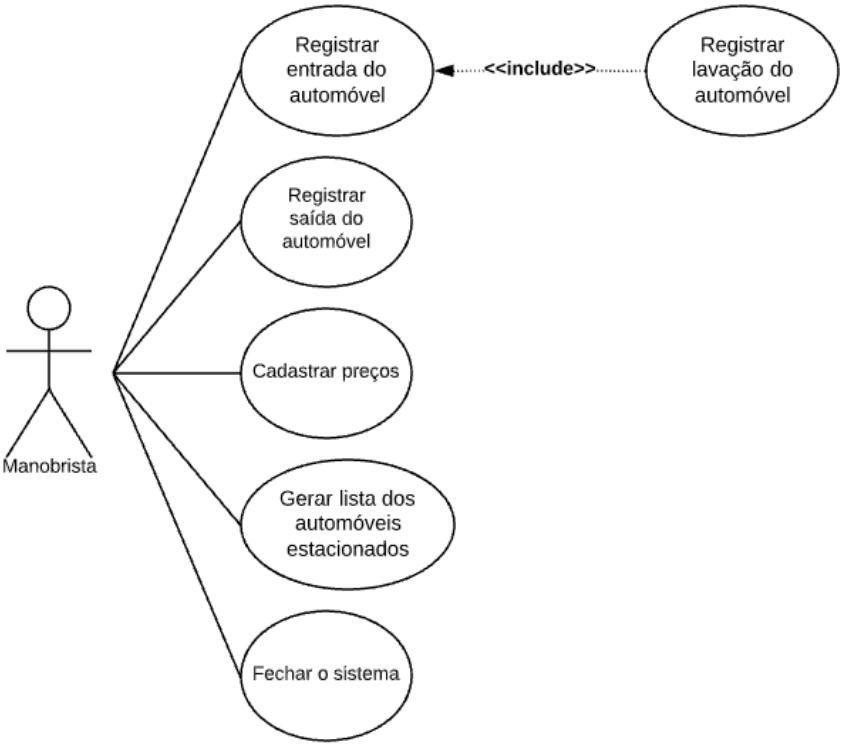
3. O diagrama de caso de uso do sistema de gerenciamento de carros abaixo está:



Alternativas:

A) Correto **B) Errado**

4. O diagrama de caso de uso do sistema de gerenciamento de carros abaixo está:



Alternativas:

A) Correto **B) Errado**

Projeto parte 3

1. A respeito do caso de uso “Cadastrar preço”. Ordene o cenário principal que detalha o caso de uso “Cadastrar Preço”:

Cenário Principal:

(1) Sistema verifica se os valores são válido

(2) Manobrista seleciona o tipo de automóvel

(3) Sistema apresenta tela para cadastro dos preços

(4) Manobrista insere os novos preços

(5) Manobrista solicita cadastrar preço
(6) Sistema salva os preços

Alternativas:
A) 5 4 2 3 1 6 **B) 5 3 2 4 1 6** C) 1 2 3 4 5 6 D) 5 4 3 2 1 6

2. Sobre o cenário principal ordenado anteriormente, onde se encaixa um possível cenário alternativo:
   Cenário Principal:
   (1) Manobrista solicita cadastrar preço
   (2) Sistema apresenta tela para cadastro dos preços
   (3) Manobrista seleciona o tipo de automóvel
   (4) Manobrista insere os novos preços
   (5) Sistema verifica se os valores são válido
   (6) Sistema salva os preços

   Alternativas:
   A) 1 **B) 3** C) 4 D) 5

3. Encontre o passo incorreto do Cenário Principal:

   (1) Manobrista solicita cadastrar preço.
   (2) Sistema apresenta tela para cadastro dos preços.
   (3) Manobrista seleciona o tipo de automóvel.
   (4) Manobrista insere os novos preços.
   (5) Sistema verifica se os valores são válido.
   (6) Sistema salva os preços.

   Cenário Alternativo:

(3.1) Sistema apresenta os tipos de veículos

(3.2) Manobrista seleciona o tipo de veículo

(3.3) Se o valor for inválido, apresenta mensagem “Valores inválido!”

(3.4) Retorna ao passo 3

Alternativas:

A) 3 B) 3.1 **C) 3.3** D) 3.4

4. Sobre o cenário principal ordenado anteriormente, onde se encaixa um possível cenário Exceção:

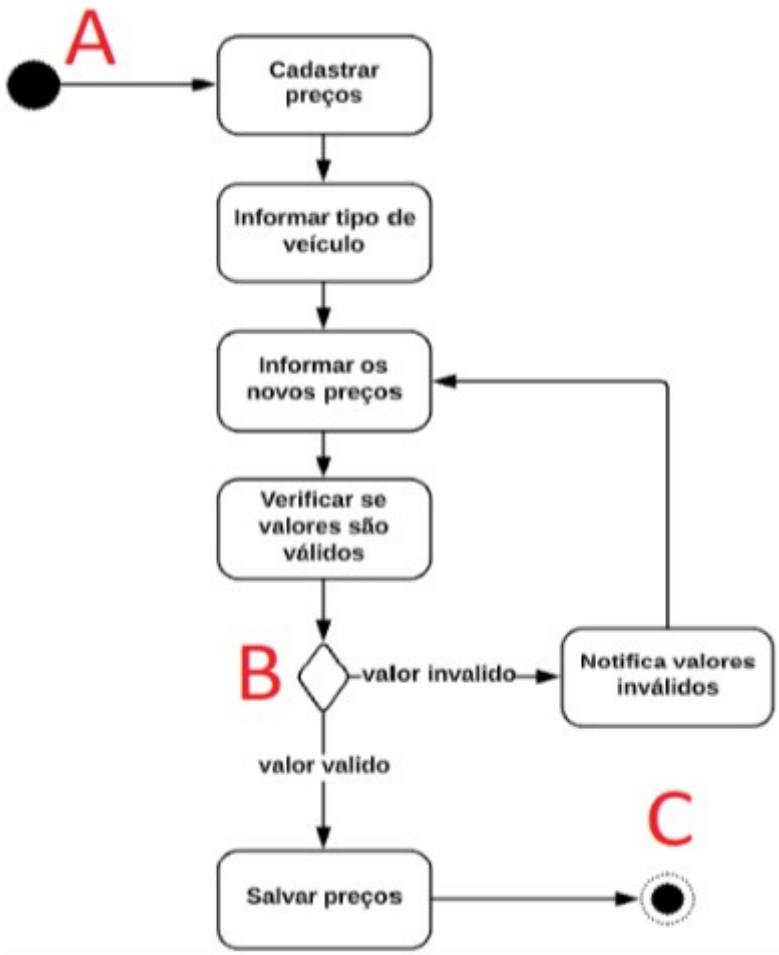
Cenário Principal:

(1) Manobrista solicita cadastrar preço.
(2) Sistema apresenta tela para cadastro dos preços.
(3) Manobrista seleciona o tipo de automóvel.
(4) Manobrista insere os novos preços.
(5) Sistema verifica se os valores são válido.
(6) Sistema salva os preços.

Alternativas:

A) 3 B) 4 **C) 5** D) 6

5. O diagrama de atividade que refina o caso de uso “Cadastrar Preço”, possui as alternativas



(1) [ ]

(2) [ ]

(3) [ ]

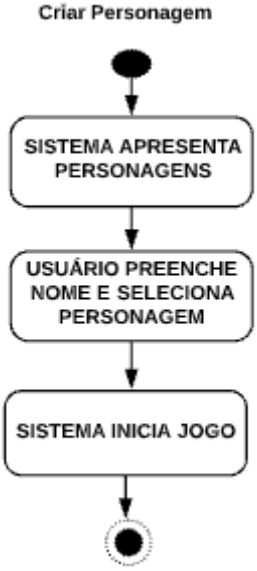
Alternativas:

A)VVF **B)VVV** C)VFV D)FVV

APÊNDICE C – DIAGRAMA DE ATIVIDADE RETRAINING

- Diagrama de atividade do caso de uso iniciar jogo

- Diagrama de atividade do caso de uso criar personagem



- Diagrama de atividade do caso de uso ver ranking

- Diagrama de atividade do caso de uso encerrar jogo

APÊNDICE D – ARTIGO SOBRE O PRESENTE TRABALHO

RETRAINING: UMA SOLUÇÃO COMPUTACIONAL PARA APOIAR O ENSINO DE ESPECIFICAÇÃO DE REQUISITOS

RETRAINING: A COMPUTATIONAL SOLUTION TO SUPPORT THE TEACHING OF REQUIREMENTS SPECIFICATION

Rafael Euclides Damasco

RESUMO

A Engenharia de Requisitos é responsável em identificar, analisar, documentar e gerenciar os requisitos de um software. Por esses motivos, esta se trata de uma fase importante no processo de criação de um software. Diversas técnicas e métodos de levantamento que facilitam a realizar a obtenção e gerenciamento desses requisitos. Porém, atualmente boa parte dos projetos de software que falham com o cumprimento de prazos e orçamentos, isso muitas vezes são ocasionados por falhas nos requisitos, já que esses são custosos de corrigir. Devido ao difícil aprendizado dessas técnicas que vêm sendo feitas normalmente através somente de maneira teórica, a elicitação e análise de requisitos acabam sendo mal feitas ou deixadas de lado durante o processo de desenvolvimento, é difícil ocorrer métodos mais práticos e dinâmicos para a transmissão do conhecimento. Este trabalho propõe criar um software para apoiar o ensinamento de técnicas de levantamento e análise de requisitos de forma mais ativa.

Palavras-chave: Ine; Ufsc; Engenharia; Engenharia de Software; Especificação; Documentação; Elicitação; Requisitos; Levantamento; Ensino-aprendizagem.

ABSTRACT

Requirements Engineering is responsible for identifying, analyzing, documenting, and managing the requirements of software. For these reasons, this is an important phase in the process of creating software. There are several techniques and survey methods that facilitate the gathering and management of these requirements. However, nowadays a good part of the software projects that fail to meet deadlines and budgets are often caused by flaws in the requirements, since these are costly to correct. Due to the difficult learning of these techniques that are usually done only in a theoretical way, requirements elicitation and analysis end up being poorly done or left aside during the development process, it is difficult to occur more practical and dynamic methods for the transmission of knowledge. This work proposes to create a software to support the teaching of requirements elicitation and analysis techniques in a more active way.

Keywords: Ine; Ufsc; Engineering; Software Engineering; Specification; Documentation; Elicitation; Requirements; Survey; Teaching-Learning.

1 INTRODUÇÃO

O termo Engenharia de Software surgiu em 1968, tendo sido utilizado pela primeira vez por uma engenheira da NASA, Margaret Hamilton, na NATO Conference on Software Engineering (ERDOGMUS; MEDVIDOVIĆ; PAULISCH, 2008), tratando o desenvolvimento de software de uma forma mais sistemática, se preocupando com todos os aspectos da produção de software e visando a empresa como um todo. “Engenharia de software é uma abordagem sistemática e disciplinada para o desenvolvimento de software” (PRESSMAN, 2006). A área de estudos dos softwares surgiu com a necessidade de lidar com o aumento de diversidade, demandas pela diminuição do tempo para entrega e desenvolvimento de software confiável, e para sanar esse problema, são utilizados processos, ou melhor uma sequência de métodos com a finalidade de desenvolver um software. Os processos de software são basicamente divididos em 4 etapas, especificação, projeto,

implementação, validação e evolução (PRESSMAN, 2006). Magela (2006) entende como Engenharia de Software.

> "O conjunto de técnicas, métodos, ferramentas e processos utilizados na
> especificação, construção, implantação e manutenção de um software que visa a garantir a gerência, o controle e a qualidade dos artefatos gerados através de recursos humanos."

Uma área importante da Engenharia de Software é a Engenharia de requisitos, que dispõe-se a facilitar a compreensão de um problema pelo engenheiro de software, e com a finalidade de sanar os problemas do desenvolvimento de aplicações.

> "Especificação de software ou engenharia de requisitos é o processo de
> compreensão e definição dos serviços requisitados do sistema e identificação de restrições relativas à operação e ao desenvolvimento do sistema. (SOMMERVILLE, 2011)".

É através da Engenharia de requisitos que pode-se viabilizar os insumos necessários para a elaboração do projeto, o que o cliente deseja do projeto e como os usuários finais vão interagir com o projeto. A principal necessidade da Engenharia de requisitos, como o próprio nome diz, é definir os requisitos do sistema, estabelecendo o que o sistema deve fazer e quais são suas restrições.

Sommerville (2005) descreve Requisitos como as descrições das suas funções e restrições, sendo o processo de descobrir, analisar, documentar e verificar essas funções e restrições denominadas de Engenharia de requisitos.

Os requisitos elicitados e analisados são escritos em um documento chamado documento de requisitos de software, ou também chamado de Especificação de Requisitos de Software, que é o produto final do processo de descobrimento de requisitos que reúne necessidades e propósitos demandados pelos stakeholders, neste documento consta todos os requisitos que o sistema deve ter. Este documento pode ser considerado "uma declaração formal de requisitos de clientes, usuários finais e desenvolvedores de software" (SOMMERVILLE & SAWYER, 1997). Um documento de requisitos, segundo a norma (IEEE, 1996), deverá conter "declarações não ambíguas e ser verificável, consistente, modificável, rastreável e usável durante todas as fases do ciclo de vida do requisito".

Para Sommerville (2011) uma das maiores dificuldades dos engenheiros de software é compreender as informações que o cliente transmite e elaborar um documento de requisito

que esteja de acordo com a realidade do software requisitado. Para que os requisitos sejam bem compreendidos é importante que não haja margem para má interpretações, este é um dos grandes motivos para os atrasos no projeto, erro na elaboração do escopo e no orçamento.

2 DESENVOLVIMENTO

Este projeto propõe um jogo computacional com o intuito de auxiliar o ensino de especificação de requisitos, tem como enfoque apresentar situações onde o usuário tenha contato com técnicas de especificação de requisitos por intermédio de um ambiente controlado. O jogo apresenta aos usuários documentos ou partes de documentos de requisitos, e o jogador/aluno deverá exercitar a habilidade de interpretar e complementar aspectos desta documentação. O jogo foi projetado para ser jogado por um único jogador, a interação online, na qual a ação de um jogador interfere em outro jogador não faz parte do escopo deste trabalho, porém o progresso de todos que utilizaram o jogo poderá ser comparado e ranqueado para uma possível questão de competitividade.

As formas de especificação a serem abordadas no jogo serão: linguagem natural, cenários, casos de uso e refinamento de casos de uso com diagrama de atividade.

Não será englobado neste trabalho técnicas de estudo de viabilidade nem qualquer outro subprocesso de engenharia de requisitos.

2.1 VISÃO GERAL DO JOGO

Assim como todo projeto de sistemas, um jogo surge de uma ideia, que posteriormente recebe pequenas melhorias ou mudanças. Há algumas técnicas para fomentar o surgimento de ideias, seja para o jogo ou qualquer outro segmento que necessite ser alimentados pela criatividade, tais como o Brainstorm, que pode ser feito individualmente ou em grupo, com a intenção de elicitar o maior número de ideias para posteriormente selecionar o que realmente será usado no projeto (ESTEVES, 2020). Para que a ideia crie corpo e torne-se de fato um jogo que proporcione uma experiência cativante ao jogador, é importante

conhecer o público-alvo e colocar-se no lugar do player, tornando assim o jogo mais desafiador e atrativo (COPLE, 2019).

No que diz respeito a Design de games, quando a ideia central do jogo já está amadurecida, o ideal é seguir para a próxima etapa, que é a criação do Storyline, o qual exerce a função de um roteiro preliminar. Para a criação do Storyline é importante compreender quem são os personagens do jogo, suas interações e a participação de cada um deles no enredo. (COPLE, 2019).

O próximo passo para o desenvolvimento do jogo é a criação do GDD (Game design document), que se trata de um conjunto de todos os elementos acerca do jogo reunidos em um único documento, ele serve como base para a elaboração da maioria dos projetos abrangendo a criação de jogos digitais (COPLE, 2019). O GDD serve para apoiar a equipe de desenvolvimento caso haja alguma dúvida sobre o que deverá ser desenvolvido e as delimitações acerca do projeto, portanto, o que não estiver especificado no GDD não faz parte do sistema desenvolvido (COPLE, 2019; FLYNT, 2005; PERUCIO, 2007).

Os tópicos abaixo foram adaptados das obras de Scott Rogers (2014) e Denis Cople (2019) para formular o GDD do presente trabalho.

Conceitos do Jogo: O jogo será voltado aos estudantes que pretendem revisar ou assimilar de forma mais descontraída o conteúdo abordado nas aulas de Engenharia de Software. O jogador estará exposto a práticas pedagógicas nas quais ele deve agir de forma ativa para resolver determinadas situações utilizando os conceitos de Engenharia de requisitos para resolver os projetos.

Nome: Retraining.

Número da versão, Autor, Data: Primeira versão do jogo Retraining; de autoria de Rafael Euclides Damasco; começou a ser projetado em 01/04/2021.

Gênero: RPG

Público Alvo: O público-alvo do jogo serão os alunos de graduação dos cursos da área de tecnologia da informação.

Descrição do estilo de jogo: O jogo se passa em uma sala de escritório, com programadores e o chefe da empresa.

Síntese da história: A maioria das pessoas quando iniciam seus estudos buscam aplicá-los na prática, e o meio mais fácil e explorado pelos estudantes é iniciar em uma empresa como estagiário, e este será o ponto de partida do jogo. Como todo bom estagiário você será responsável por fazer e servir cafezinho para os outros integrantes da equipe de desenvolvimento, aprender os conteúdos que você não prestou atenção na sala de aula e agora precisa pôr em prática, participar das reuniões diárias, ser "bombardeado" de documentos de requisitos que você deve interpretar de forma correta. Quanto melhor o seu desempenho nos projetos, maior será a confiança do chefe em você, portanto, suas chances de efetivação também serão maiores.

Esboço do Jogo: Retraining é um jogo educacional em terceira pessoa, que transporta o usuário ao cotidiano de uma empresa de tecnologia iniciando sua carreira como estagiário, e que ao final de seu contrato almeja ser efetivado. E como todo início de carreira, passamos por um período de adaptação, falta de experiência e novos relacionamentos no âmbito profissional.

Como estagiário do projeto, de início você não participará dos projetos da equipe, por isso seu trabalho será um pouco mais fácil, você passará por um período de treinamento na empresa a fim de obter os conhecimentos necessários. O treinamento é composto por algumas fases, que abordarão perguntas de conhecimento geral e você terá a tarefa de analisar e completar essas perguntas. Com o passar do tempo e ao adquirir mais experiência você passará a ter contato direto com os projetos, aumentando sua responsabilidade na empresa, tornando-se responsável por auxiliar nos projetos presentes no contexto do jogo.

As formas de especificação a serem abordadas no jogo serão: linguagem natural, cenários, casos de uso e refinamento de caso de uso com diagrama de atividade.

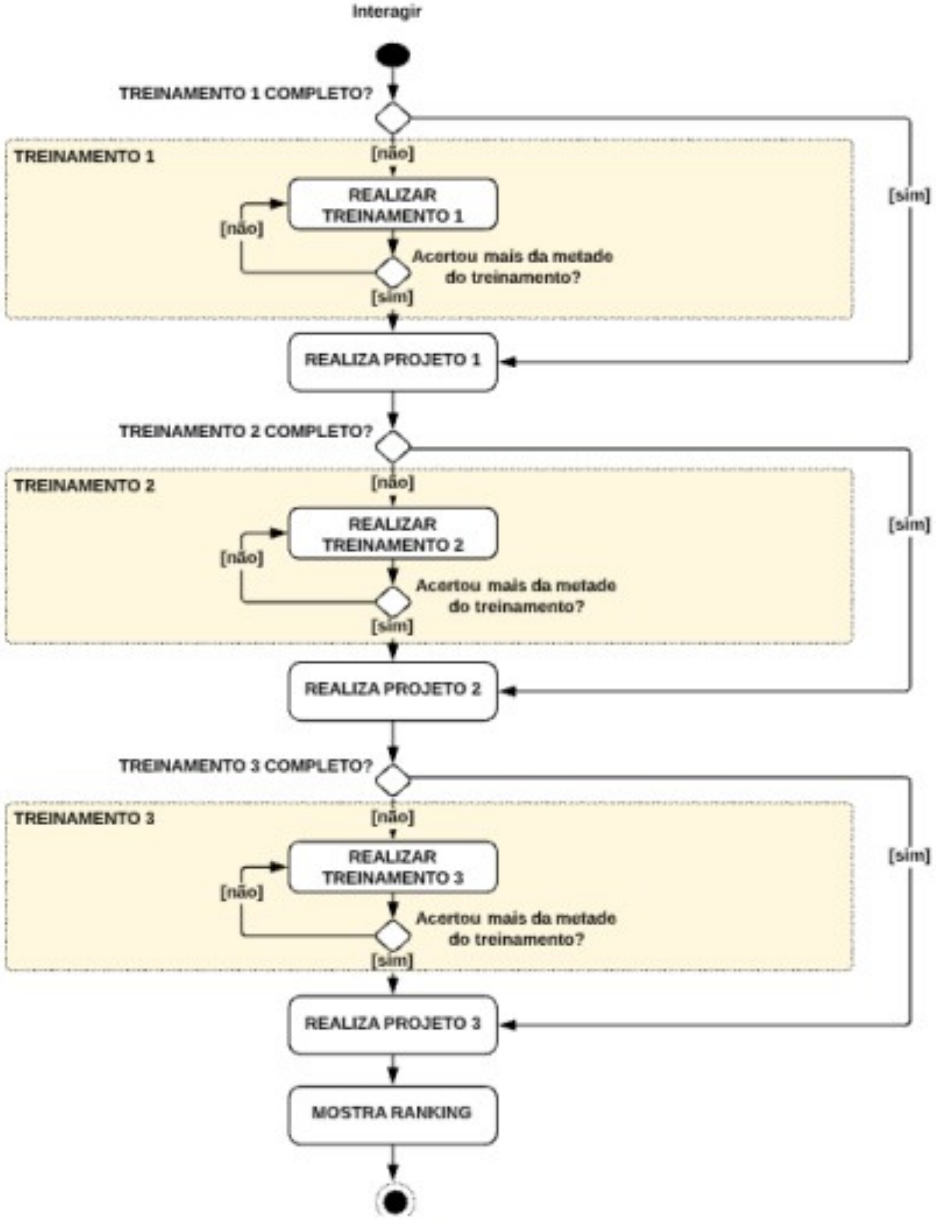
Fluxo do jogo: Ao iniciar o jogo o usuário poderá dar nome ao seu personagem e escolher algumas características para o mesmo, como por exemplo: comunicativo, focado ou ponderado. Depois de escolher seus traços de personalidade seu personagem aparecerá na tela apresentando de forma visual qual personalidade foi escolhida.

No menu principal do jogo o usuário poderá ver a quantidade de experiência total que possui.

O jogo foi dividido em 3 fases, sendo que em cada fase o jogador passará por dois períodos distintos. Primeiro haverá um treinamento, e o usuário receberá informação a respeito de um determinado conteúdo, para posteriormente responder perguntas, com o intuito reforçar o conteúdo e alcançar o maior número de experiência possível. Caso o jogador tenha acertado mais da metade do treinamento, ele poderá em seguida colaborar com a equipe da empresa em um projeto. Vale lembrar que cada fase será composta por temáticas diferentes (Figura 15).

Ao terminar todos os projetos presentes nas 3 fases do jogo, o jogo é encerrado e disponibiliza uma tabela de ranking (Figura 15).

Figura 15 – Diagrama de atividade Retraining



Fonte: (Própria, 2021)

2.2 FERRAMENTAS UTILIZADAS

Unity: A Unity é um ambiente para o desenvolvimento de jogos, possui um conjunto de ferramentas integrado que permite criar jogos tanto em 3D quanto em 2D. Com a Unity o desenvolvedor tem a capacidade de construir seus jogos para diversas plataformas e a facilidade de utilizar objetos prontos através da loja oficial da Unity ou da comunidade desenvolvedora.

O presente trabalho foi desenvolvido utilizado a versão 2021.1.12f1 da Unity, disponível em https://unity3d.com/pt/get-unity/download. Como a Unity permite que o desenvolvedor possa importar pacotes gráficos através da Asset Store, alguns dos componentes gráficos deste jogo foram importados da própria loja oficial da Unity, mas também foram utilizados pacotes vindos do site https://opengameart.org.

Dreamlo: Para o sistema de ranking, foi utilizado o Plug-in Dreamlo, que usa uma solicitação HTTP GET para armazenar o nome e a pontuação dos usuários que obtiveram maior quantidade de experiência. Com o Dreamlo foi possível mostrar na interface da Unity o ranking atual de players e atualizá-los. http://dreamlo.com.

Itch.io: É um site para que desenvolvedores “Indie” possam hospedar seus jogos e assim disponibilizar para um maior número de jogadores. Foi feita a build do jogo e exportado para o site Itch.io, e foi através da Itch.io que o jogo Retraining estará disponível para o público geral.

2.3 OPERACIONALIDADE

O jogo pode ser acessado em: https://rafaeldamasco.itch.io/retraining. Código fonte: https://github.com/RafaelDamasco/Retraining

Para uma melhor compreensão da aplicação, o jogo foi dividido em duas partes, a primeira contém informações sobre a navegação antes do início do jogo e a segundo apresenta o funcionamento do jogo.

2.3.1 Retraining - Navegação

Ao abrir o jogo, o usuário é apresentado a tela de menu inicial (Figura 18). Ao selecionar a opção “Sair” o jogo é encerrado. Ao selecionar a opção “Iniciar Jogo” a aplicação exibirá informação a respeito do funcionamento do jogo, bem como uma pequena história do

dia a dia de um estagiário em uma empresa qualquer (Figura 19). Caso o não deseje avançar no jogo ele pode voltar para o menu inicial e encerrar a aplicação.

Figura 18 – Menu Inicial

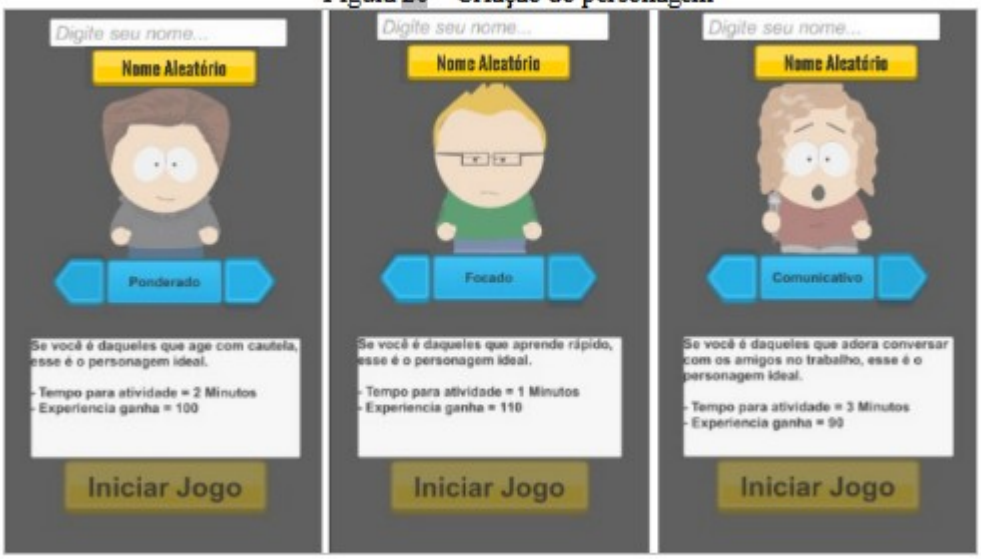
Fonte: (Própria, 2021)

Figura 19 – Apresentação Jogo

Fonte: (Própria, 2021)

Caso o usuário selecione a opção “Criar Personagem” ele poderá navegar entre três opções de personagem, cada um deles contará com uma aparência diferente e que graficamente se assemelha a uma personalidade. As personalidades disponíveis são: Comunicativo, Focado e Ponderado, cada uma delas trará um bônus e um ônus diferente ao usuário. Para habilitar o botão de “Iniciar Jogo” o usuário primeiramente deverá digitar o nome que deseja dar para o personagem ou clicar na opção de nome aleatório. A funcionalidade de gerar nome aleatório foi inserida no jogo, para que seja possível jogar em dispositivos mobile, já que não estava sendo possível usar o teclado dos aparelhos como ferramenta de entrada de informação (Figura 20).

Figura 20 – Criação do personagem



Fonte: (Própria, 2021)

2.3.1 Retraining - Jogo

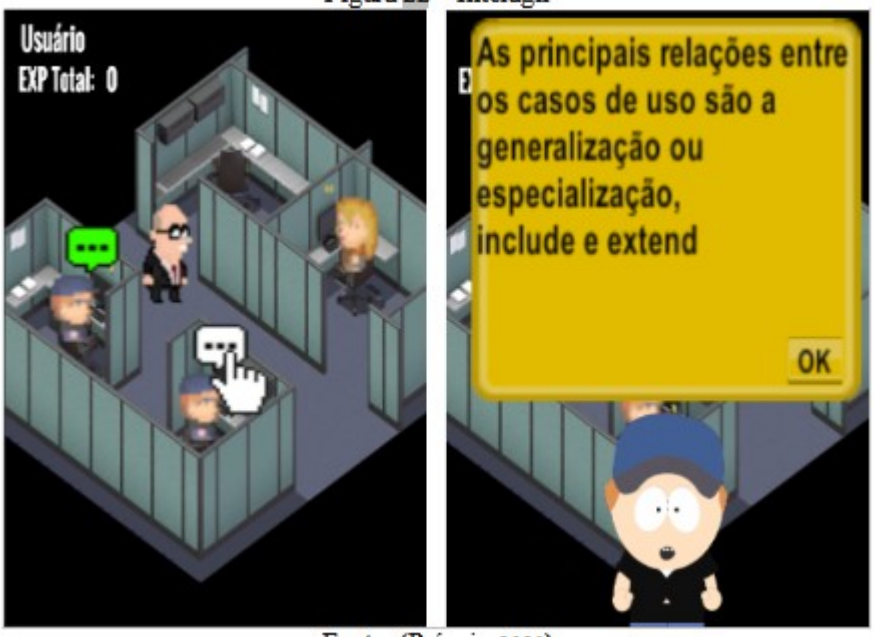
Antes de entrar em detalhes do jogo, é descrito um pouco da parte gráfica do início do jogo (Figura 21)

Figura 21 – Início Jogo

Fonte: (Própria, 2021)

O jogo se passa em um escritório, nele pode-se observar quatro personagens, sendo eles: o personagem escolhido anteriormente e três npcs (non-player character ou personagem não jogável). O primeiro deles da direita pra esquerda é o personagem que o usuário escolheu. O segundo é um npc que possui um balão branco acima da cabeça, se o usuário leu as informações a respeito do funcionamento do jogo nas telas passadas saberá que aquele npc não é obrigatório para o fluxo do jogo (Figura 22). Logo ao lado observa-se um npc muito semelhante ao anterior, porém com um balão verde acima da cabeça, como descrito anteriormente esse npc será responsável por aplicar o treinamento ao usuário. Mais a cima temos um outro npc um pouco diferente dos demais, ele parece um pouco mais velho, um pouco mais experiente e é ele o responsável por apresentar ao usuário um projeto em que a empresa precisará se dedicar.

Figura 22 – Interagir



Fonte: (Própria, 2021)

O fluxo do jogo é contínuo e feito de forma sequencial. Isso quer dizer que o usuário primeiramente passa por um período de treinamento sobre um conteúdo (Figura 23), e caso obtenha acertos maior que a metade das questões, ele estará apto a executar o projeto daquele treinamento.

Figura 23 – Realizar Treinamento

Fonte: (Própria, 2021)

O projeto será introduzido logo após o usuário completar o primeiro período de treinamento. A problemática abordada no projeto trará a tona a realidade de uma empresa que pretende gerenciar um estacionamento através de um software. Como dito anteriormente o npc “chefe” que disponibilizará o projeto para o usuário, surgirá um balão amarelo contendo um “ ! “ (Figura 24).

Será apresentada uma tela ao jogador mostrando o projeto que ele deverá cumprir. Nesse momento o chefe informa que o usuário será avaliado, quanto tempo ele tem para responder todas as perguntas e quanto de experiência ele ganha ao responder de forma correta uma pergunta, Todas essas variáveis foram definidas na hora em que o usuário escolheu a personalidade para o seu personagem. Ao responder todas as perguntas ou se o tempo se esgotar são exibidos a porcentagem de acerto das questões e quanto de experiência o jogador obteve naquele projeto (Figura 24).

Figura 24 – Projeto

Fonte: (Própria, 2021)

Este ciclo de treinamento seguido de projetos se repete até o projeto três, quando é exibida uma tela de ranking contendo o nome e a experiência dos cinco jogadores que mais obtiveram experiência (Figura 25).

Figura 25 – Ranking

Fonte: (Própria, 2021)

Todas as questões, tanto as do período de treinamento, quanto as do projeto estão contidas no apêndice A e B, respectivamente.

2.4 AVALIAÇÃO

Este capítulo tem a finalidade de descrever como foi feita a avaliação do jogo proposto.

2.4.1 Planejamento

Após a finalização do jogo Retraining, foi feita uma avaliação entre alunos selecionados nas áreas de tecnologia, para tal, decidiu-se pelo modelo de avaliação MEEGA+.

O MEEGA+ é um modelo sistemático para a avaliação de jogos educacionais voltados para o ensino de computação, este modelo permite avaliar a qualidade, eficiência e eficácia do presente jogo (SAVI, 2011). Segundo as análises do instrumento de medição do modelo chegou-se a um nível de confiabilidade aceitável para a avaliação da qualidade de jogos usados para o ensino de computação (PETRI, 2020).

O modelo utiliza a abordagem GQM (Goal-Question-Metric) para obter resultados das seguintes variáveis: atenção focada, diversão, desafio, interação social, confiança, relevância, satisfação e usabilidade (SAVI, 2011). O modelo inicia com a coleta de informações demográficas do usuário e segue para o questionário a respeito do jogo (PETRI, 2020).

O objetivo da avaliação foi de obter e atender os vieses anteriormente destacados em avaliar o jogo em questão utilizando o modelo MEEGA+. Bem como avaliar a eficácia do jogo como ferramenta complementar de ensino.

2.4.2 Execução

Em virtude da pandemia do Covid-19 o questionário de avaliação foi aplicado de maneira virtual, em que foram selecionados cinco alunos que já cursaram a disciplina de

Engenharia de Software e aplicado o questionário do modelo de avaliação de jogos educativos MEEGA+ (Anexo A).

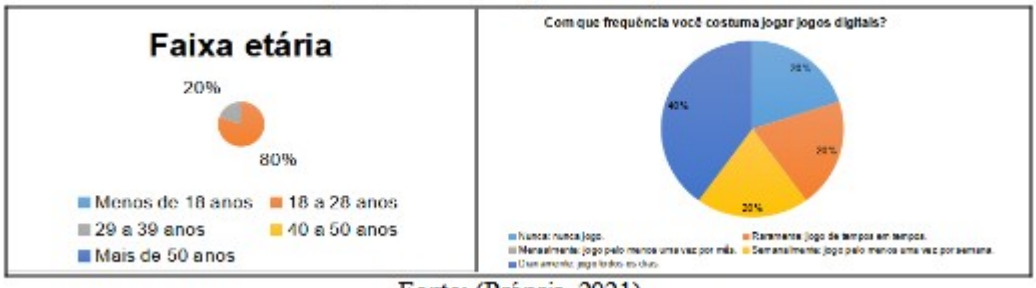
Ao avaliar o jogo a intenção foi medir se o método de ensino proposto seria complementar ao método de ensino tradicional ou não, analisando se de fato o jogo computacional proposto agregaria conhecimento para o jogador de forma lúdica, melhorando assim o aprendizado do aluno.

2.4.3 Resultados

Esta seção destina-se a apresentar os resultados das avaliações citadas na seção anterior utilizando-se do modelo MEEGA+. O questionário é dividido em 3 partes: informações demográficas, experiência do jogador e percepção da aprendizagem.

No que diz respeito a informações demográficas, a amostra é predominantemente masculina, composta por 80% de homens. A faixa etária entre os intervalos de 18 a 28 anos corresponde a 80% da amostra. Quanto a frequência em que a amostra costuma jogar jogos digitais, verificou-se que 40% jogam todos os dias, 20% da amostra nunca, raramente ou semanalmente tem contato com jogos eletrônicos e ninguém respondeu jogar mensalmente. No que diz respeito a jogos não digitais, 40% dos avaliados raramente os jogam, essa mesma porcentagem respondeu ter contato com jogos não digitais mensalmente e 20% responderam ter contato com este tipo de jogo semanalmente.

Figura 26 – Informações Demográficas



Fonte: (Própria, 2021)

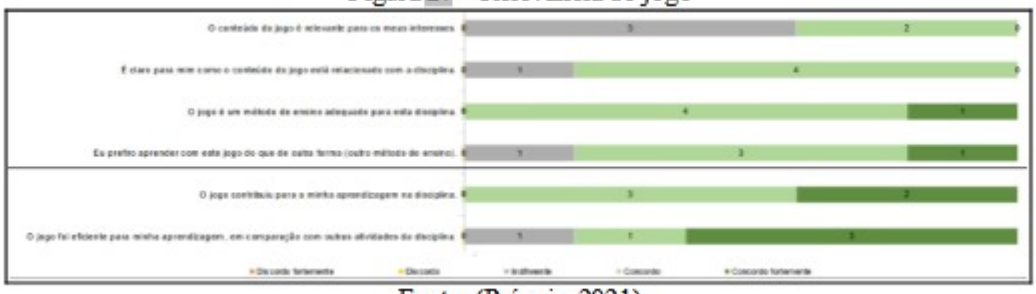
Dividiu-se os aspectos avaliados no questionário em dois grupos, o primeiro informa sobre a

relevância do jogo para o ensino, já o segundo aborda temas como satisfação, diversão, interação social, desafio, atenção focada e usabilidade do jogo.

Sobre a relevância do jogo como método de ensino obteve-se resultados positivos, todos os entrevistados afirmaram que o jogo é um método de ensino adequado para a disciplina de engenharia de requisitos (Figura 26). A percepção de aprendizagem do jogo foi outro aspecto que agradou aos usuários, a grande maioria respondeu que o jogo contribui para a aprendizagem na disciplina.

Analisando-se as outras temáticas abordadas no questionário pode-se observar que em relação a variável desafio abordado no jogo, os avaliados informaram em sua maioria que o jogo era adequado e desafiador para eles. A satisfação dos jogadores ao jogar o jogo foi descrita como boa pela ampla maioria, e, a interação social foi o aspecto que menos agradou aos jogadores, haja vista que o jogo não tinha nenhuma interação social. O elemento abordado no questionário relacionado a diversão foi o que mais agradou aos usuários, todos eles responderam que o jogo os fez sorrir. A atenção focada aplicada ao jogo pelos jogadores foi algo que não obteve bom resultado, sendo irrelevante pela maioria. Sobre sua usabilidade, pode-se dizer que foi satisfatória, menos em relação aos tamanhos de fonte e estilo das letras nos textos do jogo.

Figura 27 – Relevância do jogo



Fonte: (Própria, 2021)

Vale ressaltar que o questionário foi aplicado em um universo restrito de apenas cinco alunos, portanto trata-se de uma avaliação ainda inicial acerca do jogo e que não comprova a sua real eficiência para o ensino. Porém, foi possível observar através das respostas do questionário que há indícios de que o jogo pode ser uma opção como ferramenta complementar no ensino de especificação de requisitos.

3 CONSIDERAÇÕES FINAIS

A Engenharia de Requisitos é uma etapa fundamental para auxiliar na identificação, análise, documentação e gerenciamento de requisitos de um software, tendo em vista sua importância significativa para a elaboração de um projeto, o presente trabalho propôs uma sugestão de auxílio no processo de criação de um software.

A escolha do tema foi pautada na dificuldade de cumprimento de prazos e orçamentos dos projetos de software apresentados atualmente, pois mesmo com diversas técnicas e métodos de levantamento para melhorias na aplicação as falhas nos requisitos podem tornar sua execução difícil e cara.

Com base nisso, torna-se evidente como a dificuldade apresentada pelos alunos no processo de aprendizagem de Engenharia de Requisitos reflete na ausência ou errônea aplicação em projetos. Segundo Glasser, parte da defasagem do estudo de Engenharia de Requisitos provém do método de ensino – pautado apenas no teórico, e, devido à falta de compreensão de sua real aplicação durante o processo de desenvolvimento, os alunos não dão a devida importância à elicitação e análise de requisitos. Logo, a intenção do presente trabalho propôs a criação de um software para apoiar o ensinamento de técnicas de levantamento e análise de requisitos de forma mais ativa, visando atingir principalmente o público em âmbito universitário e em processo de aprendizagem de Engenharia de Requisitos.

A antelação do tema, provém de sua importância no meio acadêmico, pois através da aprendizagem ativa por meio do jogo computacional proposto com a análise de abordagens para o ensino de Engenharia de Requisitos, visa auxiliar e ensinar a documentação e especificação de requisitos de software, e, assim, diminuir as falhas e dificuldades de execução em futuros projetos.

A ferramenta de ensino computacional Retraining tem como finalidade ajudar os alunos a entender a matéria de Elicitação de Requisitos, e através da aplicação do questionário MEEGA+, foi possível identificar a partir das respostas, indícios de cumprimento dos objetivos propostos pelo jogo Retraining. A partir deste modelo sistemático de avaliação no universo em que foi aplicado, foi possível identificar que as respostas indicaram haver

eficácia do jogo como ferramenta complementar de ensino, e através dos dados obtidos entre os alunos selecionados nas áreas de tecnologia o resultado foi satisfatório, e o jogo cumpriu assim, os objetivos propostos neste presente trabalho.

REFERÊNCIAS

APPOLINÁRIO, Fabio. **Dicionário de Metodologia Científica**. 2. ed. São Paulo: Atlas, 2011. 295p.

BARBOSA, E. F. & MOURA, D. G. **Metodologias ativas de aprendizagem na Educação Profissional e Tecnológica**. B. Tec. Senac, Rio de Janeiro, v. 39, n.2, p.48-67, maio/ago. 2013.

BEZERRA, Eduardo. **Princípios de Análise e Projeto de Sistemas com UML.** Rio de Janeiro: Campus, 2002.

BOEHM, B.W. **Software risk management.** IEEE Computer Society Press: Washington, 1989.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. UML **Guia do Usuário**. Rio de Janeiro: Campus, 2000.

C. J. Neill and P. A. Laplante. **Requirements engineering: The state of the practice.** IEEE Software, vol 20(6):40–45, 2003.

COPLE, Denis. **Desenvolvimento De Jogos II**. 1ª Edição Rio de Janeiro: SESES, 2019.

DAVIS, A. M. **Software Requirements: Objects, Functions and States.** Englewood Cliffs, NJ: Prentice Hall, 1993.

ENGHOLM JÚNIOR, H. **Engenharia de Software na prática**. São Paulo: Novatec, 2010.

ESTEVES, Rodrigo. Brainstorm: **Como gerar ideias com mais eficiência**. Dash Editora, 2020.

FLYNT, John P. **Software Engineering for Game Developers.** Boston: Thomson Course Technology, 2005.

G. PETRI, C. G. VON WANGENHEIM, AND A. F. BORGATTO. **Meega+: Um modelo para a avaliac¸ao de jogos educacionais para o ensino de computação** Revista Brasileira de Informática na Educação27(03):52, 2020.

GIL, A. C**. Métodos e técnicas de pesquisa social**. 6 ed. São Paulo. 2008.

GLASSER, W. **Teoria da Escolha: uma nova psicologia de liberdade pessoal.** São Paulo: Mercuryo, 2001.

GUEDES, Gilleanes T. A. UML: **Uma abordagem prática**. São Paulo: Novatec, 2006.

H. Erdogmus, N. Medvidović and F. Paulisch, "**50 Years of Software Engineering,**" in *IEEE Software*, vol. 35, no. 5, pp. 20-24, September/October 2018. doi: 10.1109/MS.2018.3571240

IEEE Std. 12333-1996 IEEE **Guide for Developing System Requirements Specification.** The Institute of Electrical and Electronics Engineers. Piscataway, NJ. Guide, EUA, 1996.

ISO/IEC/IEEE 29148:2018. **Systems and Software Engineering Life Cycle Processes Requirements Engineering.** The International Organization for Standardization (ISO)/The International Electrotechnical Commission (IEC)/The Institute of Electrical and Electronics Engineers (IEEE) Computer Society. 2018.

LEITE, J. C. **Notas de Aula de Engenharia de software.** Natal, RN, 2000.

LEITE, J.C.S.P.; LEONARDI, M.C. (1998). **Business Rules as organizational policies.** In: Proceedings of the 9th International Workshop on Software Specification & Design. ISE-Shima, Japan. 1ed. USA: IEEE CSP, Los Alamitos, CA. P. 68-76, Apr.

LIMA, A. D. S. **UML 2.0: do requisito à solução.** 2. ed. São Paulo: Érica, 2007.

LUQUE, L. **Engenharia de Software II**, 2017.

MAGELA, Rogério. **Engenharia de Software Aplicada – Princípios –** 1º. Ed. – Alta Books, 2006.

MORAN, José Manuel. **Tecnologias digitais para uma aprendizagem ativa e inovadora**, Campinas, SP

MORAN, José Manuel. **A educação que desejamos: novos desafios e como chegar lá**. 2. ed. Campinas, SP: Papirus, 2007. 174p.

PERUCIO, A. S.; BERTHÊM, A. C. de; BERTSCHINGER, G. L.; MENEZES, R. R. C. **Desenvolvimento de jogos eletrônicos: teoria e prática.** 2.ª ed. São Paulo: Novatec, 2007.

PFLEEGER, S. L. **Engenharia de software: teoria e prática.** 2.ª ed. São Paulo: Prentice Hall, 2004.

PRESSMAN, R.S. **Engenharia de software.** Pearson Makron Books, 2006.

POHL, Klaus. **Requirements engineering: fundamentals, principles, and techniques.** Springer Publishing Company, Incorporated, 2010.

ROGERS, Scott. **Level Up! The guide to great video game design.** John Wiley & Sons,

2014.

SAVI, R., GRESSE VON WANGENHEIM, C., E BORGATTO, A. (2011). **Um Modelo de Avaliação de Jogos Educacionais na Engenharia de Software**. Anais do XXV Simpósio Brasileiro de Engenharia de Software (SBES 2011), pp. 194-203.

SEVERINO, Antonio Joaquim. **Metodologia do Trabalho Científico.** São Paulo: Cortez, 2007

SHANE H, S.W. **Standish group 2015 Chaos Report - Q&A**. October 2015.

SILVA, E. L., MENEZES, E. M. **Metodologia da pesquisa e elaboração de dissertação**. 3.

ed. rev. atual., Laboratório de Ensino a Distância. da UFSC, Florianópolis, Santa Catarina. 2001.

SOMMERVILLE, I. **Engenharia de Software**. 6ª. Ed. São Paulo: PEARSON Addison Wesley, 2005.

SOMMERVILLE, I. **Engenharia de Software.** 9ª. Ed. São Paulo: PEARSON Addison Wesley, 2011.

SOMMERVILLE, I. and KOTONYA, G. (1998) **Requirements Engineering: Processes and Techniques.** John Wiley & Sons, Inc., Hoboken.

VIZCAÍNO, A; PIATTINI, M.; CABALLERO, I.; MONASOR, M. J. **Preparing Students and Engineers for Global Software Development**: A Systematic Review. IEEE International Conference On Global Software Engineering. Annals of..., Princeton, 2010.

VAZQUEZ, Carlos Eduardo; SIMÕES, Guilherme Siqueira. **Engenharia de Requisitos: Software Orientado ao Negócio.** Rio de Janeiro: Brasport, 2016.

YOUNG, R. **The Requirements Engineering handbook**. Artech House, 2004.

APÊNDICE E – CÓDIGO FONTE

- **comandosBasicos.cs**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class comandosBasicos : MonoBehaviour
{
    public void carregaCena (string nomeCena)
    {
        SceneManager.LoadScene(nomeCena);
    }
    public void QuitGame()
    {
        //UnityEditor.EditorApplication.isPlaying = false;
        Application.Quit();
    }
}
```

- **CharacterSelectionMenu.cs**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class CharacterSelectionMenu : MonoBehaviour
{
    public GameObject[] playerObjects;
    public static int selectedCharacter = 0;
    public string inicioJogo = "InicioJogo";
    public string gameScene = "Character Selection Scene";
    public static string selectedCharacterDataName = "SelectedCharacter";
    public string[] nomes;
    public static int fixedTempo;
    public static int tempo;
    public static int exp;
    public Button iniciarJogo;
    public InputField playerName;

    void Start()
    {
        playerName.characterLimit = 11;
```

```
    HideAllCharacters();
    selectedCharacter = PlayerPrefs.GetInt(selectedCharacterDataName, 0);
    playerObjects[selectedCharacter].SetActive(true);
}
// Updates button's text while user is typing
public void gerarNome()
{
    playerName.text = nomes[Random.Range(0, nomes.Length)];
    iniciarJogo.interactable = true;
}
void Update()
{
    if (playerName.text == "")
    {
        iniciarJogo.interactable = false;
    }
}

private void HideAllCharacters()
{
    foreach (GameObject g in playerObjects)
    {
        g.SetActive(false);
    }
```

```
}

public void NextCharacter()
{
    playerObjects[selectedCharacter].SetActive(false);
    selectedCharacter++;
    if (selectedCharacter >= playerObjects.Length)
    {
        selectedCharacter = 0;
    }
    playerObjects[selectedCharacter].SetActive(true);
}

public void PreviousCharacter()
{
    playerObjects[selectedCharacter].SetActive(false);
    selectedCharacter--;
    if (selectedCharacter < 0)
    {
        selectedCharacter = playerObjects.Length - 1;
    }
    playerObjects[selectedCharacter].SetActive(true);
}
```

```
public void StartGame()
{
    temaJogo.playernamestr = playerName.text;
    if (selectedCharacter == 0) //COMUNICATIVO
    {
        fixedTempo = 180;
        tempo = 180;
        exp = 17;
    }

    else if (selectedCharacter == 1) //PONDERADO
    {
        fixedTempo = 120;
        tempo = 120;
        exp = 18;
    }
    else if (selectedCharacter == 2) //FOCADO
    {
        fixedTempo = 60;
        tempo = 60;
        exp = 19;
    }

    PlayerPrefs.SetInt(selectedCharacterDataName, selectedCharacter);
```

```
        SceneManager.LoadScene(inicioJogo);

    }
}
```

- **temaJogo.cs**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class temaJogo : MonoBehaviour
{

    public GameObject canvaHighScore;
    public GameObject aviso;
    public GameObject P0;
    public GameObject P1;
    public GameObject P2;
    public GameObject T0;
    public GameObject T1;
    public GameObject T2;
```

```
public GameObject D1;
public GameObject D2;
public GameObject D3;
public Text txtExp;
public Text txtNomeTema;
public Text txtNomeProjeto;
public string[] nomeTema;
public string[] nomeProjeto;
private int idAvaliacao;
public static int idTema;
public static int idProjeto;
private int exp;
public static int aprovadoP;
public static string playernamestr;
public Text playername;

void Start()
{
      playername.text = playernamestr;
   if (aprovadoP == 0)
   {
      D1.SetActive(true);
      D2.SetActive(false);
      D3.SetActive(false);
```

```
}
if (aprovadoP == 1)
{
   D1.SetActive(false);
   D2.SetActive(true);
   D3.SetActive(false);
}
if (aprovadoP == 2)
{
   D1.SetActive(false);
   D2.SetActive(false);
   D3.SetActive(true);
}
if (notaFinal.notaF <= responder.questoes / 2)
{
   T0.SetActive(true);
   P0.SetActive(false);
   notaFinal.notaF = 0;
   Debug.Log("TIROU MAIS QUE METADE DAS QUESTOES ");
   Debug.Log("aprovadop= " + aprovadoP);
   if (aprovadoP == 1)
   {
      Debug.Log("APROVADO PO ");
      //P0.SetActive(false);
```

```
            T0.SetActive(false);
            T1.SetActive(true);
            //P1.SetActive(false);
        }

        if (aprovadoP == 2)
        {
            Debug.Log("APROVADO P1 ");
            //P0.SetActive(false);
            T0.SetActive(false);
            T1.SetActive(false);
            T2.SetActive(true);
            //P1.SetActive(false);
        }
        Pontuacao.scoreTotalT = 0;
        Pontuacao.scoreTotalP = 0;
        Pontuacao.notaFinalT = 0;
        Pontuacao.notaFinalP = 0;

    }

    else if (notaFinal.notaF > responder.questoes / 2)
    {
        T0.SetActive(false);
```

```
    P0.SetActive(true);
    notaFinal.notaF = 0;

    if (aprovadoP == 1)
    {
        P0.SetActive(false);
        T0.SetActive(false);
        T1.SetActive(false);
        P1.SetActive(true);
        zerarScore();
    }
    if (aprovadoP == 2)
    {
        P0.SetActive(false);
        P1.SetActive(false);
        P2.SetActive(true);
        zerarScore();
    }
    Debug.Log("NAO TIROU MAIS QUE METADE DAS QUESTOES ");
}



//idTema = 0;
```

```
    //idProjeto = 0;
    txtNomeTema.text = nomeTema[idTema];
    txtNomeProjeto.text = nomeProjeto[idProjeto];
    txtExp.text = "EXP Total:   " + Pontuacao.highScoreP.ToString();
}
void Update()
{
    Debug.Log("script temaJogo");
}
void zerarScore()
{
    Pontuacao.scoreTotalT = 0;
    Pontuacao.scoreTotalP = 0;
    Pontuacao.notaFinalT = 0;
    Pontuacao.notaFinalP = 0;
}
public void selecioneTema(int i)
{
    idTema = i;
    txtNomeTema.text = nomeTema[i];
}
public void selecioneProjeto(int i)
{
    idProjeto = i;
```

```
    txtNomeProjeto.text = nomeProjeto[i];

}
public void jogar()
{
    SceneManager.LoadScene("T" + idTema.ToString());

}
public void avaliar()
{
    SceneManager.LoadScene("P" + idProjeto.ToString());

}
public void finalizarJogo()
{
    aviso.SetActive(false);
    canvaHighScore.SetActive(true);
    Debug.Log(playernamestr);
    Debug.Log(Pontuacao.highScoreP);

    Invoke("Upload",1f);
}
public void Upload()
{
```

```
        HighScores.UploadScore(playernamestr, Pontuacao.highScoreP);
    }
}
```

- **responder.cs**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class responder : MonoBehaviour
{
    public string resultadoT0 = "ResultadoT0";
    //public GameObject ResultadoT1;
    public GameObject T1;
    //public GameObject Avan�ar;
    //public GameObject OK;
    private int idTema;

    public Text pergunta;
    public Text respostaA;
    public Text respostaB;
```

```
public Text respostaC;
public Text respostaD;
public Text infoRespostas;

//IMAGENS DAS QUESTOES

public GameObject questao4;
public GameObject questao5;
public GameObject questao6;
public GameObject questao7;
public GameObject apagador;
public Text txtNota;
public string[] perguntas;      // armazena todas as perguntas
public string[] alternativaA;   // armazena todas as alternativas A
public string[] alternativaB;   // armazena todas as alternativas B
public string[] alternativaC;   // armazena todas as alternativas C
public string[] alternativaD;   // armazena todas as alternativas D
public string[] corretas;       // armazena todas as alternativas corretas
private int idPergunta;
public static int acertos;
public static int questoes;
public static int media;
public static int notaFinal;
public int idResultado = temaJogo.idTema;
```

```
void Start()
{
    idTema = 1;
    idPergunta = 0;
    acertos = 0;
    questoes = perguntas.Length; //QUANTAS QUESTOES TEM MEU TEMA?
    pergunta.text = perguntas[idPergunta];
    respostaA.text = alternativaA[idPergunta];
    respostaB.text = alternativaB[idPergunta];
    respostaC.text = alternativaC[idPergunta];
    respostaD.text = alternativaD[idPergunta];
}
void Update()
{
    Debug.Log("script responder");
}
public void resposta (string alternativa)
{
    if (alternativa == "A")
    {
        if (alternativaA[idPergunta] == corretas[idPergunta])
        {
            acertos += 1;
```

```
        Pontuacao.scoreTotalT++;
    }
}
else if (alternativa == "B")
{
    if (alternativaB[idPergunta] == corretas[idPergunta])
    {
        acertos += 1;
        Pontuacao.scoreTotalT++;
    }
}
else if (alternativa == "C")
{
    if (alternativaC[idPergunta] == corretas[idPergunta])
    {
        acertos += 1;
        Pontuacao.scoreTotalT++;
    }

}
else if (alternativa == "D")
{
    if (alternativaD[idPergunta] == corretas[idPergunta])
    {
```

```
            acertos += 1;
            Pontuacao.scoreTotalT++;
        }
    }
    proximaPergunta();
}
void proximaPergunta ()
{
    idPergunta += 1;

    if (idPergunta <= (questoes - 1))
    {
        questao4.SetActive(false);
        questao5.SetActive(false);
        questao6.SetActive(false);
        questao7.SetActive(false);
        apagador.SetActive(false);
        pergunta.text = perguntas[idPergunta];
        respostaA.text = alternativaA[idPergunta];
        respostaB.text = alternativaB[idPergunta];
        respostaC.text = alternativaC[idPergunta];
        respostaD.text = alternativaD[idPergunta];

        if (idPergunta == 3)
```

```
        {
            questao4.SetActive(true);
            apagador.SetActive(true);
        }
        if (idPergunta == 5)
        {
            questao5.SetActive(true);
            apagador.SetActive(true);
        }
        if (idPergunta == 6)
        {
            questao6.SetActive(true);
            apagador.SetActive(true);
        }
        if (idPergunta == 7)
        {
            questao7.SetActive(true);
        }

    }
    else //O QUE FAZER SE ACABOU AS PERGUNTAS
    {
        media = 10 * (acertos / questoes); // CALCULA A MEDIA COM BASE NO
PERCENTUAL DE ACERTO
```

```
        Pontuacao.notaFinalT += acertos * CharacterSelectionMenu.exp; //ARREDONDA A
NOTA PARA O PROXIMO INTEIRO

        // notaFinal += Pontuacao.scoreTotalT;

        if (notaFinal > PlayerPrefs.GetInt("notaFinal" + idTema.ToString())) // GRAVA
NOVO RECORD SE ELE FOR MAIOR QUE O ANTERIOR

        {

            PlayerPrefs.SetInt("notaFinal" + temaJogo.idTema.ToString(), acertos); //gravado
quando bate o record

            PlayerPrefs.SetInt("acertos" + temaJogo.idTema.ToString(), (int)acertos);

        }

        PlayerPrefs.SetInt("notaFinalTemp" + temaJogo.idTema.ToString(), acertos);
//sempre sao gravados

        PlayerPrefs.SetInt("acertosTemp" + temaJogo.idTema.ToString(), (int)acertos);

        Debug.Log("acertos teste " + acertos);

        Debug.Log("notaFinal responder " + notaFinal);

        avaliarProjeto.notaFinal = 0;

        acertos = 0;

        SceneManager.LoadScene("ResultadoT" + temaJogo.idTema.ToString());

    }

  }

}
```

- **avaliarProjeto.cs**

```
using System.Collections;

using System.Collections.Generic;
```

```
using UnityEngine;
using UnityEngine.UI;

public class avaliarProjeto : MonoBehaviour
{

    public Text tempoTxt;
    public Text tempoParadoTxt;
    public Text expTxt;
    public Text txtResultadoP0;
    private int idTema;
    public Text pergunta;
    public Text respostaA;
    public Text respostaB;
    public Text respostaC;
    public Text respostaD;
    public Text infoRespostas;
    public Text txtNota;
    public string[] perguntas;      // armazena todas as perguntas
    public string[] alternativaA;   // armazena todas as alternativas A
    public string[] alternativaB;   // armazena todas as alternativas B
    public string[] alternativaC;   // armazena todas as alternativas C
    public string[] alternativaD;   // armazena todas as alternativas D
    public string[] corretas;       // armazena todas as alternativas corretas
```

```
private int idPergunta;
private int acertos;
private float questoes;
public static float media;
public static int notaFinal;
public GameObject Resultado;
public GameObject caixaMsg;
public GameObject questaofinal1;
public GameObject questaofinal2;
public GameObject questaofinal3;
public GameObject questaofinal4;
public GameObject questaofinal5;

public GameObject apagadorfinal;

void Start()
{
    CharacterSelectionMenu.tempo = CharacterSelectionMenu.fixedTempo;
    tempoParadoTxt.text = CharacterSelectionMenu.tempo.ToString();
    expTxt.text = CharacterSelectionMenu.exp.ToString();
    //expTxt.text = CharacterSelectionMenu.exp.ToString();
    idTema = 1;
    idPergunta = 0;
    questoes = perguntas.Length;
```

```
    pergunta.text = perguntas[idPergunta];
    respostaA.text = alternativaA[idPergunta];
    respostaB.text = alternativaB[idPergunta];
    respostaC.text = alternativaC[idPergunta];
    respostaD.text = alternativaD[idPergunta];
}
void Update()
{
    /*Debug.Log("script avaliarProjeto");
    Debug.Log("acertos " + acertos);
    Debug.Log("media " + media);
    Debug.Log("notaFinal " + notaFinal);
    Debug.Log("idTema " + temaJogo.idTema);
    Debug.Log("exp " + CharacterSelectionMenu.exp); */
}
public void registrarTempo()
{
    Debug.Log("tempo " + CharacterSelectionMenu.tempo);
}
public void iniciaContagem()
{
    StartCoroutine("contagemRegressiva");
}
IEnumerator contagemRegressiva()
```

```
    {
        tempoTxt.text = CharacterSelectionMenu.tempo.ToString();
        yield return new WaitForSeconds(1);
        CharacterSelectionMenu.tempo -= 1;
        if(CharacterSelectionMenu.tempo > 0)
        {
            StartCoroutine("contagemRegressiva");
        }
        else
        {
            media = 100 / questoes * acertos; // CALCULA A MEDIA COM BASE NO
PERCENTUAL DE ACERTO
            notaFinal = Mathf.RoundToInt(acertos);
            Pontuacao.notaFinalP += acertos * CharacterSelectionMenu.exp;
            Pontuacao.highScoreP += Pontuacao.notaFinalP;
            //ARREDONDA A NOTA PARA O PROXIMO INTEIRO
            //Debug.Log("nota do multiplicador " + notaFinal);
            if (notaFinal > PlayerPrefs.GetInt("notaFinal" + idTema.ToString())) // GRAVA
NOVO RECORD SE ELE FOR MAIOR QUE O ANTERIOR
            {
                notaFinal = acertos;
                PlayerPrefs.SetInt("notaFinal" + idTema.ToString(), notaFinal); //gravado quando
bate o record
                PlayerPrefs.SetInt("acertos" + idTema.ToString(), (int)acertos);
            }
```

```
            notaFinal = Mathf.RoundToInt(acertos);
            PlayerPrefs.SetInt("notaFinalTemp" + idTema.ToString(), notaFinal); //sempre sao
gravados
            PlayerPrefs.SetInt("acertosTemp" + idTema.ToString(), (int)acertos);
            // temaJogo.idProjeto++;
            // temaJogo.idTema++;
            caixaMsg.SetActive(false);
            Resultado.SetActive(true);
            responder.notaFinal = 0;
            Debug.Log("notaFinal avaliarProjeto " + responder.notaFinal);
            temaJogo.aprovadoP++;
            //idPergunta = 0;
            txtResultadoP0.text = "Parabens você acertou " + media.ToString() +
        "% do projeto e obteve " + Pontuacao.notaFinalP.ToString() + " pontos de experiência";
            notaFinal = 0;
            CharacterSelectionMenu.tempo = CharacterSelectionMenu.fixedTempo;



            // O QUE ACONTECE SE O TEMPO ACABAR
        }
    }
    public void resposta(string alternativa)
    {
        if (alternativa == "A")
        {
```

```
    if (alternativaA[idPergunta] == corretas[idPergunta])
    {
        acertos += 1;
        Pontuacao.scoreTotalP++;
    }
}
else if (alternativa == "B")
{
    if (alternativaB[idPergunta] == corretas[idPergunta])
    {
        acertos += 1;
        Pontuacao.scoreTotalP++;
    }
}
else if (alternativa == "C")
{
    if (alternativaC[idPergunta] == corretas[idPergunta])
    {
        acertos += 1;
        Pontuacao.scoreTotalP++;
    }
}
else if (alternativa == "D")
{
```

```
        if (alternativaD[idPergunta] == corretas[idPergunta])
        {
            acertos += 1;
            Pontuacao.scoreTotalP++;
        }
    }

    proximaPergunta();
}
void proximaPergunta()
{
    idPergunta += 1;
    if (idPergunta <= questoes - 1)
    {
        questaofinal1.SetActive(false);
        questaofinal2.SetActive(false);
        questaofinal3.SetActive(false);
        questaofinal4.SetActive(false);
        questaofinal5.SetActive(false);
        apagadorfinal.SetActive(false);
        pergunta.text = perguntas[idPergunta];
        respostaA.text = alternativaA[idPergunta];
        respostaB.text = alternativaB[idPergunta];
        respostaC.text = alternativaC[idPergunta];
```

```
    respostaD.text = alternativaD[idPergunta];

    if (idPergunta == 1)
    {
        questaofinal2.SetActive(true);
        apagadorfinal.SetActive(true);
    }
    if (idPergunta == 2)
    {
        questaofinal3.SetActive(true);
        apagadorfinal.SetActive(true);
    }
    if (idPergunta == 3)
    {
        questaofinal4.SetActive(true);
        apagadorfinal.SetActive(true);
    }
    if (idPergunta == 4)
    {
        questaofinal5.SetActive(true);
        apagadorfinal.SetActive(true);
    }
}
```

```
    else //O QUE FAZER SE ACABOU AS PERGUNTAS
    {
        questaofinal1.SetActive(false);
        questaofinal2.SetActive(false);
        questaofinal3.SetActive(false);
        questaofinal4.SetActive(false);
        questaofinal5.SetActive(false);
        apagadorfinal.SetActive(false);
        //notaFinal = notaFinal * exp;
        media = 100 / questoes * acertos; // CALCULA A MEDIA COM BASE NO
PERCENTUAL DE ACERTO
        notaFinal = Mathf.RoundToInt(acertos);
        Pontuacao.notaFinalP += acertos * CharacterSelectionMenu.exp;
        Pontuacao.highScoreP += Pontuacao.notaFinalP;
            //ARREDONDA A NOTA PARA O PROXIMO INTEIRO
        //Debug.Log("nota do multiplicador " + notaFinal);
        if (notaFinal > PlayerPrefs.GetInt("notaFinal" + idTema.ToString())) // GRAVA
NOVO RECORD SE ELE FOR MAIOR QUE O ANTERIOR
        {
            notaFinal = acertos;
            PlayerPrefs.SetInt("notaFinal" + idTema.ToString(), notaFinal); //gravado quando
bate o record
            PlayerPrefs.SetInt("acertos" + idTema.ToString(), (int)acertos);
        }
```

```
            notaFinal = Mathf.RoundToInt(acertos);
            PlayerPrefs.SetInt("notaFinalTemp" + idTema.ToString(), notaFinal); //sempre sao
gravados
            PlayerPrefs.SetInt("acertosTemp" + idTema.ToString(), (int)acertos);
            // temaJogo.idProjeto++;
            // temaJogo.idTema++;
            caixaMsg.SetActive(false);
            Resultado.SetActive(true);
           responder.notaFinal = 0;
            Debug.Log("notaFinal avaliarProjeto " + responder.notaFinal);
            temaJogo.aprovadoP ++;
            //idPergunta = 0;
            txtResultadoP0.text = "Parabens você acertou " + media.ToString() +
      "% do projeto e obteve " + Pontuacao.notaFinalP.ToString() + " pontos de experiência";
            notaFinal = 0;
            CharacterSelectionMenu.tempo = CharacterSelectionMenu.fixedTempo;
        }
    }
}
```

- **notaFinal.cs**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class notaFinal : MonoBehaviour
{
    public Text txtNota;
    public GameObject ResultadoT1;
    public GameObject RepetirT1;
    public GameObject AvancarT1;
    public static int notaF;
    public static int acertos;

    void Start()
    {
        notaF = PlayerPrefs.GetInt("notaFinalTemp" + temaJogo.idTema.ToString());
        acertos = PlayerPrefs.GetInt("acertosTemp" + temaJogo.idTema.ToString());
        if (acertos > responder.questoes / 2)
        {
            Debug.Log("ENTREOU");
            ResultadoT1.SetActive(true);
            ResultadoT1.SetActive(true);
            AvancarT1.SetActive(true);
            RepetirT1.SetActive(false);
            temaJogo.idTema++;
```

```
            txtNota.text = "Você foi muito bem no periodo de treinamento, obteve "
                + acertos.ToString() + " acertos de " + responder.questoes.ToString();
        }
        else
        {
            ResultadoT1.SetActive(true);
            AvancarT1.SetActive(false);
            RepetirT1.SetActive(true);
            Pontuacao.scoreTotalT =- responder.media;
            txtNota.text = "Você precisa se esforçar no periodo de treinamento! obteve "
                + acertos.ToString() + " acertos de " + responder.questoes.ToString();
        }
        Debug.Log("acertos" + acertos);
        Pontuacao.notaFinalT = 0;
    }
    public void jogarNovamente()
    {
        SceneManager.LoadScene("T" + temaJogo.idTema.ToString());
    }
}
```

- **Pontuacao.cs**

```
using System.Collections;
```

```
using System.Collections.Generic;
using UnityEngine;

public class Pontuacao : MonoBehaviour
{
    public static int scoreTotalP;
    public static int scoreTotalT;
    public static int notaFinalT;
    public static int notaFinalP;
    public static int highScoreP;

    void Update()
    {
        Debug.Log("Score de Prova" + scoreTotalP);
        //Debug.Log("Score de Treinamento " + scoreTotalT);
        Debug.Log("Score de Prova Final" + notaFinalP);
        //Debug.Log("Score de Treinamento Final" + notaFinalT);
    }
    public void zerarPontos()
    {
    }
}
```

- **HighScores.cs**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;


public class HighScores : MonoBehaviour
{
    const string privateCode = "dFM7dHDEdEK52qAukJ7v1AhgAhyigvbU-zGVBQIzMRxg";
//Key to Upload New Info
    const string publicCode = "612863328f40bb6e98b52d53";   //Key to download
    const string webURL = "https://www.dreamlo.com/lb/"; //  Website the keys are for
    public PlayerScore[] scoreList;
    DisplayHighscores myDisplay;
    static HighScores instance; //Required for STATIC usability
    void Awake()
    {
        instance = this; //Sets Static Instance
        myDisplay = GetComponent<DisplayHighscores>();
    }


    public static void UploadScore(string username, int score)  //CALLED when Uploading
new Score to WEBSITE
    {//STATIC to call from other scripts easily
        instance.StartCoroutine(instance.DatabaseUpload(username, score)); //Calls Instance
    }
```

```
IEnumerator DatabaseUpload(string userame, int score) //Called when sending new score to Website
{
    WWW www = new WWW(webURL + privateCode + "/add/" + WWW.EscapeURL(userame) + "/" + score);
    yield return www;
    if (string.IsNullOrEmpty(www.error))
    {
        print("Upload Successful");
        DownloadScores();
    }
    else print("Error uploading" + www.error);
}
public void DownloadScores()
{
    StartCoroutine("DatabaseDownload");
}
IEnumerator DatabaseDownload()
{
    //WWW www = new WWW(webURL + publicCode + "/pipe/"); //Gets the whole list
    WWW www = new WWW(webURL + publicCode + "/pipe/0/10"); //Gets top 10
    yield return www;
    if (string.IsNullOrEmpty(www.error))
    {
        OrganizeInfo(www.text);
```

```
            myDisplay.SetScoresToMenu(scoreList);
            print("Download Successful");
        }
        else print("Error uploading" + www.error);
    }
    void OrganizeInfo(string rawData) //Divides Scoreboard info by new lines
    {
        string[] entries = rawData.Split(new char[] { '\n' },
System.StringSplitOptions.RemoveEmptyEntries);
        scoreList = new PlayerScore[entries.Length];
        for (int i = 0; i < entries.Length; i++) //For each entry in the string array
        {
            string[] entryInfo = entries[i].Split(new char[] { '|' });
            string username = entryInfo[0];
            int score = int.Parse(entryInfo[1]);
            scoreList[i] = new PlayerScore(username, score);
            print(scoreList[i].username + ": " + scoreList[i].score);
        }
    }
}
public struct PlayerScore //Creates place to store the variables for the name and score of each
player
{
    public string username;
    public int score;
```

```
    public PlayerScore(string _username, int _score)
    {
        username = _username;
        score = _score;
    }
}
```

- **DisplayHighscores.cs**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class DisplayHighscores : MonoBehaviour
{
    public TMPro.TextMeshProUGUI[] rNames;
    public TMPro.TextMeshProUGUI[] rScores;
    HighScores myScores;

    void Start() //Fetches the Data at the beginning
    {
        for (int i = 0; i < rNames.Length;i ++)
        {
            rNames[i].text = i + 1 + ". Fetching...";
```

```
    }
    myScores = GetComponent<HighScores>();
    StartCoroutine("RefreshHighscores");
  }
  public void SetScoresToMenu(PlayerScore[] highscoreList) //Assigns proper name and score for each text value
  {
    for (int i = 0; i < rNames.Length;i ++)
    {
      rNames[i].text = i + 1 + ". ";
      if (highscoreList.Length > i)
      {
        rScores[i].text = highscoreList[i].score.ToString();
        rNames[i].text = highscoreList[i].username;
      }
    }
  }
  IEnumerator RefreshHighscores() //Refreshes the scores every 30 seconds
  {
    while(true)
    {
      myScores.DownloadScores();
      yield return new WaitForSeconds(30);
    }
```

```
    }
}
```

6. ANEXO A – FORMULÁRIO DE AVALIAÇÃO

Questionário para a avaliação da qualidade de jogos

Nome do jogo: _____

Gostaríamos que você respondesse as questões abaixo sobre a sua percepção da qualidade do jogo para nos ajudar a melhorá-lo. Todos os dados são coletados anonimamente e somente serão utilizados no contexto desta pesquisa. Algumas fotografias poderão ser feitas como registro desta atividade, mas não serão publicadas em nenhum local sem autorização.

Nome do pesquisador responsável: _____
Local e data: _____

Informações Demográficas	
Instituição:	
Curso:	
Disciplina:	
Faixa etária:	☐ Menos de 18 anos ☐ 18 a 28 anos ☐ 29 a 39 anos ☐ 40 a 50 anos ☐ Mais de 50 anos
Sexo:	☐ Masculino ☐ Feminino
Com que frequência você costuma jogar jogos digitais?	☐ Nunca: nunca jogo. ☐ Raramente: jogo de tempos em tempos. ☐ Mensalmente: jogo pelo menos uma vez por mês. ☐ Semanalmente: jogo pelo menos uma vez por semana. ☐ Diariamente: jogo todos os dias.
Com que frequência você costuma jogar jogos não-digitais (de cartas, tabuleiro, etc.)?	☐ Nunca: nunca jogo. ☐ Raramente: jogo de tempos em tempos. ☐ Mensalmente: jogo pelo menos uma vez por mês. ☐ Semanalmente: jogo pelo menos uma vez por semana. ☐ Diariamente: jogo todos os dias.

Por favor, **marque uma opção** de acordo com o quanto você concorda ou discorda de cada afirmação abaixo.

Usabilidade					
Afirmações	**Marque uma opção conforme sua avaliação**				
	Discordo totalmente	**Discordo**	**Nem discordo, nem concordo**	**Concordo**	**Concordo totalmente**
O design do jogo é atraente (tabuleiro, cartas, interfaces, gráficos, etc.).	☐	☐	☐	☐	☐
Os textos, cores e fontes combinam e são consistentes.	☐	☐	☐	☐	☐
Eu precisei aprender poucas coisas para poder começar a jogar o jogo.	☐	☐	☐	☐	☐
Aprender a jogar este jogo foi fácil para mim.	☐	☐	☐	☐	☐
Eu acho que a maioria das pessoas aprenderiam a jogar este jogo rapidamente.	☐	☐	☐	☐	☐
Eu considero que o jogo é fácil de jogar.	☐	☐	☐	☐	☐
As regras do jogo são claras e compreensíveis.	☐	☐	☐	☐	☐
As fontes (tamanho e estilo) utilizadas no jogo são legíveis.	☐	☐	☐	☐	☐
As cores utilizadas no jogo são compreensíveis.	☐	☐	☐	☐	☐

Experiência do Jogador					
Afirmações	Marque uma opção conforme sua avaliação				
	Discordo totalmente	Discordo	Nem discordo, nem concordo	Concordo	Concordo totalmente
A organização do conteúdo me ajudou a estar confiante de que eu iria aprender com este jogo.	☐	☐	☐	☐	☐
Este jogo é adequadamente desafiador para mim.	☐	☐	☐	☐	☐
O jogo oferece novos desafios (oferece novos obstáculos, situações ou variações) com um ritmo adequado.	☐	☐	☐	☐	☐
O jogo não se torna monótono nas suas tarefas (repetitivo ou com tarefas chatas).	☐	☐	☐	☐	☐
Completar as tarefas do jogo me deu um sentimento de realização.	☐	☐	☐	☐	☐
É devido ao meu esforço pessoal que eu consigo avançar no jogo.	☐	☐	☐	☐	☐
Me sinto satisfeito com as coisas que aprendi no jogo.	☐	☐	☐	☐	☐
Eu recomendaria este jogo para meus colegas.	☐	☐	☐	☐	☐
Eu pude interagir com outras pessoas durante o jogo.	☐	☐	☐	☐	☐
O jogo promove momentos de cooperação e/ou competição entre os jogadores.	☐	☐	☐	☐	☐
Eu me senti bem interagindo com outras pessoas durante o jogo.	☐	☐	☐	☐	☐
Eu me diverti com o jogo.	☐	☐	☐	☐	☐
Aconteceu alguma situação durante o jogo (elementos do jogo, competição, etc.) que me fez sorrir	☐	☐	☐	☐	☐
Houve algo interessante no início do jogo que capturou minha atenção.	☐	☐	☐	☐	☐
Eu estava tão envolvido no jogo que eu perdi a noção do tempo.	☐	☐	☐	☐	☐
Eu esqueci sobre o ambiente ao meu redor enquanto jogava este jogo.	☐	☐	☐	☐	☐
O conteúdo do jogo é relevante para os meus interesses.	☐	☐	☐	☐	☐
É claro para mim como o conteúdo do jogo está relacionado com a disciplina.	☐	☐	☐	☐	☐
O jogo é um método de ensino adequado para esta disciplina.	☐	☐	☐	☐	☐
Eu prefiro aprender com este jogo do que de outra forma (outro método de ensino).	☐	☐	☐	☐	☐
O jogo contribuiu para a minha aprendizagem na disciplina.	☐	☐	☐	☐	☐
O jogo foi eficiente para minha aprendizagem, em comparação com outras atividades da disciplina.	☐	☐	☐	☐	☐

O que você mais gostou no jogo?_____
_____
_____
_____

O que poderia ser melhorado no jogo?_____
_____
_____
_____

Gostaria de fazer mais algum comentário?_____
_____
_____
_____

Muito obrigado pela sua contribuição!