Agile Game Development Process: A Game Development Process

ABSTRACT: The search for processes and good practices that support the electronic games development is increasingly common. However, although there is a lot of information about these methodologies aimed at traditional computer systems, there is little data in the literature about these approaches for the context of digital games and, when they exist, they are insufficient or very prolific, making an application in real environments unfeasible. This work presents the Agile Game Development Process, a proposal for a process for the digital games development, which combines aspects of other processes found in bibliographic surveys with some agile practices. Hence, the models identified in the literature were examined, extracting the best practices for the context of electronic games in order to build the proposed process. The article elucidates the flows, actors, tasks, artifacts and technologies in a succinct way in order to make the implementation viable.

Keywords: Digital Games, Game Development, Agile Methodology, Software Process

Agile Game Development Process: Um Processo para o Desenvolvimento de Jogos

RESUMO: A procura por processos e boas práticas que apoiem o desenvolvimento de jogos eletrônicos está cada vez mais comum. Entretanto, embora existam muitas informações de tais metodologias voltadas para sistemas computacionais tradicionais, há poucos dados na literatura sobre essas abordagens para o contexto dos jogos digitais e, quando existem, são insuficientes ou muito prolixas, inviabilizando a aplicação em ambientes reais. Esse trabalho apresenta o Agile Game Development Process, uma proposta de um processo para o desenvolvimento de jogos digitais a qual combina aspectos de outros processos encontrados em levantamentos bibliográficos com algumas práticas ágeis. Por conseguinte, foram examinados os modelos identificados na literatura, extraindo as melhores práticas para o contexto dos jogos eletrônicos de modo a construir o processo proposto. O artigo elucida os fluxos, atores, tarefas, artefatos e tecnologias de modo sucinto com o intuito de oportunizar a implementação.

Palavras-chave: Jogos Digitais, Desenvolvimento de Jogos, Metodologia Ágil, Processo de Software.

Agradecimentos: O autor deseja agradecer ao PIBIC/CNPq pela concessão de bolsa de iniciação científica para a condução desta pesquisa. Este trabalho pertence ao projeto SPIDER (*http://www.spider.ufpa.br*).

1 INTRODUÇÃO

É fato que, cada vez mais, os jogos eletrônicos e o mercado de *games* estão ganhando alcance no Brasil e no mundo. No que tange ao consumo, a edição de 2020 da Pesquisa Game Brasil (PGB), realizada pela *Sioux Group*, uma agência nacional de tecnologia interativa, constatou que cerca de 73% dos brasileiros são jogadores de *games*, incluindo diferentes faixas etárias, plataformas, gêneros, estilos de jogabilidade (seja casual ou *hardcore*), entre uma série de outros aspectos. Em complemento, em relação às questões mercadológicas, uma pesquisa realizada pela *Newzoo*, uma empresa de consultoria da área de jogos, revelou que, em 2018, o Brasil movimentou aproximadamente US$ 1,5 bilhão e que a nação ocupava a 13ª colocação mundial no mercado de games, sendo líder na América Latina.

Concomitante a isso, para desenvolver *games* com qualidade e eficiência, faz-se indispensável o emprego de boas práticas no Desenvolvimento de Jogos Digitais. A Engenharia de Software, área do conhecimento que estrutura cientificamente um conjunto de técnicas, métodos e ferramentas para dar suporte ao desenvolvimento de sistemas computacionais, é crucial nesse cenário, devendo ser adaptada para as características específicas da criação de jogos, como equipes multidisciplinares e produção de conteúdos audiovisuais.

Outros levantamentos mostram que o número de Desenvolvedores de Jogos Eletrônicos cresce a cada ano no Brasil - segundo um censo de 2018 da Indústria Brasileira de Jogos Digitais, financiado a partir de um acordo entre o Ministério da Cultura e a Unesco, no início de tal ano, existiam cerca de 375 (trezentos e setenta e cinco) Desenvolvedoras de Jogos digitais no país, aproximadamente o dobro da quantidade identificada no censo anterior, datado de 2014. No entanto, ressalta-se que muitas dessas empresas, principalmente as pequenas e médias, encontram desafios na execução e organização de projetos de *games*, falhando em seguir limitações de orçamentos, cronogramas e escopo, por exemplo.

Diante da problemática exposta, realizou-se uma revisão na literatura sobre Processos de Jogos Eletrônicos e, embora alguns tenham sido encontrados, foram identificadas informações prolixas ou em pequena quantidade para estes, consequentemente, dificultando a implementação em ambientes reais.

Assim sendo, o objetivo principal deste trabalho é propor, apresentar e elucidar um Processo Ágil para dar suporte ao Desenvolvimento de Jogos Digitais de forma eficaz, baseando-se no *Game Waterfall Process*, no *Extreme Game Development* (XGD) e no *Scrum*, outros 3 (três) processos com essa mesma orientação encontrados no levantamento bibliográfico mencionado. O autor denominou o modelo de *Agile Game Development Process*.

O trabalho tem seu início com o aprofundamento de conceitos relacionados ao Desenvolvimento de Games, à Metodologia Ágil e aos processos que foram utilizados como base para o *Agile Game Development Process*. Além disso, foram analisados trabalhos relacionados, a saber propostas de metodologias similares, fazendo-se uma comparação com o diferencial deste trabalho.

Em sequência, *o Agile Game Development Process* foi explicitado, apresentando-se detalhadamente o fluxo, os atores, as tarefas, os artefatos e tecnologias recomendadas. Após tais especificações, foi analisada a aderência em relação aos elementos das 3 (três) abordagens nas quais o processo proposto foi baseado.

As próximas seções deste trabalho estão organizadas da seguinte forma: na Seção 2 é apresentada uma contextualização referente à fundamentação teórica utilizada; na Seção 3 tem-se a análise de alguns trabalhos relacionados; a Seção 4 explicita o processo; a Seção

5 retrata a aderência do *Agile Game Development Process* em relação aos processos encontrados; e, por fim, a Seção 6 conclui o trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção, alguns conceitos sobre o contexto serão abordados para uma melhor caracterizaçãoo da pesquisa relatada neste artigo.

2.1 Desenvolvimento de Jogos Digitais

Segundo SALEN e ZIMMERMAN (2004), "Um jogo é um sistema no qual os jogadores se envolvem em um conflito artificial, definido por regras, que implica um resultado quantificável". Nesse sentido, entende-se que um jogo representa um ambiente de interatividade no qual tem-se a figura dos jogadores, que necessitam solucionar conflitos fictícios, seguindo algumas restrições predefinidas, com um determinado objetivo, ao mesmo tempo em que as ações tomadas nesse sistema implicam na geração de dados contáveis que determinarão um resultado, podendo ser representado pela vitória ou derrota, por exemplo. Vale ressaltar que os jogos apresentam diversos gêneros, como RPG, ação, aventura, estratégia, entre outros.

Por outro lado, o Jogo Digital ou Jogo Eletrônico ocorre quando essa dinâmica é trazida para os meios digitais ou eletrônicos. Segundo os autores citados no parágrafo anterior, nesses meios, as características são mais complexas devido a presença de fatores adicionais, a exemplo da comunicação em redes e da interatividade imediata ao mesmo tempo que mais restrita.

Em adição aos conceitos apresentados, o Desenvolvimento de Jogos Digitais ou Desenvolvimento de Jogos Eletrônicos representa o processo de produção de tais jogos. Assim sendo, é de suma importância o entendimento a respeito dos Papéis Envolvidos no processo em questão e do Ciclo de Vida dos Jogos.

De acordo com ROGERS (2013), existem 8 (oito) Papéis Principais Envolvidos no Desenvolvimento de Jogos: Programador, Artista, *Designer*, Produtor, Testador, Compositor, *Sound Designer* e Redator.

- Programador: Com foco nas funcionalidades, é o ator responsável pela elaboração do código-fonte;
- Artista: É o ator responsável pela confecção dos elementos visuais, tais como personagens e cenários;
- *Designer*: Representa o projetista do jogo, sendo encarregado da definição das regras, sistema de pontuação, *design* das armas, além de outros aspectos do projeto do game;
- Produtor: Atua como um Gerente de Projetos, supervisionando os demais atores, planejando e monitorando todas as áreas e fases do projeto;
- Testador: Faz a execução dos testes, apurando inconsistências de código e avaliando fatores de diversão, com o intuito de assegurar qualidade no produto;
- Compositor: É responsável pela confecção das trilhas sonoras;
- *Sound Designer*: É responsável pela produção dos efeitos sonoros;
- Redator: Escreve o roteiro e a história que se passam no *game*.

Segundo SLOPER (2002), o processo de Desenvolvimento de Jogos constitui-se de 5 (cinco) etapas principais: Concepção, Pré-Produção, Produção, Pós-Produção e Pós-Lançamento.

- Concepção: Nesta etapa a Empresa Publicadora realizará uma análise de viabilidade para o projeto com o objetivo de definir um conceito de jogo o qual seja

bem sucedido no mercado, de forma a maximizar os lucros com o game. Todas as decisões tomadas em tal fase serão registradas em um artefato denominado Documento Conceitual (*Conceptual Document*);

- Pré-Produção: Em seguida, ocorrerá a especificação dos elementos precisos para desenvolver o game. Nessa fase, gera-se o Documento de Game Design (*Game Design Document*), no qual são apresentadas características áudio-visuais e mecânicas do *game* e o Documento de Design Técnico (*Technical Design Document*), em que são registradas especificações técnicas do desenvolvimento, a exemplo das ferramentas ou tecnologias que serão utilizadas;
- Produção: É a fase de elaboração do código-fonte e dos elementos áudio-visuais, seguindo às documentações geradas nas etapas anteriores. Geram-se mais 2 (dois) artefatos, que passam por muitas atualizações: o Documento de Arte (*Art List Document*) e o Documento de Som (*Sound List Document*). Enquanto o primeiro apresenta os elementos visuais a serem produzidos pela equipe de arte, desde armas até cenários, o segundo apresenta os elementos sonoros a serem produzidos pela equipe de som, isto é, as trilhas e efeitos sonoros;
- Pós-Produção: Após a Produção, ocorre a geração de um Plano de Testes, contendo um guia para a execução de testes sobre o produto, voltados tanto para a equipe de desenvolvimento quanto para os jogadores ou consumidores finais. Somado a isso, são levantados pontos positivos, negativos e lições aprendidas do projeto e registrados em um Documento de *Post Mortem*, visando minimizar erros em projetos futuros;
- Pós-lançamento: Por fim, com o produto no mercado, são executadas as atividades de Marketing e são acompanhadas as reações do público. A partir disso, é estabelecido um canal de comunicação para sanar dúvidas dos consumidores sobre a operação do jogo e a publicadora analisa possibilidades de lançamentos de mais versões, seja para outras línguas ou para outras plataformas, mais sequências, isto é, um novo jogo com a continuação da história ou mais expansões, ou seja, customizações especiais para o jogo já lançado.

2.2 Metodologia Ágil

A Metodologia Ágil surgiu nos últimos anos, no contexto da economia moderna, onde as condições mercadológicas alteram-se rapidamente, os clientes mudam de ideia a respeito do produto de *software* e ameaças competitivas de sistemas computacionais similares surgem sem aviso prévio. Nessa lógica, o desenvolvimento ágil de software tenta evitar os pontos fracos da Engenharia de *Software* tradicional, uma vez que os processos sequenciais possuem baixa flexibilidade e alto custo para a gestão de mudanças.

No ano de 2001, Kent Beck e outros 16 (dezesseis) renomados desenvolvedores de *software* se agruparam e assinaram o "Manifesto para o Desenvolvimento Ágil de *Software*", o qual reúne 4 (quatro) valores:

- Indivíduos e interações acima de processos e ferramentas;
- Software operacional acima de documentação completa;
- Colaboração dos clientes acima de negociação contratual;
- Resposta a mudanças acima de seguir um plano.

Sendo assim, a Metodologia Ágil pode ser entendida como um conjunto de processos, atividades, ferramentas e comportamentos os quais seguem as orientações do Manifesto Ágil. Nessa conjuntura, ao invés de seguir uma sequência de etapas fixas, passando por análise, implementação, testes e manutenção, a Metodologia Ágil se caracteriza por ser incremental ou iterativa. O projeto é dividido em ciclos rápidos, nos

quais as equipes fazem entregas funcionais incrementais ao cliente, registrando o *feedback* e adaptando o produto com mais fluidez e dinamicidade às mudanças requisitadas. Em comparação com a Engenharia de *Software* Tradicional, os projetos são mais simplificados, a colaboração entre equipes e clientes é muito maior, a entrega de valor ocorre mais rapidamente e a burocracia, bem como os custos com mudanças de escopo, são mais reduzidos. A Programação Extrema (XP), o *Scrum* e o *Crystal* são exemplos de Modelos de Processos Ágeis.

Faz-se imprescindível acrescentar que PRESSMAN (2011) afirma que a nova realidade trazida pela economia moderna e a consequente demanda por agilidade não anulam princípios e fundamentos da Engenharia de *Software* estudados antes desse panorama, posto que tais quais as demais disciplinas da Engenharia, a Engenharia de *Software* pode evoluir e se adaptar às mudanças de cenário postas em cheque.

2.3 Modelos de Processo para o Desenvolvimento de Jogos

Nesse tópico, serão analisados 3 (três) modelos de processo para o desenvolvimento de jogos eletrônicos, que foram encontrados a partir de revisão na literatura, sendo respectivamente: *Game Waterfall Process*, *Extreme Game Development* (XGD) e *Scrum*.

2.3.1 *Game Waterfall Process*

O *Game Waterfall Process* ou Processo Cascata para Jogos consiste na adaptação do tradicional Modelo Cascata para o contexto de Desenvolvimento de Jogos Digitais. Nesse sentido, de modo similar ao processo para *softwares* convencionais, todas as tarefas do projeto seguem sequencialmente, uma após a outra, somado a característica de que o *Game Waterfall Process* é baseado no Ciclo de Vida dos Jogos, explicado na seção 2.1. Assim sendo, as etapas do Processo Cascata para Jogos são 5 (cinco): Concepção, Pré-Produção, Produção, Pós-Produção e Pós-Lançamento, em que são seguidas as mesmas recomendações de documentações e procedimentos descritos na seção anterior. De modo análogo à metodologia tradicional, tem-se: Requisitos, Projeto, Implementação, Testes e Manutenção.

2.3.2 *Extreme Game Development* (XGD)

O *Extreme Game Development* (XGD) (DEMACHY, 2013) é uma Metodologia Ágil para o desenvolvimento de jogos baseada na Programação Extrema ou *Extreme Programming* (XP). Nessa perspectiva, o foco do XGD está em adaptar os valores e práticas do XP para o contexto dos *games*, dado que a Programação Extrema foi criada por programadores e as equipes de desenvolvimento de jogos são multidisciplinares, sendo compostas por artistas, músicos, *game designers*, entre outros novos atores. Além disso, diferentemente dos *softwares* convencionais, os jogos digitais contam com conteúdos multimídia e fatores de divertimento dos jogadores.

No que concerne aos valores do XGD, de forma similar ao XP, tem-se simplicidade, comunicação, *feedback*, coragem e respeito. Simplicidade, pois o código-fonte do jogo deve ser funcional da forma mais simples possível, sobretudo se não for reutilizado; Comunicação, já que é um fator crucial com as equipes multidisciplinares, sendo relevante priorizar conversas orais e registro de documentos os mais curtos possíveis, a exemplo de reduzir a complexidade de um Documento de *Game Design*, de modo que os programadores consigam transformar as ideias em linhas de código; *Feedback*, de modo a obter respostas rapidamente tanto do processo com os times, quanto do *game* com a Empresa Publicadora, visando reduzir custos de possíveis adaptações; Coragem, para que as equipes aceitem *feedbacks* regulares e eventuais críticas da

Publicadora; Respeito, entre todos os envolvidos no processo, servindo como um valor de sustentação a todos os outros, porque caso não exista, o processo torna-se impraticável.

No que tange às práticas do processo referido, o *Extreme Game Development* manteve algumas das dezenas de práticas do *Extreme Programming*, com poucas alterações, algumas das principais são: *Whole Team*, Desenvolvimento Orientado a Testes e *User Stories*.

A prática de *Whole Team* significa que toda a equipe de desenvolvimento tem um nível de conhecimento semelhante do código-fonte, não havendo a noção de programadores especializados em partes, isto é, caso um desenvolvedor deixe o time, isso não gerará um grande impacto, pois os demais são competentes de fazerem esforços para substituí-lo. A mesma regra vale para os artistas e para os músicos, sendo preciso cuidado do gerente de projetos ao eleger um artista especializado em personagens ou um músico com especialização em efeitos sonoros.

A prática de Desenvolvimento Orientado a Testes visa antecipar a identificação e correção de falhas durante o desenvolvimento, com o uso massivo da abordagem de testes. No XGD, os testes são divididos em 2 (duas) abordagens principais: os Testes Funcionais, que serão conduzidos pelo gerente de projetos, pelo *game designer* e pelo testador, onde são escritos *scripts* específicos para cada requisito do jogo e tal funcionalidade é validada somente se o *script* for executado com êxito; e os Testes Unitários Automatizados, conduzidos pela equipe de desenvolvimento para garantir que cada unidade do código-fonte está funcionando como o esperado, valendo ressaltar que os artistas também podem executar esse tipo de teste com métricas quantitativas visuais, a exemplo da quantidade de pixels. Também é válido destacar que os testes manuais não são anulados, podendo ser aplicados em outros aspectos, como a jogabilidade.

Em conclusão, as *User Stories* representam uma forma simples e ágil de escrever os requisitos do *game* especificados pela Empresa Publicadora. As *User Storie*s também tornam mais fácil a definição dos testes funcionais mencionados no parágrafo anterior.

2.3.3 *Scrum*

O *Scrum* é um método ágil de gerência e planejamento de projetos de *software*. Embora seja voltado para sistemas computacionais mais convencionais e não para jogos, foram encontrados registros de implementação por algumas empresas de *games* em torno do mundo. É baseado em 3 (três) pilares fundamentais: transparência, inspeção e adaptação. Transparência, pois todos os papéis envolvidos têm conhecimento dos requisitos de entrega, dos procedimentos e do andamento do projeto de modo geral; Inspeção, posto que todas as atividades que estão sendo realizadas são monitoradas frequentemente; Adaptação, uma vez que tanto o produto como o próprio processo em si estão sujeitos a sofrer alterações de acordo com o cenário.

Primeiramente, no que diz respeito aos papéis envolvidos nessa metodologia, são 3 (três): *Product Owner*, *Scrum Maste*r e Time de Desenvolvimento. O *Product Owner* representa o ator responsável pela decisão sobre o produto e as funcionalidades que devem ser desenvolvidas, priorizando os requisitos e estabelecendo uma ordem para desenvolvimento. O *Scrum Master* consiste em um papel de liderança do processo, assegurando que todos os fundamentos do *Scrum* sejam devidamente aplicados e é de sua responsabilidade lidar com as dificuldades abordadas pela equipe da melhor maneira possível. Por fim, o Time de Desenvolvimento consta na equipe de desenvolvimento, encarregada na construção do produto em si. Essa equipe é auto-organizada, visto que os próprios membros tomam as decisões relativas à maneira como o desenvolvimento será feito.

Em relação aos artefatos, tendo em vista que o *Scrum* é um processo ágil, tem-se somente 2 (dois) obrigatórios: *Product Backlog* e *Sprint Backlog*. O primeiro refere-se à lista das funcionalidades requeridas para o produto, não havendo a necessidade de estar completo inicialmente, já que novos requisitos podem surgir com o tempo e serem adicionados nesse artefato posteriormente, o *Product Owner* faz uma priorização e especifica quais itens são os mais relevantes. O segundo diz respeito ao conjunto das tarefas que a equipe deverá desenvolver na próxima *Sprint* ou iteração de desenvolvimento do processo, sendo extraídas diretamente do *Product Backlog*.

Há mais 2 (dois) artefatos que geralmente são utilizados para dar suporte ao *Scrum*, porém não são obrigatórios: Quadros *Kanban* e *Release Burndown*. Os Quadros *Kanban* constituem-se em uma estratégia de visualização do fluxo de trabalho, nos quais diferentes status são colocados em colunas adjacentes, como "*Backlog*", "Em Progresso", "Testes" e Pronto" e as tarefas são catalogadas em notas e alocadas no status correspondente. O *Release Burndown*, por outro lado, é um gráfico cujo o eixo horizontal mostra os iterações ou *Sprints* e o vertical, a quantidade de trabalho ou esforço que ainda precisa ser empregado no início da cada *Sprint*, permitindo um acompanhamento do andamento do projeto.

Finalmente, vale pontuar os eventos chave do *Scrum*: *Sprint Planning*, Execução da *Sprint*, *Daily Scrum*, Revisão da *Sprint* e Retrospectiva da *Sprint*, a saber:

- *Sprint Planning*: Tendo a visão geral do produto, trazida pelo *Product Owner* e consequentemente, a elaboração do *Product Backlog* com as funcionalidades priorizadas, em conjunto com o *Scrum Master*, acontece o planejamento da *Sprint*, onde todos os atores estarão presentes para discutir e definir as tarefas que serão executadas na *Sprint* e para registrar no *Sprint Backlog*. Essa decisão é tomada seguindo a priorização definida e a velocidade dos membros do Time de Desenvolvimento;
- Execução da *Sprint*: Em seguida do *Sprint Planning*, é o evento de construção do produto, segundo os itens do *Sprint Backlog*. Em geral, ocorre em períodos fixos denominados de *time-box*, que variam de 2 (duas) até 4 (quatro) semanas;
- *Daily Scrum*: Reunião diária que se passa na Execução da *Sprint* com a participação do *Scrum Master* e do *Scrum Team*. 3 (três) questionamentos são feitos: "O que fez ontem?", "O que fará hoje?" e "Existem impedimentos?". Caso existam entraves, o *Scrum Master* deve agir em prol de encontrar alguma solução e otimizar a produtividade;
- Revisão da *Sprint*: Assim como no *Sprint Planning*, todos os atores estão presentes. Nesse evento, após o término da Execução da *Sprint*, toda a produção é validada e são avaliadas necessidades de mudanças no produto;
- Retrospectiva: Após a Revisão da *Sprint*, nessa etapa, são avaliadas necessidades de mudanças no processo, a partir do levantamento de pontos positivos, negativos e possíveis soluções para os problemas identificados.

3 TRABALHOS RELACIONADOS

Nesta seção, serão analisados pontos fortes e fracos de 2 (dois) Trabalhos Relacionados encontrados na literatura seguidos das explicações dos aspectos diferenciais deste trabalho.

3.1 *Agile Game Development With Scrum* (KEITH, 2010)

O livro em questão aborda uma adaptação do *Scrum* (aprofundado na seção 2.3.3) para o Desenvolvimento de Games. Entre os pontos fortes deste trabalho, apresenta muitas práticas eficientes para adequar a versão padrão do Processo Ágil referido para a criação de

jogos, como o *outsourcing* e lideranças para as equipes multidisciplinares. Além disso, detalha todos os fluxos, atores, etapas e artefatos que precisam ser executados em cada iteração do projeto. Enfim, considera que a distribuição de trabalho e esforço entre o total de iterações necessárias para finalizar o projeto não é igual, dividindo o total do projeto em fases:

- Conceito: As *Sprints* iniciais são mais curtas e dedicadas à geração de ideias e prototipagens, de forma a determinar um conceito para o jogo. Em outras palavras, o propósito é criar conhecimento para as equipes e os *stakeholders*, não é o momento de gerar valor para o consumidor;
- Pré-Produção: Em sequência, essa etapa é focada em estabelecer os elementos atrativos de divertimento, agregando valor e conhecimento sobre os custos de produção;
- Produção: As equipes desenvolvem o jogo com base nos pontos que foram identificados na pré-produção, melhorando o produto incrementalmente;
- Pós-Produção: As últimas *Sprints* são voltadas para ajustes, "polimento" e correção de erros. Nessa fase, a maior parte dos itens no *Backlog* representam tarefas desse tipo.

Não obstante, o livro detalha excessivamente o *Scrum* clássico (para desenvolvimento de softwares convencionais), tornando-se um guia exaustivo, em contrapartida com os princípios da Metodologia Ágil. Outro ponto fraco é o fato de que, embora apresente detalhadamente o que precisa ser feito dentro de cada incremento, tem-se somente uma visão fragmentada, em outras palavras, o *Scrum* é examinado em um dos capítulos e somente no próximo encontram-se as explicações das técnicas aplicadas, não havendo uma visão geral do processo unificado, tanto textual como em forma de um diagrama esquemático. Como último fator, não foram encontradas sugestões de ferramentas as quais possam apoiar o processo.

3.2 Game-Scrum: An Approach to Agile Game Development (GODOY E BARBOSA, 2010)

O artigo propõe o processo Game-Scrum, um Método Ágil para Desenvolvimento de Jogos Digitais baseado no Scrum e no *Extreme Programming* (XP). Primeiramente, um dos pontos fortes do trabalho é o fato de que o modelo é acessível tanto para profissionais mais experientes como para indivíduos com pouco conhecimento na área de Desenvolvimento de Jogos. Outro aspecto positivo é que foi aplicado na criação de um jogo educativo para o ensino de Engenharia de Software e obteve resultados satisfatórios. Por fim, similarmente ao trabalho de KEITH (2010), também estrutura o total do projeto em fases, sendo estas:

- Pré-Produção: As *Sprints* iniciais funcionarão como prototipagens e são realizadas para descobrir o conceito do jogo e esclarecer o fator de divertimento, ou seja, os atrativos do *game*, com o intuito de evitar que isso seja delegado para o desenvolvimento. Nessa etapa, será produzido o Documento de *Game Design*;
- Produção: Após a etapa de Pré-Produção, é necessário que o escopo esteja bem definido e que o Documento de *Game Design* seja traduzido no *Product Backlog*. Essas Sprints consistirão no desenvolvimento do jogo em si, tanto do código como dos elementos multimídia;
- Pós-Produção: As últimas *Sprints* consistirão em testes de jogabilidade para assegurar a qualidade do *game* e na criação de um Documento de *Post Mortem* para registrar pontos positivos, soluções, pontos negativos, problemas críticos, entre outras experiências do desenvolvimento para levar em consideração em projetos futuros.

Entretanto, ainda que o processo aponte como os ciclos estão divididos ao longo do tempo total do projeto, não há um diagrama esquemático para visualização. Além disso, o artigo não foca na estrutura de cada iteração, não apresentando detalhes do fluxo, atores, tarefas, procedimentos e artefatos de um determinado ciclo, apenas mencionando alguns desses elementos, como o Documento de *Post Mortem*. Finalmente, também não foram encontradas sugestões de tecnologias para dar suporte ao processo.

3.3 Diferencial deste Trabalho

Em relação aos 2 (dois) trabalhos citados, o trabalho em desenvolvimento possui a peculiaridade de evidenciar as iterações com um alto de grau de detalhamento.

Primeiramente, tem-se um diagrama esquemático na notação BPMN (*Business Process Model and Notation*) mostrando todos os elementos inerentes ao processo em um fluxo sequencial.

Somado a isso, há o detalhamento de cada uma das tarefas presentes no gráfico citado acima, com especificações dos atores responsáveis e artefatos de entrada e saída. Para alguns dos referidos artefatos, também foram definidas recomendações de padrões a serem seguidos, com os campos necessários e as informações a serem registradas em cada campo. Enfim, também existe uma relação de tecnologias as quais podem ser empregadas para a otimização do processo.

Vale ressaltar que o modelo também é flexível a prototipagens para as *Sprints* iniciais, quando o conceito do jogo não está muito bem definido, bastando executar as atividades que forem mais convenientes para as equipes na fase em que o projeto se encontra.

4 *AGILE GAME DEVELOPMENT PROCESS*

Nesta seção, serão abordados todos os aspectos referentes ao processo proposto neste trabalho, evidenciando o fluxo, os atores, as tarefas, os artefatos e as ferramentas que possam apoiar a metodologia.

4.1 O Fluxo

A Figura 1 mostra um diagrama esquemático sequencial do Agile *Game Development Process* sob a notação BPMN (*Business Process Model and Notation*), uma notação formal utilizada universalmente no paradigma de negócios para a modelagem e automatização de processos.

4.2 Papéis Recomendados

No *Agile Game Development Process*, são 5 (cinco) os papéis recomendados: *Product Owner*, *Scrum Master*, Equipe de Desenvolvimento, Equipe de Arte e Equipe de Som. Os atores escolhidos foram baseados majoritariamente no *Scrum*, modelo de processo descrito no item 2.3.2, com algumas alterações para a inserção na perspectiva dos *games*, a saber:

- PO (*Product Owner*): É o ator que, representando a empresa publicadora, é responsável pelas decisões sobre o jogo e sobre as funcionalidades as quais devem ser construídas, estabelecendo uma prioridade de desenvolvimento;
- *Scrum Master*: É o papel de liderança do processo. Deve garantir que todos os princípios e valores da metodologia ágil sejam seguidos e possui a função de agir como um facilitador para as equipes, ou seja, deve procurar soluções para os impasses levantados pelos membros. No contexto dos *games*, também é importante que o líder tenha conhecimentos sobre projetos de jogos;

- Equipe de Desenvolvimento: Os integrantes dessa equipe possuem a função de elaboração do código-fonte do jogo. Assim como no Scrum, tal grupo é auto-organizado, pois são os próprios membros que decidem como construir o produto;
- Equipe de Arte: Esse é um novo papel em relação à abordagem tradicional do Scrum. Os participantes dessa equipe são os encarregados de produzir os elementos visuais para o game, isto é, as imagens e as animações especiais. Tal qual a Equipe de Desenvolvimento, é um grupo auto-organizado;
- Equipe de Som: Outro autor ausente no Scrum clássico. Os integrantes desse grupo têm a incumbência de criar os elementos sonoros do jogo, ou seja, as faixas de música e os efeitos sonoros. Vale ressaltar que os participantes também são autônomos em relação ao método de trabalho.

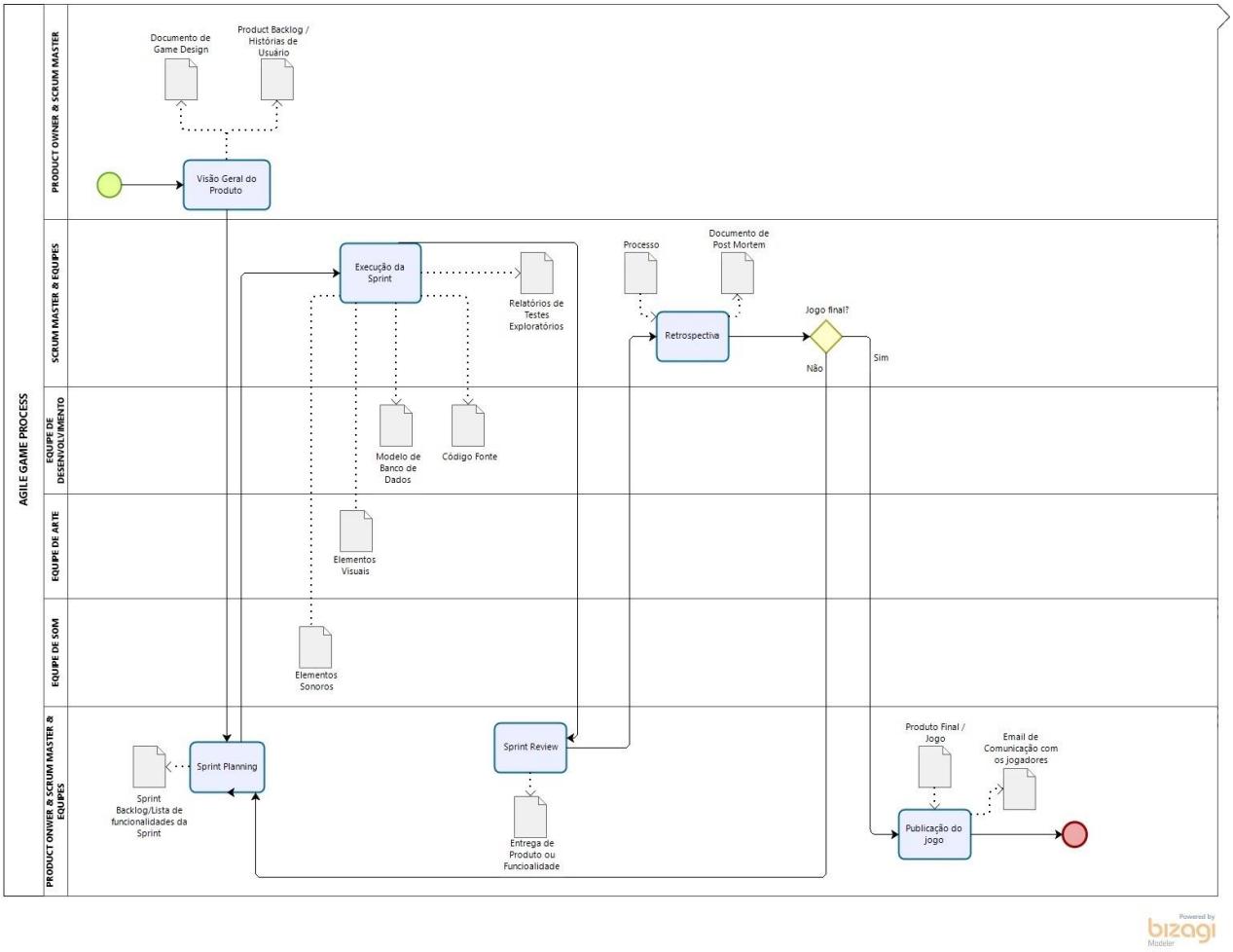


Figura 1 – Fluxo do Processo sob a notação BPMN (*Business Process Model and Notation*)
Fonte: Elaboração própria (2020).

4.3 Tarefas do Fluxo

No que diz respeito às tarefas do fluxo, conforme ilustradas nas figuras geométricas azuis na Figura 1, tem-se, respectivamente: Visão Geral do Produto, *Sprint Planning*, Execução da *Sprint*, *Sprint Review*, Retrospectiva e Publicação do Jogo. Estas tarefas serão melhor detalhadas nas seções a seguir.

4.3.1 Visão Geral do Produto

Primeiramente, o *Product Owner* apresenta uma modelagem de negócios, a qual ilustra, de forma mais inicial, o jogo que será desenvolvido. O Scrum Master analisa o material apresentado e, em conjunto com o PO, escrevem o Documento de Game Design, um artefato que cataloga características chave do game, como descrição dos personagens, cenários e missões, servindo como um guia para todo o desenvolvimento (esse e os demais artefatos serão apresentados com maior detalhamento na seção 4.3).

Tendo o Documento de Game Design, os 2 (dois) atores mencionados traduzem tais características para uma lista de funcionalidades, sob a sintaxe das *User Stories*, registrando no artefato *Product Backlog*. Serão incluídos requisitos não somente para a Equipe de Desenvolvimento, como também para as equipes de Arte e de Som. Também é preciso indicar a prioridade de desenvolvimento e os cenários de aceitação ou critérios de aceite (esses enunciados serão utilizados posteriormente para a validação) de cada uma das *User Stories*, anexando juntamente ao *Backlog* referido. Enfim, vale destacar que a recomendação do processo é que o *Product Backlog* seja uma coluna em um *Domain Expert Kanban*, um quadro onde será possível acompanhar o andamento do projeto como um todo.

4.3.2 *Sprint Planning*

Seguindo o fluxo, tem-se o *Sprint Planning*, etapa que reúne todos os atores do processo. De início, são identificados os requisitos mais prioritários do *Product Backlog*, para que cada uma das equipes, separadamente, estime a quantidade de esforço necessário para cumprir tais estórias.

Exemplificando, para uma determinada estória para a Equipe de Desenvolvimento de prioridade elevada, os membros desse time jogam *Planning Poker*, um jogo em que, para cada participante, são distribuídas as mesmas cartas, com os valores: 1, 2, 3, 5, 8, 13, 20, 40 e 100. Tais numerações correspondem a um nível de dificuldade em ordem crescente, em outras palavras, os cartões 1, 2 e 3 significam dificuldade baixa, 5 dificuldade média, 8 e 13 dificuldade elevada e 20, 40 e 100, dificuldade imensurável, levando a possíveis mudanças naquela funcionalidade ou divisão em 2 (duas) funcionalidades diferentes. Após o entendimento do requisito e iniciada uma rodada, o participante deve mostrar uma das cartas, havendo consenso entre todos, o valor daquela *User Story* fica definido como a numeração e não havendo, os desenvolvedores dialogam até que exista um acordo e uma nova rodada se inicia. Enfim, esse procedimento é repetido para todos os itens prioritários levantados para a Equipe de Desenvolvimento e para as outras 2 (duas) equipes, separadamente.

Em sequência da estimativa, levando em consideração a velocidade das equipes, a saber, a quantidade de pontos de *Planning Poker* que cada equipe consegue desenvolver em uma dada iteração, são acordadas as estórias que serão desenvolvidas no próximo ciclo e registradas no artefato *Sprint Backlog*. A recomendação do *Agile Game Development Process* é que o *Sprint Backlog* seja uma coluna no *Domain Expert Kanban*, possibilitando uma visualização do andamento geral do projeto em relação ao do ciclo e outra coluna em *Team Kanban*, quadro voltado para o monitoramento da realização das tarefas do novo incremento do *game*.

4.3.3 Execução da *Sprint*

A Execução da *Sprint* visa a elaboração de uma parte ou incremento do *game*, de acordo com os itens presentes no *Sprint Backlog* e sempre seguindo as diretrizes do Documento de *Game Design*, ocorrendo em um intervalo de tempo fixo denominado de *time box*, que

varia de 2 até 4 semanas. Todos os papéis recomendados participam da tarefa em questão, com exceção do PO.

No início de todos os dias da Sprint, os membros realizam uma reunião na qual necessitam responder 3 (três) perguntas básicas: “O que fez ontem?”, “O que fará hoje?” e “Há algum impedimento?”. Caso a resposta da terceira pergunta seja afirmativa, o Scrum Master deve ajudar o indivíduo a encontrar uma solução viável.

Nessa etapa do *Agile Game Development Process*, cabe à Equipe de Desenvolvimento identificar os dados necessários a serem armazenados, criar uma modelo de banco de dados e desenvolver o código-fonte nas tecnologias escolhidas. A técnica de *Pair Programming* deve ser incentivada, de modo que 2 (dois) programadores trabalhem juntos: enquanto um programa, o outro monitora, possibilitando a troca de conhecimento. É de suma importância frisar que, já que é um processo em que a comunicação e o alinhamento são elementares, o código obrigatoriamente precisa ser armazenado em um repositório central de acesso a todos os integrantes.

Concomitantemente, as mesmas regras valem para as Equipes de Arte e de Som. A Equipe de Arte desenvolverá todos os elementos visuais necessários, sejam personagens, cenários, objetos, recompensas, entre outros, devendo compartilhar em um Repositório de Arte. De modo igual, todos as faixas de músicas e efeitos sonoros produzidos pela Equipe de Som durante a *Sprint* devem ser colocados em um Repositório de Som de acesso a todos. Assim, os programadores poderão obter esses recursos e integrar com as linhas de código.

Finalmente, é crucial testar a produção. Mais uma vez, objetivando a agilidade, recomenda-se o uso dos Testes Exploratórios, os quais dispensam a definição demorada de numerosos casos de testes. Nessa abordagem, com o intuito de identificar inconsistências de forma rápida e com a maior cobertura possível, é realizada uma exploração em massa de uma determinada área do *software*. Para isso, é definida uma meta, ou seja, uma área do produto a ser explorada, bem como uma estratégia de exploração e um tempo de execução dos testes, sempre registrando todas as métricas e resultados encontrados, sejam *bugs* ou apenas observações adicionais. No final da execução, portanto, será gerado um relatório com todas as inconsistências encontradas a serem corrigidas.

4.3.4 *Sprint Review*

Após o final do *time box* da Execução da *Sprint*, inicia-se a *Sprint Review*, onde todos os atores estão presentes. Preliminarmente, todas as equipes mostram a produção realizada na *Sprint* para o PO, o qual irá fazer validações. Para cada meta da *Sprint*, ou seja, cada item presente no *Sprint Backlog*, o PO irá validar a produção das equipes para aquele requisito segundo os critérios de aceite, ou seja, se todos os cenários de aceitação estiverem sendo atendidos, a *User Story* é colocada como pronta, caso contrário, continua no *Backlog* e os defeitos encontrados são registrados. Em suma, consiste avaliar se os objetivos da iteração foram concluídos ou não.

Além disso, após visualizar a incremento produzido, o *Product Owner* pode solicitar mudanças no produto, levando a possíveis modificações nos artefatos que descrevem o *game*: o Documento de *Game Design* e o *Product Backlog*. Assim sendo, caso necessário, esses documentos são atualizados.

4.3.5 Retrospectiva

Tal qual a Execução da *Sprint*, na Retrospectiva, o PO é o único ator que está ausente. Enquanto na *Sprint Review* são avaliadas necessidades de mudanças no jogo, na Retrospectiva, são averiguadas necessidades de alterações no processo. O foco é que os

membros das equipes debatam a experiência que presenciaram na última iteração, sob a mediação do *Scrum Master*. Ademais, é fundamental que cada indivíduo tenha sua vez de falar e que pontos positivos e negativos do processo sejam levantados. Para os pontos negativos, soluções devem ser discutidas, e, após acordadas, enfatiza-se que sejam aplicadas na próxima *Sprint*.

Sugere-se que todos os aspectos levantados nessa reunião sejam registrados em um Documento de *Post Mortem* e que tal documentação seja alocada em um repositório de Documentação *Post Mortem*, garantindo que todas as lições aprendidas estejam devidamente disponíveis para consulta para que os erros sejam minimizados e os acertos maximizados em *Sprints* futuras e até mesmo em projetos futuros.

4.3.6 Publicação do Jogo

A última etapa do *Agile Game Development Process* é a de Publicação do Jogo, que ocorre caso a última entrega tenha sido a versão final do *game*, caso contrário, retorna-se para a fase de *Sprint Planning* para planejar uma nova *Sprint*.

Entre as ações tomadas aqui, promove-se o *marketing* e a divulgação do jogo, a partir de anúncios que apresentem os atrativos do produto, de modo a atrair novos consumidores. Também é estabelecido um canal de comunicação com os jogadores, visando sanar eventuais dúvidas que possam surgir com a operação do *game*. Em relação ao canal, o processo recomenda a criação de uma conta de *e-mail* e um corpo de *e-mail* de boas-vindas, enviado automaticamente quando um novo jogador adquire o produto.
Somado a isso, com base na reação do público, são discutidas possibilidades de novos lançamentos: novas expansões, ou seja, mais opções de customização para o jogo, novas versões, isto é, o *game* lançado em outros idiomas ou outras plataformas, ou ainda, uma nova sequência, sendo um jogo novo com a continuação da história atual.

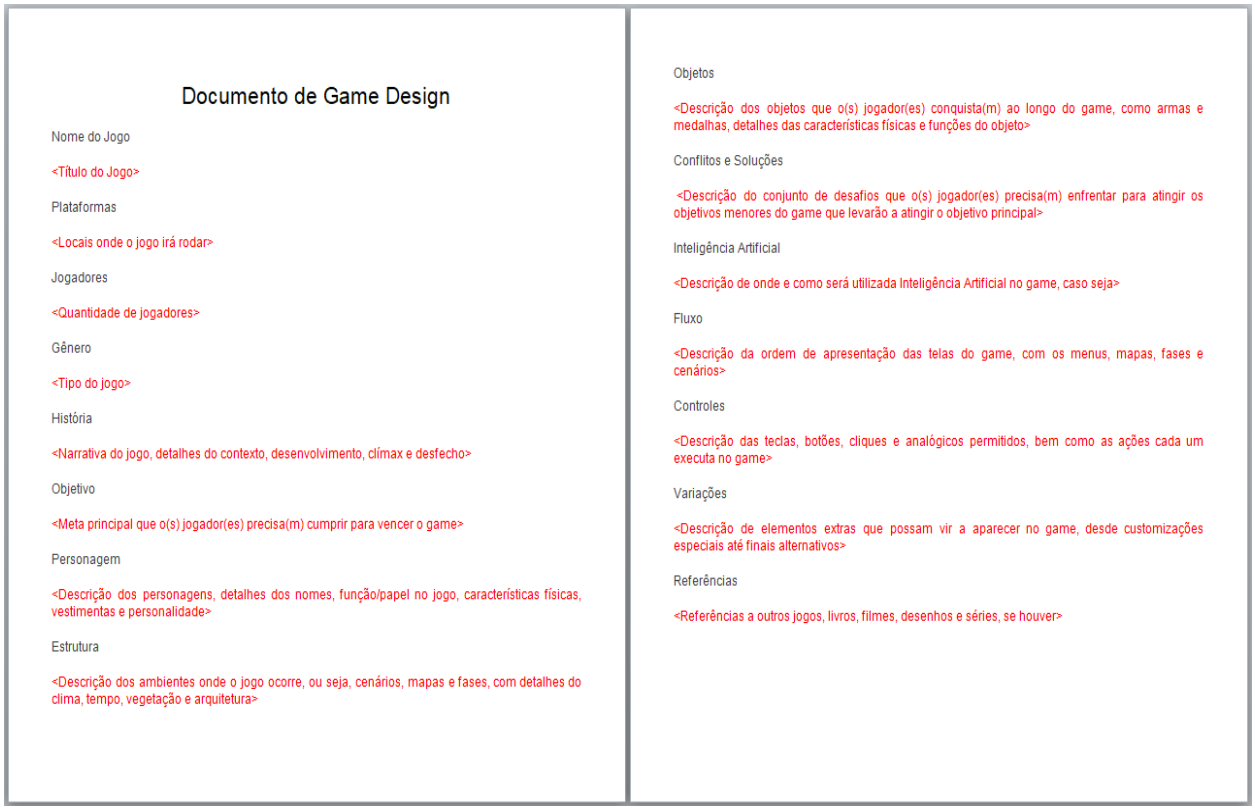
4.4 Artefatos

Nessa seção, serão detalhados os artefatos gerados no *Game Agile Process*. No diagrama BPMN ilustrado na Figura 1, esses elementos estão conectados com as tarefas (formas geométricas azuis) por intermédio das setas pontilhadas. Vale assinalar que nas imagens dos artefatos que serão mostradas nessa seção, foi utilizada a notação de caracteres “<” (menor que) e “>” (maior que) para representar que a sentença localizada entre esses sinais descreve as informações a serem registradas no respectivo campo do artefato.

4.4.1. Documento de *Game Design*

O Documento de *Game Design*, gerado na etapa de Visão Geral do Produto pelo *Product Owner* e pelo *Scrum Master*, possui todas as informações relativas ao projeto do *game*, incluindo desde a temática do jogo até especificidades dos personagens. Constantemente, tal documentação sofre alterações, por isso, é preciso escolher uma ferramenta que disponibilize fácil edição.

Outrossim, não existe uma maneira padrão de escrever esse artefato, desde que cumpra com sua função de servir como um guia elucidativo para que as equipes possam ter uma base sólida para construir o produto. O autor do processo proposto montou um padrão a ser seguido baseado na abordagem de PAUL SCHUYTEMA (2006) na ferramenta *Google Docs* (essa e demais tecnologias serão aprofundadas na seção 4.5), o qual pode ser visualizado na Figura 2.



Documento de Game Design

Nome do Jogo

<Título do Jogo>

Plataformas

<Locais onde o jogo irá rodar>

Jogadores

<Quantidade de jogadores>

Gênero

<Tipo do jogo>

História

<Narrativa do jogo, detalhes do contexto, desenvolvimento, clímax e desfecho>

Objetivo

<Meta principal que o(s) jogador(es) precisa(m) cumprir para vencer o game>

Personagem

<Descrição dos personagens, detalhes dos nomes, função/papel no jogo, características físicas, vestimentas e personalidade>

Estrutura

<Descrição dos ambientes onde o jogo ocorre, ou seja, cenários, mapas e fases, com detalhes do clima, tempo, vegetação e arquitetura>

Objetos

<Descrição dos objetos que o(s) jogador(es) conquista(m) ao longo do game, como armas e medalhas, detalhes das características físicas e funções do objeto>

Conflitos e Soluções

<Descrição do conjunto de desafios que o(s) jogador(es) precisa(m) enfrentar para atingir os objetivos menores do game que levarão a atingir o objetivo principal>

Inteligência Artificial

<Descrição de onde e como será utilizada Inteligência Artificial no game, caso seja>

Fluxo

<Descrição da ordem de apresentação das telas do game, com os menus, mapas, fases e cenários>

Controles

<Descrição das teclas, botões, cliques e analógicos permitidos, bem como as ações cada um executa no game>

Variações

<Descrição de elementos extras que possam vir a aparecer no game, desde customizações especiais até finais alternativos>

Referências

<Referências a outros jogos, livros, filmes, desenhos e séries, se houver>

Figura 2 – Documento de *Game Design*
Fonte: Elaboração própria (2020).

4.4.2. *Product Backlog*

O *Product Backlog* apresenta a lista de funcionalidades a serem desenvolvidas para o *game* e também são escritas na fase de Visão Geral do Produto pelo *Product Owner* em conjunto com o *Scrum Master*, sendo voltadas para a Equipe de Desenvolvimento, para a Equipe de Arte e à Equipe de Som.

Segundo o padrão do *Agile Game Development Process*, cada requisito deve: possuir um título, ser detalhado sob o padrão das *User Stories*, apresentar 1 (um) ou mais Critérios de Aceite sob a sintaxe dos Cenários de Aceitação para a validação, apresentar um nível de prioridade (baixa, média ou alta) de desenvolvimento e ter uma sinalização de que é uma nova história caso se trate de uma mudança sobre o *Backlog* inicial.

Recomenda-se que seja criado um *Domain Expert Kanban* com as colunas "*Product Backlog*", "*Sprint Backlog*" e "Entregue" com cada requisito em seu respectivo status como uma estratégia de acompanhamento do andamento do projeto, ou seja, caso um item ainda esteja pendente, continua na primeira coluna, caso seja alocado para desenvolvimento na *Sprint*, é colocado na segunda coluna e se for validado pelo PO, situa-se na terceira. A Figura 3 ilustra esse quadro elaborado na ferramenta *Trello*.

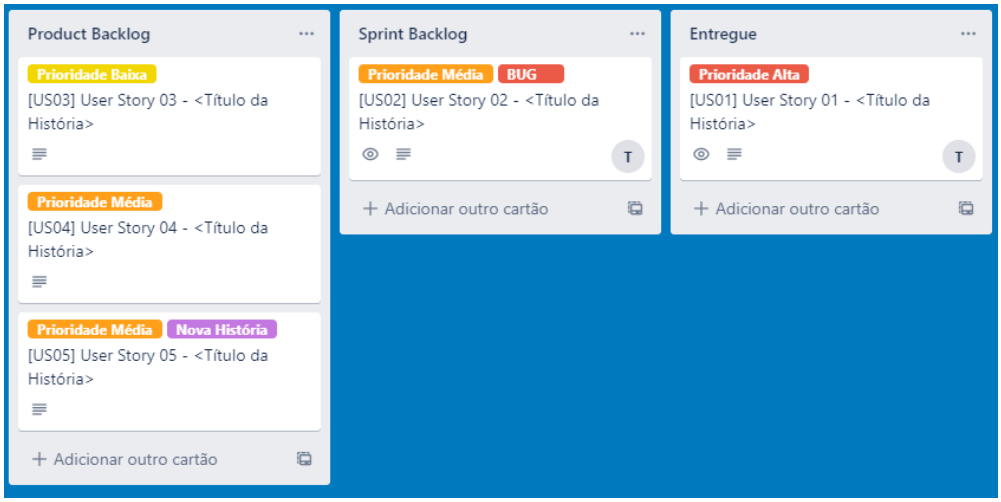


Figura 3 – *Domain Expert Kanban*
Fonte: Elaboração própria (2020).

A Figura 4 ilustra um exemplo de aplicação das sintaxes recomendadas em um dos itens do *Product Backlog* representado na Figura 3.

Figura 4 – Item do *Product Backlog*
Fonte: Elaboração própria (2020).

4.4.3. *Sprint Backlog*

O *Sprint Backlog* é gerado na etapa de *Sprint Planning* e contém a lista de requisitos acordados entre todos os atores do processo a serem desenvolvidos na próxima *Sprint*.

Em relação às funcionalidades, uma vez que vêm do *Product Backlog*, todas as regras recomendadas para cada item continuam válidas, havendo somente 2 (duas) mudanças, pois é preciso: anexar os membros responsáveis por cada requisito e, caso sejam encontradas falhas na execução dos testes internos da Equipe de Desenvolvimento ou o item já tenha sido apresentado para o PO, mas não tenha passado na validação, sinalizar os *bugs*.

No que diz respeito ao quadro, a sugestão é criar um *Team Kanban*, enquanto o *Domain Expert Kanban* mostra o andamento do *game* como um todo, esse quadro refere-se ao acompanhamento das metas da *Sprint*. Para tanto, as colunas "*Sprint Backlog*", "Em Progresso", "Testando" e "Concluído" são designadas, em que cada *User Story* será

alocada de acordo com seu status. Se o item estiver pendente, continua na primeira coluna, caso estiver sendo desenvolvido, situa-se na segunda, se estiver na fase de testes da equipe, aloca-se na terceira e caso já estiver sido finalizado segundo as diretrizes definidas, move-se para a última. A Figura 5 ilustra esse panorama, também montado no *Trello*.

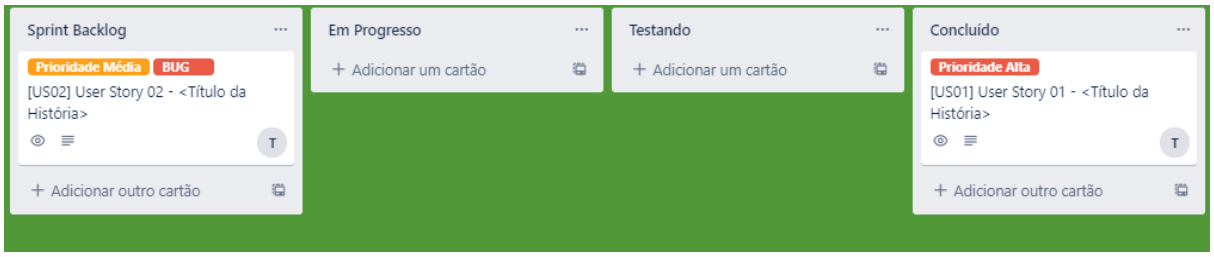


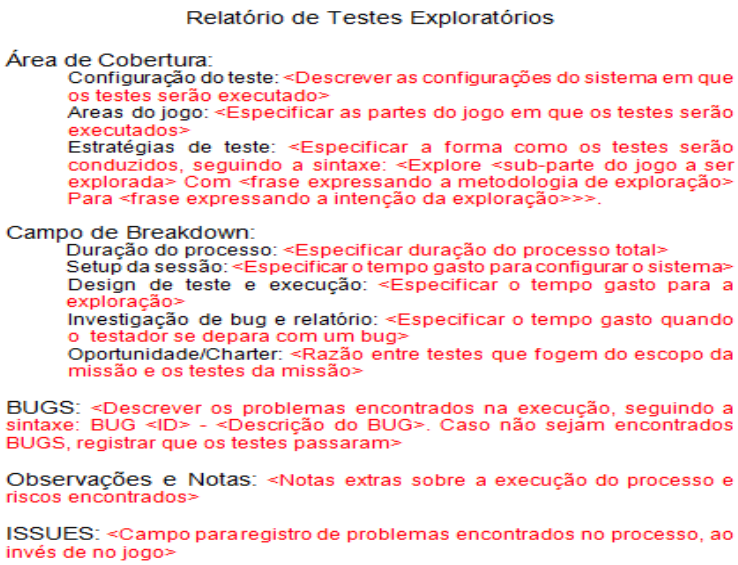
Figura 5 – *Team Kanban*
Fonte: Elaboração própria (2020).

4.4.4 Relatório de Testes Exploratórios

Em sequência, tem-se os Relatórios de Testes Exploratórios, que são elaborados na Execução da *Sprint*, resultantes da execução dos Testes Exploratórios.

Para uma dada área do jogo, são realizadas explorações em massa em um intervalo de tempo fixo e os dados relacionados ao procedimento são registrados minuciosamente no relatório. Os detalhes desse artefato podem ser visualizados na Figura 6, mas, em linhas gerais, são anotadas: configurações da execução dos testes, a área do *game* explorada, a estratégia de exploração, métricas da execução, inconsistências encontradas e notas adicionais.

Aponta-se que é preciso anexar o relatório correspondente em todas as *User Stories* que foram testadas, correspondentes a área do game explorada, de modo encaminhar para correção nos locais onde se localizaram falhas. Em um cenário ideal, tudo é corrigido antes do término da *Sprint*.


Relatório de Testes Exploratórios

Área de Cobertura:
Configuração do teste: <Descrever as configurações do sistema em que os testes serão executado>
Areas do jogo: <Especificar as partes do jogo em que os testes serão executados>
Estratégias de teste: <Especificar a forma como os testes serão conduzidos, seguindo a sintaxe: <Explore <sub-parte do jogo a ser explorada> Com <frase expressando a metodologia de exploração> Para <frase expressando a intenção da exploração>>>.

Campo de Breakdown:
Duração do processo: <Especificar duração do processo total>
Setup da sessão: <Especificar o tempo gasto para configurar o sistema>
Design de teste e execução: <Especificar o tempo gasto para a exploração>
Investigação de bug e relatório: <Especificar o tempo gasto quando o testador se depara com um bug>
Oportunidade/Charter: <Razão entre testes que fogem do escopo da missão e os testes da missão>

BUGS: <Descrever os problemas encontrados na execução, seguindo a sintaxe: BUG <ID> - <Descrição do BUG>. Caso não sejam encontrados BUGS, registrar que os testes passaram>

Observações e Notas: <Notas extras sobre a execução do processo e riscos encontrados>

ISSUES: <Campo para registro de problemas encontrados no processo, ao invés de no jogo>

Figura 6 – Relatório de Testes Exploratórios
Fonte: Elaboração própria (2020).

4.4.5 Documento de *Post Mortem*

O Documento de *Post Mortem* é produzido na reunião de Retrospectiva. Essa documentação tem como objetivo armazenar todos os pontos positivos levantados pelas Equipes sobre o último ciclo do processo, bem como todos os pontos negativos e soluções acordadas para tais entraves. O Agile Game Development Process sugere a criação de um repositório de Documentação Post Mortem para que todas as lições aprendidas ao final de cada iteração estejam disponíveis para consulta não somente em projetos futuros, mas em *Sprints* futuras do projeto atual. Enfim, a Figura 7 mostra o padrão recomendado.

Documento de Post Mortem

Data: <Registrar data de acontecimento da reunião>
Participantes: <Registrar todos os membros da equipe que participaram da reunião>

Pontos Positivos	
<Registrar ponto positivo ocorrido na execução da Sprint anterior>	
<Registrar ponto positivo ocorrido na execução da Sprint anterior>	
<(...)>	
Pontos Negativos	Soluções
<Registrar ponto negativo ocorrido na execução da Sprint anterior>	<Registrar solução para ponto negativo colocado na coluna ao lado> <Registrar solução para ponto negativo colocado na coluna ao lado>
<Registrar ponto negativo ocorrido na execução da Sprint anterior>	<Registrar solução para ponto negativo colocado na coluna ao lado> <Registrar solução para ponto negativo colocado na coluna ao lado>
<(...)>	<(...)>

Figura 7 – Documento de *Post Mortem*
Fonte: Elaboração própria (2020).

4.4.6 Corpo do *e-mail*

Tendo explicada a importância de estabelecer um canal de comunicação com os jogadores na seção 4.3.6, o processo definido apresenta uma sugestão de um corpo de *e-mail* de boas-vindas a ser enviado para os novos jogadores que adquirirem o produto (vide Figura 8).

4.4.7 Outros artefatos

Como é possível observar no fluxo ilustrado na Figura 1, outros artefatos são gerados, como Código-Fonte, Modelo de Banco de Dados, Elementos Visuais, Elementos Sonoros e Incremento Funcional do *game*. No entanto, prezando-se pelo princípio de auto-organização das equipes, no qual os próprios membros são autônomos na decisão da forma de desenvolvimento do jogo, não foram definidos padrões para esses outros artefatos. Cabe

a cada equipe decidir como elaborar o *game* da forma que melhor se adeque ao ambiente de desenvolvimento inserido.

Assunto: Bem vindo a comunidade de <Nome do Jogo>!
Corpo:
Olá, <Nome de Usuário do Jogador>
Seja muito bem vindo a comunidade de <Nome do Jogo>.
Agradecemos a sua aquisição, seu download estará pronto da próxima vez que fizer login em nossa plataforma.
Qualquer dúvida, mande um e-mail para <Email de Contato>.
Atenciosamente,
Equipe de <Nome do Jogo>.

Figura 7 – Corpo de *E-mail* de Boas-Vindas
Fonte: Elaboração própria (2020).

4.5 Ferramentas Recomendadas

O Quadro 1 projeta todas as ferramentas recomendadas para o *Agile Game Development Process*, explicando suas funções e onde podem ser empregadas no processo.

Quadro 1 - Ferramentas Recomendadas

FERRAMENTA	O QUE É A FERRAMENTA E PARA QUE SERVE?	ONDE PODE SER USADA NO PROCESSO?
Google Docs	O *Google Docs* é um *software* que consiste em um pacote de ferramentas de edição de documentos de texto, planilhas, apresentações e formulários, o qual oferece a possibilidade de uso e armazenamento *online*, devido a tecnologia de computação em nuvem ou *offline*, com armazenamento no próprio dispositivo. Consiste em um sistema colaborativo, uma vez que é possível compartilhar um arquivo com outros indivíduos e permitir que também editem, deixem comentários ou somente visualizem.	Para a definição do Documento de *Game Design*, dos Relatórios de Testes Exploratórios e do Documento de *Post Mortem*.
Trello	O *Trello* é uma ferramenta gratuita de quadros virtuais para organização e gestão de tarefas, equipes e projetos, de forma facilitada e dinâmica. É um sistema baseado no método de "Quadros *Kanban*".	Para a definição do *Domain Expert Kanban* e do *Team Kanban*.
Spider Poker	O *Spider Poker* é uma ferramenta online auto-hospedada desenvolvida para estimar esforço de uma atividade a partir da técnica de *Planning Poker*.	Para a execução do *Planning Poker*.
GitHub	O *GitHub* é uma plataforma de repositório de código-fonte, com recursos de controle de versão. Consta em um sistema colaborativo, pois os programadores cadastrados na ferramenta podem contribuir com projetos privados e de código aberto.	Para repositório e controle de versão do código-fonte gerado pela equipe de desenvolvimento.
	O BrModelo é uma ferramenta código aberto, desktop e gratuita voltada para a modelagem de Bancos de Dados	Para a definição do Modelo de Banco de Dados

FERRAMENTA	O QUE É A FERRAMENTA E PARA QUE SERVE?	ONDE PODE SER USADA NO PROCESSO?
BrModelo	Relacionais, apresentando uma interface simples e funcional, a qual disponibiliza todos os elementos do modelo relacional para a criação de esquemas.	Entidade-Relacionamento gerado.
TestLink	O *TestLink* é um *software web* voltado para o acompanhamento de testes de *software*, oferecendo suporte para diferentes abordagens de teste, além de geração de relatórios e métricas.	Para a definição do Relatório de Testes Exploratórios.
Google Drive	O *Google Drive* é um *software* de serviço de armazenamento de arquivos a partir do conceito de computação em nuvem, permitindo a organização dos arquivos em pastas.e o compartilhamento com outros usuários.	Para a definição dos repositórios de Elementos Visuais, de Elementos Sonoros e de Documentos *Post Mortem*.
Gmail	O *Gmail* consta em um serviço gratuito de correio eletrônico. Oferece comunicação autenticada e criptografada, controle de spam, etiquetas para organizações de mensagens, alta capacidade de armazenamento, entre outros recursos.	Para a definição do canal de comunicação com os clientes/jogadores do game.
Audacity	O *Audacity* é um *software desktop* livre para a edição de áudio digital. Permite copiar, recortar, colar e apagar áudio com facilidade, remover ruídos, mixar áudios em múltiplas faixas, entre outros recursos.	Para a criação e edição dos elementos sonoros.
Photoshop	O *Photoshop* é uma das maiores plataformas de edição de imagem e *design gráfico*. Oferece uma quantidade numerosa de recursos de edição, como controle de transparência e de suavidade. Entre as ferramentas propostas para o processo, essa é a que oferece menor aproveitamento da licença gratuita, pois dura somente 7 dias, tendo que pagar por mais tempo de uso.	Para a criação e edição dos elementos visuais.
Unity	O *Unity* é um *software* voltado para o desenvolvimento de jogos. Consiste em um *game engine*, por isso, oferece uma série de recursos que facilitam o desenvolvimento, envolvendo a parte de *scripts*, imagens, animações e sons. Entre suas vantagens, dá suporte a criação de jogos 2D e 3D para múltiplas plataformas e aceita uma grande quantidade de formatos de arquivo. Possui versão gratuita, sendo elegível para pessoas e pequenas empresas que faturaram menos de US$ 100 mil nos 12 meses anteriores.	Para o desenvolvimento do *game* na execução da *Sprint*.

Fonte: Elaboração própria (2020).

5 ADERÊNCIA AO PROCESSO EM RELAÇÃO AOS PROCESSOS DEFINIDOS NA FUNDAMENTAÇÃO TEÓRICA

O Quadro 2 mostra a aderência do *Agile Game Development Process* em comparação aos ativos constantes nos 3 (três) processos definidos na Fundamentação Teórica (Seção 2): *Scrum*, *Extreme Game Development* (XGD) e *Game Waterfall Process*.

Quadro 2 - Aderência do *Agile Game Development Process* em relação ao *Scrum*, XGD e *Game Waterfall Process*

AGILE GAME DEVELOPMENT PROCESS	***SCRUM***	***EXTREME GAME DEVELOPMENT*(XGD)**	***GAME WATERFALL PROCESS***	**JUSTIFICATIVA**
VISÃO GERAL DO PRODUTO	Visão Geral do Produto			Uma vez que a estrutura do ciclo do processo definido é baseada na estrutura do ciclo do *SCRUM*, o nome da etapa inicial do *SCRUM* manteve-se.
		Documento de *Game Design*	Documento de *Game Design*	Tem-se o Documento de *Game Design*, recomendado pelo XGD e definido na etapa de pré-produção do *Game Waterfall Process*, para o detalhamento das características do jogo e guiar o desenvolvimento.
		User Story		A escrita dos requisitos segue as regras das *User Stories*, uma das práticas recomendadas pelo XGD.
	Product Backlog			Todas as *User Stories* são registradas no Product Backlog, artefato do *SCRUM*.
	Priorização de Funcionalidades			De modo similar ao *SCRUM*, cada uma das funcionalidades recebe um grau de prioridade de produção.
	Product Owner			Dado que os atores são baseados no *SCRUM*, tem-se o *Product Owner* - Dono do Produto, responsável por executar todas as tarefas que serão feitas na atividade de Visão Geral do

AGILE GAME DEVELOPMENT PROCESS	***SCRUM***	***EXTREME GAME DEVELOPMENT*(XGD)**	***GAME WATERFALL PROCESS***	**JUSTIFICATIVA**
				Produto em questão.
	Scrum Master			O *SCRUM Master*, outro ator do *SCRUM*, tem o papel de auxiliar o *Product Owner* em todas as tarefas mencionadas acima.
SPRINT PLANNING	*Sprint Planning*			Uma vez que a estrutura do ciclo do processo definido é baseada na estrutura do ciclo do *SCRUM*, o nome desta segunda atividade do *SCRUM* manteve-se.
		Comunicação		A valor de Comunicação do XGD faz-se presente de modo a garantir que todo o time fique coeso e alinhado em relação ao objetivo da *Sprint*.
	Sprint Backlog			Assim como no *SCRUM*, este artefato possui a lista de funcionalidades que serão desenvolvidas na *Sprint*.
	Time SCRUM			Nessa atividade, todos os atores estão presentes: *Product Owner*, *SCRUM Master* e Equipes. De modo similar, o conjunto de todos os atores recebe o nome de Time *SCRUM* no *SCRUM*.
	Execução da *Sprint*			Uma vez que a estrutura do ciclo do processo definido é baseada na estrutura do ciclo do *SCRUM*, o nome desta terceira atividade do *SCRUM* manteve-se.

AGILE GAME DEVELOPMENT PROCESS	***SCRUM***	***EXTREME GAME DEVELOPMENT*(XGD)**	***GAME WATERFALL PROCESS***	**JUSTIFICATIVA**
EXECUÇÃO DA *SPRINT*	Reuniões Diárias	*Stand-Up Meetings*		Tem-se reuniões rápidas diariamente com todas as equipes do time, de forma similar às Reuniões Diárias do *SCRUM* e às *Stand-Up Meetings* do XGD, com o intuito de deixar os participantes alinhados em relação ao andamento do projeto.
	Código-Fonte	Código-Fonte	Código-Fonte	Tem-se a elaboração do Código-Fonte do jogo, artefato comum a todos os 3 (três) modelos de processo relacionados.
		Shared Code		A prática de *Shared Code* - Código Compartilhado, do XGD faz-se presente no processo definido, dado que tem-se a criação de um repositório central para o código fonte de acesso a todo o time.
		Pair Programming		A prática de *Pair Programming* - Programação em Pares, também faz-se presente no processo em questão, já que tem-se a recomendação de que os membros das diferentes equipes trabalhem em pares, permitindo a troca de conhecimento.
		Whole Team		Outra prática do XGD presente é o *Whole Team*. Em todas as Equipes, Desenvolvimento,

AGILE GAME DEVELOPMENT PROCESS	***SCRUM***	***EXTREME GAME DEVELOPMENT*(XGD)**	***GAME WATERFALL PROCESS***	**JUSTIFICATIVA**
				Arte e Som, os membros têm níveis semelhantes de conhecimento, não havendo a ideia de especialização em determinada área.
		Elementos Visuais	Elementos Visuais	Os elementos visuais - imagens e animações - são artefatos do XGD e do *Game Waterfall Process* e são produzidos nessa atividade de Execução da *Sprint* do processo definido.
		Elementos Sonoros	Elementos Sonoros	Os elementos sonoras - faixas de música e efeitos sonoros - são artefatos do XGD e do *Game Waterfall Process* e são produzidos nessa atividade de Execução da *Sprint* do processo definido.
	Testes	Testes	Testes	São executados testes sobre as produções realizadas de modo a garantir qualidade, prática comum aos 3 (três) processos relacionados.
	Equipe de Desenvolvimento	Equipe de Desenvolvimento	Equipe de Desenvolvimento	Tem-se a Equipe de Desenvolvimento representando o ator responsável pela elaboração do código-fonte, ator comum a todos os 3 (três) modelos de processo relacionados.
		Artistas	Equipe de Arte	Os artistas do XGD e a Equipe de Arte do *Game Waterfall Process* representam

AGILE GAME DEVELOPMENT PROCESS	***SCRUM***	***EXTREME GAME DEVELOPMENT*(XGD)**	***GAME WATERFALL PROCESS***	**JUSTIFICATIVA**
				os atores responsáveis pela produção dos elementos visuais, tal qual a Equipe de Arte do processo definido.
		Músicos	Equipe de Som	Os músicos do XGD e a Equipe de Som do *Game Waterfall Process* representam os atores responsáveis pela produção dos elementos sonoros, tal qual a Equipe de Som do processo definido.
	SCRUM Master			O *SCRUM Master*, tal qual no *SCRUM*, consiste no ator responsável por liderar as equipes, potencializando o trabalho de todos os membros.
SPRINT REVIEW	*Sprint Review*			Uma vez que a estrutura do ciclo do processo definido é baseada na estrutura do ciclo do *SCRUM*, o nome desta quarta atividade do *SCRUM* manteve-se.
	Validação	Validação	Validação	Após a Execução da Sprint, é feita a validação das metas do desenvolvimento, atividade comum a todos os 3 (três) modelos de processo relacionados.
	Mudança de Escopo	Mudança de Escopo		Nessa atividade do processo definido, serão avaliadas necessidades de alteração no escopo do jogo. Mudanças no produto também

AGILE GAME DEVELOPMENT PROCESS	***SCRUM***	***EXTREME GAME DEVELOPMENT*(XGD)**	***GAME WATERFALL PROCESS***	**JUSTIFICATIVA**
				são gerenciadas no *SCRUM* e no XGD, uma vez que são métodos ágeis.
	Entrega Funcional	Entrega Funcional	Entrega Funcional	Tem-se a Entrega Funcional no final de um *Time Box* de desenvolvimento. O artefato é comum a todos os 3 (três) modelos de processo relacionados.
	Time SCRUM			Todos os atores do processo participam dos procedimentos de validação do desenvolvimento e de avaliação de mudanças. Novamente, de forma similar ao *Time Scrum*.
RETROSPECTIVA	Retrospectiva			Uma vez que a estrutura do ciclo do processo definido é baseada na estrutura do ciclo do *SCRUM*, o nome desta quinta atividade do *SCRUM* manteve-se.
	Registro da Reunião de Retrospectiva	Documento de *Post Mortem*	Documento de *Post Mortem*	Tem-se o registro dos pontos positivos, negativos e planos de ação no Documento de *Post Mortem*, artefato em comum com o XGD e *Game Waterfall Process*. No SCRUM, não há um artefato definido mas há o registro dessas decisões.
	Equipes	Equipes	Equipes	Todas os membros das Equipes participam da atividade de avaliação do processo. As Equipes consistem em um

AGILE GAME DEVELOPMENT PROCESS	***SCRUM***	***EXTREME GAME DEVELOPMENT*(XGD)**	***GAME WATERFALL PROCESS***	**JUSTIFICATIVA**
				ator em comum a todos os 3 (três) processos relacionados.
PUBLICAÇÃO DO JOGO			Pós-Lançamento	A atividade final do processo definido está baseada na atividade de Pós-Lançamento do *Game Waterfall Process*.
	Produto Final	Versão Final do Jogo/*Game*	Jogo / *Game*	Tem-se a publicação da versão final do Jogo, correspondendo ao Produto de *Software* Final do *SCRUM*, à Versão Final do Jogo/*Game* no XGD e ao Jogo/*Game* no *Game Waterfall Process*.
			Marketing	Um plano de Marketing do jogo é estabelecido para promover a divulgação do game, tal como no *Game Waterfall Process*.
			Canal de Comunicação	Um Canal de Comunicação é estabelecido com os jogadores do *Game* para sanar eventuais dúvidas, tal qual é feito no *Game Waterfall Process*.
			Análise de Viabilidade de Continuações	Assim como no *Game Waterfall Process*, são discutidas possibilidades de prosseguir o lançamento com mais versões, expansões e sequências.

Fonte: Elaboração própria (2020).

6 CONCLUSÕES

A partir das revisões na literatura, foi possível identificar algumas abordagens para o Desenvolvimento de Jogos Digitais, possibilitando a criação de um modelo de processo baseado nestas.

Este trabalho une o Scrum tradicional, que não inclui elementos do contexto dos *games*, como o *Game Design*; ao XGD, que não prescreve uma estrutura de iteração a ser seguida, somente um conjunto de valores e práticas juntamente à interação massiva com os clientes; e ao Game Waterfall Process, o qual sugere excessivas documentações e é uma abordagem cascata, não sendo recomendada para a maioria dos casos na economia moderna. Portanto, foram analisados e combinados os elementos desses processos mais recomendados ao Desenvolvimento Ágil de Games com outras práticas ágeis para a montagem do processo proposto.

Tem-se detalhamentos do fluxo, dos atores, das tarefas, dos artefatos, além de tecnologias que podem dar suporte ao Agile Game Development Process. Nessa conjuntura, este trabalho funciona como um guia para a implementação do processo, fornecendo suporte a qualquer Organização de Desenvolvimento de Games que opte por implementar.

Pontua-se que o processo em questão não foi testado em ambientes reais de desenvolvimento, devendo-se levar em consideração fazer implementações e realizar trabalhos que analisem os resultados, e, a partir disso, averiguar melhorias para o processo, caso necessário.

Também é possível fazer aplicações para desenvolvimento de diferentes gêneros de Jogos Digitais e analisar se o processo se adapta melhor a alguns tipos de *games* em relação a outros.

REFERÊNCIAS BIBLIOGRÁFICAS

GODOY, A. BARBOSA, E. Game-Scrum: An Approach to Agile Game Development. SBGames 2010.

KEITH, C. Agile Game Development with Scrum. 2010.

SALEN, K. ZIMMERMAN, E. Rules of Play: Game Design Fundamentals, 2004.

ROGERS, S. Level Up: Um Guia para o Design de Grandes Jogos. 1a edição. 2013.

AGILE ALLIANCE. Retrieved from https://www.agilealliance.org/.

PRESSMAN, Roger. Engenharia de Software: Uma Abordagem Profissional. 7a edição. Porto Alegre: MCGraw - Hill, 2011.

DEMACHY, T. Extreme Game Development: Right on Time, Every Time. Retrieved from https://www.gamasutra.com/view/feature/131236/extreme_game_development_right_on_.php.

BARROS, O. Análise de Metodologias de Desenvolvimento de Software aplicadas ao Desenvolvimento de Jogos Eletrônicos. Trabalho de Graduação - CIN/UFPE. 2007.

SLOPER, T. How the Game Development/Production Process Works. Retrieved from https://www.sloperama.com/advice/lesson10.htm

SHWABER, K. SUTHERLAND, J. The Definitive Guide to Scrum: The Rules of the Game. Retrieved from https://www.scrumguides.org/

MEDEIROS, M. Extreme Programming – Conceitos e Práticas. Retrieved from https://www.devmedia.com.br/extreme-programming-conceitos-e-praticas/1498

CARVALHO, E. Os 5 Desprezíveis: Desenvolvimento de um Jogo Eletrônico Utilizando os Princípios de Engenharia de Software. Universidade de Brasília: Brasília, 2013.

OLIVEIRA, F. Conhecendo a Área de Desenvolvimento de Games. Retrieved from https://www.fabricadejogos.net/posts/artigo-conhecendo-a-area-de-desenvolvimento-de-games/

SCHUYTEMA, P. Game Design: A Pratical Approach. 2006.