UNIVERSIDADE FEDERAL DO CEARÁ

INSTITUTO UFC VIRTUAL

CURSO DE GRADUAÇÃO EM SISTEMAS E MÍDIAS DIGITAIS

HENRIQUE ARTUR CORDEIRO GOMES

ANÁLISE DE METODOLOGIAS ÁGEIS COMO RECURSOS PARA DESENVOLVIMENTO DE JOGOS DIGITAIS DE PEQUENO ESCOPO

FORTALEZA

2021

HENRIQUE ARTUR CORDEIRO GOMES

ANÁLISE DE METODOLOGIAS ÁGEIS COMO RECURSOS PARA DESENVOLVIMENTO DE JOGOS DIGITAIS DE PEQUENO ESCOPO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas e Mídias Digitais do Instituto UFC Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas e Mídias Digitais.

Orientador: Prof. Dr. Henrique Barbosa Silva

FORTALEZA

2021

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

G614a Gomes, Henrique Artur Cordeiro.
Análise de metodologias ágeis como recursos para desenvolvimento de jogos digitais de pequeno escopo / Henrique Artur Cordeiro Gomes. – 2021.
63 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Instituto UFC Virtual, Curso de Sistemas e Mídias Digitais, Fortaleza, 2021.
Orientação: Prof. Dr. Henrique Barbosa Silva.

1. Metodologias ágeis. 2. Jogos. 3. Produção. 4. Scrum. 5. Feature-Driven Development. I. Título.
CDD 302.23

HENRIQUE ARTUR CORDEIRO GOMES

ANÁLISE DE METODOLOGIAS ÁGEIS COMO RECURSOS PARA DESENVOLVIMENTO DE JOGOS DIGITAIS DE PEQUENO ESCOPO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Sistemas e Mídias Digitais do Instituto UFC Virtual da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Sistemas e Mídias Digitais.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Henrique Barbosa Silva (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. George Allan Menezes Gomes
Universidade Federal do Ceará (UFC)

Prof. Dr. Abraão Freires Saraiva Júnior
Universidade Federal do Ceará (UFC)

Às minhas irmãs Maria Cecília e Sofia e a todos os meus primos, para que a educação, o amor e perseverança pelo seus objetivos sejam sempre as respostas para os momentos mais difíceis e incertos da nossa vida.

AGRADECIMENTOS

Ao Prof. Dr. Henrique Barbosa Silva, por ter tanta paciência com meus extensos discursos mirabolantes e por me orientar em meu Trabalho de Conclusão de Curso (TCC).

Ao Prof. Dr. George Allan Menezes Gomes e ao Prof. Dr. Abraão Freires Saraiva Júnior, por aceitarem participar da banca avaliadora deste TCC.

Ao Prof. Dr. Gabriel Antoine Louis Paillard e ao Prof. Dr. Alysson Diniz dos Santos, por todo apoio e pela orientação na fundação e formalização da Empresa Jr. TGD Studio.

À Profa. Ma. Mara Franklin Bonate e à Profa. Dra. Ticiana Linhares Coelho da Silva, por serem minhas referências na área de Sistemas.

A todos os professores com os quais estudei, por me proporcionarem o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender.

Aos secretários Allan George Bezerra e Monalisa Conceição Batista de Menezes, pelo companheirismo e por sempre se fazerem presentes em auxiliar em todos problemas referentes ao setor administrativo da Universidade Federal do Ceará (UFC) e de SMD.

À aluna de graduação em SMD e amiga Gabriely Rodrigues de Lima (Gabi), por ter me acompanhado e apoiado durante toda a nossa graduação, sempre resiliente!

A todos os membros da TGD Studio, principalmente aos alunos Álvaro Carvalho Monteiro, Bárbara de Sousa Ramos, Daniel Portela Bandeira, Júlio Cesar de Castro Ribeiro e Raoni Coelho Vieira, por acreditarem no nosso projeto de fazermos parte do plano de crescimento do mercado de jogos no Ceará e por serem uma fonte de inspiração.

Aos membros da Contêiner Digital Jr., por todo acolhimento e ensinamentos, principalmente à aluna Rafaelly Freitas Ferreira (Rafinha), por todo apoio.

Aos meus pais, irmãs, avós, demais familiares e amigos, que, nos momentos de minha ausência dedicados ao estudo superior, sempre fizeram entender que o futuro é feito a partir da constante dedicação no presente!

Ao Doutorando em Engenharia Elétrica Ednardo Moreira Rodrigues e seu assistente Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da UFC.

"I’ll never break
Under their madness
I’ll be miles above
My head up to the sky
I’ll never break
Under their pressure
I’ll keep it grow
My rhythm leadin’ on"

(Ace)

RESUMO

A produção de um jogo digital é uma atividade que necessita de diversas áreas de conhecimento. Para isso, há a necessidade de uma equipe multidisciplinar. Entretanto, esse desenvolvimento pode acabar sendo extremamente conturbado, caso não haja planejamento e acompanhamento das atividades, mesmo que em projetos de pequeno escopo. Daí, surge a necessidade da utilização de ferramentas ou métodos para que haja uma melhor comunicação entre essas equipes. Partindo dessa ideia, este trabalho tem como objetivo analisar as características e a aplicabilidade de 5 (cinco) metodologias ágeis para o desenvolvimento de jogos de pequeno escopo, incluindo o *framework* Scrum e as metodologias eXtreme Programming (XP), Kanban, Feature-Driven Development (FDD) e Rational Unified Process (RUP), por terem sido identificadas como as metodologias mais abordadas em estudos correlatos. Os dados foram analisados a partir da aplicabilidade teórica das metodologias ágeis no ciclo básico de produção de jogos e no ciclo de produção de jogos por área. Como resultado da pesquisa, tem-se a indicação das metodologias Kanban e Feature-Driven Developmente (FDD) como aquelas que mais se alinham às necessidades percebidas no desenvolvimento de um jogo de pequeno escopo, sem descartar a possibilidade de desenvolvimento de uma metodologia completamente nova voltada especificamente para o mercado de jogos digitais de pequeno escopo.

Palavras-chave: Metodologias ágeis. Jogos. Produção. Scrum. eXtreme Programming. Kanban. Feature-Driven Development. Rational Unified Process.

ABSTRACT

The production of a digital game is an activity that requires different areas of knowledge. Because of that, there is a need for a multidisciplinary team. However, this development can end up being extremely troubled, if there is no planning and monitoring of activities, even in projects with a small scope. Hence, there is a need to use tools or methods to have better communication between these teams. Based on this idea, this work aims to analyze the use of 5 (five) agile methodologies for the development of small scope games, including the Scrum framework and the eXtreme Programming (XP), Kanban, Feature-Driven Development (FDD) methodologies and Rational Unified Process (RUP), for having been identified as the most discussed methodologies in related studies. The data were analyzed based on the theoretical applicability of the methodologies agile in the basic game production cycle and in the game production cycle by area. As a result of the research, the Kanban and Feature-Driven Development (FDD) methodologies are indicated as those that are most in line with the perceived needs in the development of a small-scope game, without discarding the possibility of developing a completely new methodology aimed at specifically for the small scope digital game market.

Keywords: Agile methodologies. Games. Production. Scrum. eXtreme Programming. Kanban. Feature-Driven Development. Rational Unified Process.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AAA	*Triple A*
AGVMM	*Agile Game Vision Management Method*
DLC	*Downloadable Content*
FDD	*Feature-Driven Development*
GDD	*Game Design Document*
GUP	*Game Unified Process*
JIT	*Just-in-Time*
POC	*Proof of Concept*
RUP	*Rational Unified Process*
SMD	Sistemas e Mídias Digitais
TGD Studio	The Guardian Dog Studio
XP	*eXtreme Programming*

SUMÁRIO

1 INTRODUÇÃO

A produção de jogos digitais desde a sua concepção envolve conhecimento em diversas áreas, como *game design*, *design*, *concepts*, programação, músicas, entre outras, que variam dependendo das especificidades de cada projeto. Os profissionais de cada uma dessas áreas possuem seus próprios jargões, tempos distintos de desenvolvimento de seus trabalhos e, para que essas áreas se entendam e que os objetivos do projeto sejam guiados e realizados com sucesso, deve-se haver uma figura para ligar todo esse conhecimento. Para Chandler (2012, p. 1), "O produtor é a principal força condutora que guia o processo de desenvolvimento de jogos para assegurar que o trabalho seja concluído a tempo e conforme o orçamento."

A figura de um produtor de jogos digitais é recorrente em grandes projetos de desenvolvimento de jogos, porém essa função, no cenário de desenvolvimento *indie*, por vezes, é pouco explorada ou, em alguns casos, negligenciada.

> "Mesmo sendo tão importante para o sucesso de um projeto, com frequência, o produtor não recebe treinamento algum para desempenhar a função. Geralmente, ele assume um papel difícil, sem um treinamento formal de gerenciamento. É promovido para o cargo e espera-se que aprenda por osmose, fazendo, ou que simplesmente saiba o que fazer de modo intuitivo." (CHANDLER, 2012, p. xiii)

Mesmo que em pequenos escopos, o planejamento do desenvolvimento de um projeto é de extrema importância para o entendimento do mesmo, para o gerenciamento dos riscos e para o sucesso, como citado por Chandler (2012).

As atividades diárias de um produtor não diferem muito das atividades de um administrador de projetos, que pode buscar soluções para a gerência dos projetos em ferramentas e técnicas que já estão consolidadas dentro dessas áreas. Destaque para as ferramentas fundamentadas em Metodologias Ágeis, que são empregadas em diversos projetos de teor audiovisual, como jogo ou de outras áreas.

Durante o período como produtor *indie* da empresa jr. TGD Studio e pela experiência em desenvolvimento de jogos digitais em disciplinas cursadas no curso de Sistemas Mídias Digitais - UFC, foi notada a necessidade pelas equipes de encontrar ferramentas que auxiliassem as entregas de cada etapa da produção e que possibilitassem mudanças durante o desenvolvimento do produto, já que jogos são produtos extremamente mutáveis durante todo o seu processo de criação, e as metodologias ágeis, com pilares do manifesto para desenvolvimento ágil de *software*, foram as que melhor se adequaram às situações apresentadas nos projetos.

Com o avanço das tecnologias e o aumento da complexidade dos projetos ao longo dos anos, surgiu a necessidade de se criar soluções inovadoras para o gerenciamento desses projetos.

Mais recentemente, no final do séc. XX, houve um movimento da comunidade de desenvolvimento de *softwares* para tentar solucionar os principais problemas de planejamento e gestão desses tipos de projeto, o que culminou na construção do Manifesto Ágil (BECK *et al.*, 2001).

Diversas metodologias e *frameworks* ágeis foram criados seguindo os pilares propostos por Beck *et al.* (2001), contemplando a gestão de projetos de desenvolvimento de *softwares*, incluindo os jogos digitais.

Com base nessas premissas, este trabalho tem como motivação fazer uma análise das metodologias ágeis para produtores que estejam inseridos no contexto de produção de jogos digitais de pequeno escopo.

1.1 Objetivos

1.1.1 Objetivo Principal

Analisar metodologias ágeis que se aplicam à produção de jogos digitais de pequenos escopos no contexto de estúdios de desenvolvimento *indie*, auxiliando o processo de gestão dos projetos por parte dos produtores.

1.1.2 Objetivos Específicos

a) Compreender as dificuldades de estúdios *indies* relacionadas à gestão;
b) Levantar metodologias/técnicas de gestão aplicadas a jogos e metodologias ágeis;
c) Correlacionar as metodologias ágeis de gestão de projeto;
d) Analisar as metodologias ágeis;
e) Identificar quais metodologias se aplicam a estúdios *indies* que trabalham com escopos pequenos;
f) Sugerir aplicações das metodologias escolhidas.

Ao logo do trabalho, o objetivo "a)"será alcançado a partir das teorias e ideias dos autores apresentados no referencial teórico e das próprias observações feitas pelo autor deste trabalho. O objetivo "b)"será alcançado com o desenvolvimento do Capítulo 4, de metodologias.

Já os objetivo "c)", "d)", "e" e "f)"serão trabalhados no Capítulo 5, Análises e Discussão dos Resultados.

1.2 Divisão do trabalho

O presente trabalho esta dividido em 6 Capítulos. No Capítulo 1, Introdução, encontra-se uma visão mais generalizada de como será abordado o trabalho, bem como as explicações sobre suas divisões. Já o Capítulo 2, Referencial Teórico, e 3, Referencial Teórico das Metodologias Ágeis, serão responsáveis por apresentar autores, ideias e conceitos utilizados ao longo do texto. No Capítulo 4, Metodologia, encontram-se as metodologias utilizadas no desenvolvimento do trabalho. Já o Capítulo 5, Análises e Discussão dos Resultados, será utilizado para apresentar as observações feitas acerca de cada uma das Metodologias Ágeis trabalhadas ao longo do texto. Por fim, o Capítulo 6, Conclusão e Trabalhos Futuros,trará as conclusões, as perspectivas de continuidade do trabalho e especificará as limitações encontradas no desenvolvimento do mesmo.

No Capítulo a seguir, será apresentada a primeira parte dos referencias teóricos que permitiram fundamentar as ideias e os conceitos que este trabalho utilizou como base para a pesquisa.

2 REFERENCIAL TEÓRICO

Neste capítulo, é apresentada a fundamentação teórica deste trabalho. No inicio, na seção 2.1, encontra-se um resumo de todos os autores aqui referenciados; na seção 2.2, é dada a definição de jogo adotada pela pesquisa; a seção 2.3 se atém à definição de produção em jogos; em seguida, na seção 2.4, é abordada a definição de gerenciamento de projetos; logo após, na seção 2.5, são apresentados conceitos de escopo; já na seção 2.6, são listadas as referências para o cenário *indie*; por fim, na seção 2.7, são apresentados os trabalhos correlatos ao tema da pesquisa realizada.

O referencial teórico foi dividido para uma melhor organização do trabalho. Assim sendo, as referências sobre as metodologias ágeis se encontram no Capítulo seguinte.

Para a escolha dos autores e metodologias discutidas ao longo do referencial teórico, foram utilizados diferentes critérios, de modo a contemplar a fundamentação necessária para cada uma das seções deste Capítulo. Na seção de Jogos Digitais foram selecionados autores que compartilham de visão similar ao autor do texto. Já para as escolhas de metodologias ágeis, utilizou-se o critério de recorrência em trabalhos correlatos. Por fim, para a fundamentação da produção e de cada metodologia individual, utilizou-se como critério a escolha de autores de referência na sua área ou autores responsáveis pela criação da metodologia em si.

2.1 Quadro de referências

Dada a natureza inerente de pesquisa para reunir diversos materiais publicados, este trabalho utilizou-se de agregadores de conhecimento, como Google Scholar, ERIC, Periódico da CAPES, anais de eventos da área de desenvolvimento de jogos e sites, blogs e mídias especializadas em produção de jogos. Ao final da pesquisa bibliográfica, fez-se necessária a construção de um quadro guia dos autores agrupados por tema referenciados no texto, para uma melhor organização. O quadro gerado encontra-se a seguir:

Quadro 1 – Quadro de autores referenciados

ÁREA	AUTORES
Jogos digitais	Huizinga (1938) Salen e Zimmerman (2003)
Produção	Chandler (2012) Mencher (2006) Nakano *et al.* (2012)
Gerenciamento de projetos	Amaral *et al.* (2011) Santos (2014)
Escopo	Kruchten (2004) Possi *et al.* (2006)
Cenário *indie*	Keogh (2015) Vieira (2015)
Trabalhos correlatos	Godoy e Barbosa (2010) Laubisch e Clua (2010) Posvolski *et al.* (2014) Sales (2013)
Metodologias	Dias e Silva (2009) Gil (2010) Prodanov e Freitas (2013)
Metodologias Ágeis	Beck (1999) Beck *et al.* (2001) Boeg (2010) Chandler (2020) Faccioli e Silva (2020) Ferreira e Lima (2006) Guimarães e Falsarella (2008) Highsmith (2002) Rational Software Corporation (2001) Schwaber e Sutherland (2013) Schwaber e Sutherland (2020) Soares (2004) Silva e Anastácio (2019) Tamaki (2007)

Fonte: O Autor.

2.2 Jogos Digitais

Para Huizinga (1938, p. 28), jogo

> "É uma atividade ou ocupação voluntária exercida dentro de certos e determinados limites de tempo e espaço, segundo regras livremente consentidas, mas absolutamente obrigatórias, dotado de um fim em si mesmo, acompanhado de um sentimento de tensão e alegria e de uma consciência de ser diferente da vida cotidiana".

Numa visão mais moderna, que complementa a definição de Huizinga, temos que "Um jogo é um sistema no qual os jogadores engajam-se em um conflito artificial definido por regras, que resultam em um resultado quantificável." (SALEN; ZIMMERMAN, 2003, p. 96)

A união das duas citações mencionadas nesta subseção foi adotada por compreender a maior diversidade de entendimento dos vários tipos de jogos digitais e permitir a correlação

com outros conceitos apresentados na seção.

2.3 Produção

Para Chandler (2012), o trabalho de um produtor, logo daquele que cuida da produção de um jogo, envolve diversas atividades, como gerenciar cronogramas, calcular orçamentos, compartilhar a visão do design, compreender jargões e termos das diversas equipes de desenvolvimento (arte, programação, áudio, marketing etc.), ter conhecimentos jurídicos e possuir eloquência em sua oratória e escrita.

Já na visão de Mencher (2006), o produtor deve ter as habilidades de organização, programação de atividades, capacidade de construção de consenso entre os membros da equipe e direcionamento do desenvolvimento cumprindo prazos.

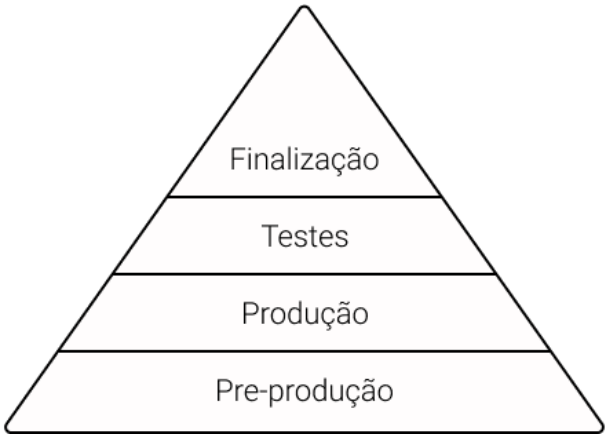
"[...] as atribuições dos produtores são variadas, mas se concentram no gerenciamento do projeto, incluindo orçamentos, cronograma e pessoal. Produtores também realizam a interface entre desenvolvedores e publicadores." (NAKANO *et al.*, 2012, p. 8)

Chandler (2012) cita em seu trabalho dois ciclos de produção de jogos:

a) Ciclo básico de produção de jogos
   - Pré-produção
   - Produção
   - Testes
   - Finalização
b) Ciclos de produção por área
   - Design
   - Artístico
   - Programação
   - Trabalho em conjunto

O ciclo básico de produção divide a produção em 4 grandes etapas, como mostrado na figura 1.

Figura 1 – Ciclo básico de produção de jogos.



Fonte: Chandler (2012).

A fase de Pré-produção, de acordo com Chandler (2012), dura entre 10 e 25% do tempo total de desenvolvimento, porém, pelas experiências trabalhando em projeto de pequeno escopo na empresa jr. e projetos da UFC, esse tempo, em média, fica entre 20 e 30%.

A autora usa como referências produções de jogos *Triple A* (AAA), grande orçamento e funcionários. No contexto *indie*, os orçamentos são muito pequenos comparados às grandes produções, bem como à quantidade de desenvolvedores.

Nessa fase, de acordo com a autora, o jogo é conceituado, os requisitos necessários são elencados, o planejamento é elaborado e os riscos são avaliados. No caso de *indie*, por experiência prévia, e pequeno escopo, a elaboração de estratégias de marketing deve ser definida nessa fase, pois o desenvolvimento pode durar de 1 a 12 meses. Portanto, quanto mais cedo for tomada essa decisão, menores serão os impactos negativos da mesma.

Na fase de Produção, são desenvolvidos os *Assets* (elementos gráficos) e o código do jogo. Nesta fase, deve ser implementado o plano desenvolvido na pré-produção, o rastreamento do progresso e a conclusão de tarefas. A avaliação de riscos é constante durante esta etapa. Porém, em se tratando de jogos de pequeno escopo, esta fase não necessita de tanta atenção como em projetos grandes nos quais a capacidade de ficar reavaliando é muito baixa por conta da quantidade de atividades para uma equipe *indie*.

A fase de Teste, apesar de aparecer acima da fase de Produção na figura 1, ocorre durante todas as outras fases. Um jogo deve estar em constante teste, não só para a validação do código, mas também para saber se o *design* está bem elaborado, se condiz com conceito estabelecido na fase de Pré-produção, bem como para realizar ajustes em balanceamento do jogo, caso necessário.

E, por fim, temos a fase de Pós-produção, que consiste na avaliação e documentação das experiências do desenvolvimento do jogo e arquivamento do código e *assets*. Porém, nos dias atuais, essa fase também pode ser usada para o desenvolvimento de conteúdos extras do jogo, chamados de *Downloadable Content* (DLC).

Já em se tratando das principais áreas no desenvolvimento de jogos, temos as especificidades de cada uma das equipes durante o desenvolvimento de um jogo as quais são explicitadas no quadro 2.

Quadro 2 – Ciclos de produção por área

CICLO	DESCRIÇÃO
Design	Implementação da jogabilidade, ajustar recursos de jogabilidade, emitir *feedbacks* para outras equipes, desenvolver diálogos e *voiceover* para o jogo, evoluir as *features* e conduzir *playtests*.
Artístico	Criação dos diversos *assets*, aperfeiçoar *assets* já desenvolvidos e comunicação constante com programadores para refino da *pipeline*.
Programação	Codificar, depurar recursos do jogo, identificar *crashes* e *bugs*, criar *builds* e fazer a manutenção da *pipeline* de produção.
Trabalho em conjunto	Comunicação e interação constante entre todas as equipes, emitir *feedbacks* construtivos e verificação de atividades de outros membros da equipe.

Fonte: Chandler (2012).

2.4 Gerenciamento de Projetos

O gerenciamento de projeto é parte fundamental do trabalho do produtor de jogos, sendo considerado uma atividade de extrema importância dentro do contexto do desenvolvimento, que necessita de monitoramento e controle constantes, dada sua característica de ser extremamente mutável, desde sua concepção até a entrega do produto final.

"Conforme o Guia PMBOK, o gerenciamento de projetos é a aplicação de conhecimento, habilidades, ferramentas e técnicas às atividades do projeto, com o objetivo de atender a seus requisitos." (SANTOS, 2014, p. 12)

"O Gerenciamento Ágil seria então uma maneira de proceder baseada em um conjunto de elementos (princípios, técnicas, etc.) em que essa atividade é conduzida por meio de equipes autogeridas e utilizando técnicas de gerenciamento simplificadas." (AMARAL *et al.*, 2011, p. 11)

2.5 Escopo

Quando se trata de gerenciamento de projetos, existem 3 pilares que são base para essa área, quais sejam: escopo, tempo e custo. Tais pilares são de extrema importância e são utilizados juntos, porém o presente trabalho dará foco apenas no escopo, pois este é o que caracteriza melhor um jogo *indie*.

Uma parte fundamental do desenvolvimento de jogos é o escopo. Muitos jogos sofrem problemas de escopo, pois são mal dimensionados para a capacidade técnica e até mesmo de recursos humanos. Não existe uma métrica para estabelecer o que é um projeto de pequeno, médio ou grande escopo, pois existem vários fatores envolvidos. Cada equipe possui características diversas e diferentes. Além disso, cada projeto apresenta suas próprias especificidades. Neste trabalho, consideraremos pequeno escopo como jogos de, no máximo, 1 ano de desenvolvimento por equipes de, no máximo, 5 pessoas.

> "Escopo do Projeto - o trabalho que precisa ser realizado para entregar o produto, serviço ou resultado com as características e funções especificadas. Escopo do Produto - as características e funções que descrevem um produto, serviço ou resultado a ser gerado pelo projeto." (POSSI *et al.*, 2006, p. 96)

No desenvolvimento de *software*, o escopo pode ser definido como "Capturar o contexto, os requisitos e restrições mais importantes para assim, definir os critérios de aceitação do produto final." (KRUCHTEN, 2004, p. 67)

2.6 Cenário *Indie*

"A cena *indie* contemporânea - onde indivíduos ou pequenas equipes podem ter esperança de obter lucros - surge ao longo dos anos 2000 em resposta a essa indústria baseada em produtos cada vez mais homogeneizada e corporalizada." (KEOGH, 2015, p. 155)

> "Entende-se como 'independentes' estúdios que apresentam um ambiente de trabalho livre, sem obrigações contratuais que possam impedir a liberdade criativa, salvo em casos de concursos com temáticas pré-estabelecidas, que ainda permitem liberdade de criação e inovação." (VIEIRA, 2015, p. 38)

O Cenário *indie*, a partir do ano de 2012, começa a ganhar mais espaço com o barateamento das ferramentas de desenvolvimento, a disseminação de conhecimento e os primeiros jogos *indies*, ganhando, assim, visibilidade. São jogos normalmente feitos por pequenas equipes, sem grandes orçamentos e com muita paixão. Alguns autores, como Vieira (2015), colocam

"inovação" como uma característica de um jogo *indie*. Porém, essa afirmação é questionável, pois, hoje, muitos dos jogos *indies* não possuem inovações. Em vez de inovar, os jogos refinam *designs* já estabelecidos anteriormente ou, até mesmo, usam algo que já está estabelecido no mercado de uma forma ligeiramente diferente.

2.7 Trabalhos Correlatos

Para a pesquisa dos trabalhos correlatos, foram consultados, principalmente, agregadores de trabalhos acadêmico na área de jogos, como o site do SBGames, e motores de busca, como o Google e Google Scholar. Para a seleção dos trabalhos, aplicou-se o critério da utilização ou citação de metodologias ágeis no contexto de desenvolvimento de jogos.

A seguir, serão apresentadas citações de autores que, em sues trabalhos, relatam quais as dificuldades do desenvolvimento de jogos e quais ferramentas ou técnicas utilizaram para a resolução dos problemas. Também serão listadas as experiências adquiridas com a utilização das metodologias ágeis.

Godoy e Barbosa (2010) fazem um apanhado das dificuldades no processo de desenvolvimento de jogos. O trabalho dos autores se concentra em especificar as Metodologias Scrum e *eXtreme Programming* (XP) aplicadas ao contexto de jogos.

> "Por ser uma metodologia focada na simplicidade, agilidade e redução da burocracia, o Scrum se foca na quantidade reduzida de métodos de trabalho para gerar produtividade. Uma vez que são tais métodos que levam do desenvolvimento do projeto ao resultado em si, são justamente eles que devem ser observados e, posteriormente, alterados conforme a necessidade dos projetos de Games." (LAUBISCH; CLUA, 2010, p. 183)

Sales (2013) propõe o *Agile Game Vision Management Method* (AGVMM), que é fortemente influenciado pelo Scrum. O objetivo da metodologia destacada pelo autor é trabalhar a visão do projeto para oferecer um método eficiente para o maior número de projetos possíveis.

> "[...] todo jogo é na realidade um *software* com várias particularidades, o presente artigo visa o estudo e proposta de uma metodologia híbrida para desenvolvimento de jogos a partir de características positivas de três metodologias ágeis de desenvolvimento de *software*: Scrum, Fast Driven Development e Running Lean. Tais metodologias foram selecionadas por serem ágeis e apresentarem características complementares, resultando em uma única metodologia, chamada AgiGame, com etapas e processos bem definidos e sem '*gaps*'." (POSVOLSKI *et al.*, 2014, p. 1075)

Os trabalhos correlatos apresentados nesta subseção serviram como ponto de partida para a análise da aplicabilidade das metodologias ágeis no desenvolvimento de jogos digitais

de pequeno escopo dentro do contexto do desenvolvimento *indie*. Os critérios para as análises realizadas no Capítulo 5 consideraram as experiências prévias do autor deste trabalho e as ideias e conceitos defendidos pelos autores listados neste capítulo de referencial teórico.

No próximo Capítulo, serão apresentados os referencias teóricos sobre as metodologias ágeis.

3 REFERENCIAL TEÓRICO DAS METODOLOGIAS ÁGEIS

De acordo com Beck *et al.* (2001), o Manifesto Ágil, base para as metodologias ágeis, defende como valores:

a) **Indivíduos e interações** mais que processos e ferramentas;

b) **Software em funcionamento** mais que documentação abrangente;

c) **Colaboração com o cliente** mais que negociação de contratos;

d) **Responder a mudanças** mais que seguir um plano.

Para os autores, os trechos em destaque possuem um valor maior aos não destacados, porém estes possuem suas importâncias.

Para Ferreira e Lima (2006, p. 112), as metodologias ágeis devem possuir "Condições favoráveis para as interações e as retroalimentações entre os usuários e o sistema durante todo o projeto", ser "estruturadas de modo a atender a natureza mutável e dinâmica do processo de concepção do sistema" e "[...] maior proximidade do usuário = necessidades reais".

As metodologias ágeis e ferramentas listadas nas próximas subseções foram definidas a partir do registro recorrente das mesmas em trabalhos correlatos encontrados em sites e publicações relacionadas à produção de jogos.

A utilização de uma ferramenta ou metodologia não exclui necessariamente a utilização de outra. O Kanban, por exemplo, pode ser incorporado a outras metodologias, com algumas alterações que destaquem sua características. Entretanto, o mesmo Kanban pode ser incompatível com outras metodologias que não se adéquam aos princípios da ferramenta. Essas características serão melhor explicitadas e descobertas durante a análise que este trabalho se propõe desenvolver.

As metodologias presentes neste trabalho foram escolhidas com base na citações dos textos presentes nos trabalhos correlatos.

Nas metodologias ágeis, os papéis na equipe são importantes em alguns contextos, porém, neste trabalho, apenas foram citados os papéis que podem ser correlacionados a papéis no desenvolvimento de jogos.

3.1 Scrum

Embora seja altamente adaptável, o desenvolvimento de jogos possui várias especificidades e a utilização do Scrum necessita de adaptações como propostas por Godoy e Barbosa

(2010).

Para Schwaber e Sutherland (2020, p. 4), "Scrum é um *framework* leve que ajuda pessoas, times e organizações a gerarem valor por meio de soluções adaptativas para problemas complexos" e "o *framework* Scrum é propositalmente incompleto, apenas definindo as partes necessárias para implementar a teoria Scrum. O Scrum é construído sobre a inteligência coletiva das pessoas que o utilizam."

3.1.1 Scrum Team

No Scrum, as equipes são denominadas *Scrum Team* e são compostas por 3 (três) principais atores:

a) *Scrum Master*
b) *Product Owner*
c) *Developers*

Fazendo uma analogia entre os papéis desempenhados pelas equipes responsáveis pelo desenvolvimento de jogos e o *Scrum Team*, temos que as responsabilidades de um *Scrum Master* são próximas às de um Produtor, aa de um *Product Owner* é próxima às do *Game Designer* e as do *Developer* às do restante do time.

Cabe ao *Scrum Master*, segundo Schwaber e Sutherland (2020):

a) Liderar, treinar e orientar a organização na adoção do Scrum;
b) Planejar e aconselhar implementações de Scrum dentro da organização;
c) Ajudar os funcionários e os *stakeholders* a compreender e aplicar uma abordagem empírica para trabalhos complexos;
d) Remover barreiras entre *stakeholders* e *Scrum Teams*.

Em equipes pequenas de desenvolvimento de jogos, não há uma subdivisão das funções de um produtor, como descreve Chandler (2020), então as funções diárias acabam se concentrando muito em um único elemento, que, para essa realidade, acaba tendo de cuidar da gerência e da organização do projeto, bem como da equipe, além de resolver problemas que extrapolam a parte de criação do produto.

Ao *Product Owner*, também segundo Schwaber e Sutherland (2020), cabe:

a) Desenvolver e comunicar explicitamente a meta do produto;
b) Criar e comunicar claramente os itens do *Product Backlog*;
c) Ordenar os itens do *Product Backlog*;

d) Garantir que o *Product Backlog* seja transparente, visível e compreensível.

O *Game Designer* é responsável pelo projeto/imaginação das regras do jogo e pela ambientação, entre outros aspectos do jogo, bem como pelo desenvolvimento do *Game Design Document* (GDD) que, no contexto do Scrum, deve ser utilizado de guia para o *Product Backlog*. O *Game Designer* também é requisitado sempre que surgir alguma dúvida sobre aspectos do jogo.

Ao *Developer*, ainda segundo Schwaber e Sutherland (2020), cabe a função de:

a) Criar um plano para a *Sprint* e para o *Sprint Backlog*;

b) Introduzir gradualmente qualidade, aderindo a uma definição de pronto;

c) Adaptar seu plano a cada dia em direção à meta da *Sprint*;

d) Responsabilizar-se mutuamente com os profissionais.

Por fim, os *Developers* executam todas as outras funções no desenvolvimento do projeto. Cada projeto demanda dos *Developers* habilidades e conhecimentos diferentes, que podem variar muito de acordo com os objetivos dos projetos e com a composição das equipes.

3.1.2 Eventos Scrum

Os eventos do Scrum são as ferramentas de gerenciamento, organização e melhoramento dos processos empregados na metodologia. De acordo com Schwaber e Sutherland (2020, p. 8), "esses eventos são projetados especificamente para permitir a transparência necessária. A falha em operar quaisquer eventos conforme prescrito resulta em oportunidades perdidas de inspeção e adaptação."

3.1.2.1 Sprint

A *Sprint* é o *core* do Scrum, nela é onde o produto ou jogo é de fato desenvolvido. Normalmente, tem duração fixa que pode variar de uma a mais semanas dependendo do projeto.

"*Sprints* permitem previsibilidade, garantindo a inspeção e adaptação do progresso em direção a uma meta do Produto ao menos uma vez por mês."(SCHWABER; SUTHERLAND, 2020, p. 9)

3.1.2.2 Sprint Planning

Nesse evento, como o próprio nome já diz, é feito o planejamento da *Sprint*, onde serão respondidas 3 perguntas para guiar a *Sprint* atual:

a) Por que esta *Sprint* é valiosa?

b) O que pode ser feito nesta *Sprint*?

c) Como o trabalho escolhido será realizado?

É nesta fase que o *Game Designer* na função de *Product Owner* vai definir juntamente com a equipe qual parte do projeto vai ser priorizada e o porquê. Normalmente, no desenvolvimento de jogos, mesmo os de pequenos escopos, há cortes verticais no escopo por diversos motivos, então é importante que se priorizem as mecânicas principais e todos os *assets* que as envolvem nas primeiras *Sprints*.

3.1.2.3 Daily Scrum

"O propósito da *Daily Scrum* é inspecionar o progresso em direção à meta da *Sprint* e adaptar o *Sprint Backlog* conforme necessário, ajustando o próximo trabalho planejado."(SCHWABER; SUTHERLAND, 2020, p. 10)

As *Daily Scrum* devem ser reuniões rápidas, realizadas nos primeiros 15 minutos do dia de trabalho para identificar as dificuldades e elencar possíveis soluções. Entretanto, quando falamos de desenvolvimento de jogos, dependendo da tarefa, é necessário até mais que uma reunião ao longo do dia para resolução de problemas, exatamente pela própria características extremamente mutável de um jogo e pela complexidade resultante do envolvimento de vários setores em uma única atividade.

3.1.2.4 Sprint Review

Assim como o *Sprint Planning*, o nome é auto-explicativo. É nesse evento que o time vai explicitar o que e como foram desenvolvidos os processos da *Sprint* finalizada.

"O propósito da *Sprint Review* é inspecionar o resultado da *Sprint* e determinar as adaptações futuras. O *Scrum Team* apresenta os resultados de seu trabalho para os principais *stakeholders* e o progresso em direção à Meta do Produto é discutido."(SCHWABER; SUTHERLAND, 2020, p. 11)

3.1.2.5 Sprint Retrospective

"O propósito da *Sprint Retrospective* é planejar maneiras de aumentar a qualidade e a eficácia."(SCHWABER; SUTHERLAND, 2020, p. 11)

Nesta fase, que encerra o ciclo, o *Scrum Team* discute sobre os diversos aspectos, pessoas, processos, etc. para identificar possíveis problemas e propõe "as mudanças mais úteis para melhorar sua eficácia. As melhorias mais impactantes são endereçadas o mais rápido possível. Essas podem até ser adicionadas ao *Sprint Backlog* para a próxima *Sprint*."(SCHWABER; SUTHERLAND, 2020, p. 11)

3.1.3 Scrum Artifacts

Os *Scrum Artifacts* são ferramentas e técnicas que agregam valor aos pilares de transparência e organização do Scrum. Para Schwaber e Sutherland (2020), os *Scrum Artifacts* representam trabalho ou valor. Eles são projetados para maximizar a transparência das principais informações. Assim, todos os que os inspecionam têm a mesma base para adaptação.

Para o *Scrum*, são recomendados 3 (três) artefatos, quais sejam: *Product Backlog*, que representa a meta do produto; *Sprint Backlog*, que representa a meta da *Sprint*; e Incremento, que representa a definição de pronto. No caso do desenvolvimento de jogos, um artefato que deve ser considerado de extrema importância e indispensável para a transparência do processo é o GDD, pois é neste documento que se encontram todas as descrições do jogo, de forma detalhada para guiar o time.

3.1.3.1 Product Backlog

Como já citado anteriormente, o *Product Backlog* é uma lista de atividades que devem ser desenvolvidas durante a produção. No caso de projetos dedicados ao desenvolvimento de jogos, o *Product Backlog* anda junto com o GDD, possibilitando ordenar as atividades por prioridade ou mecânicas *core* do jogo.

3.1.3.2 Sprint Backlog

O *Sprint Backlog* é um recorte do *Product Backlog*, porém mais detalhado, que irá guiar o time durante a *Sprint*.

"O *Sprint Backlog* é um plano feito por e para os *Developers*. É uma imagem

altamente visível, em tempo real do trabalho que os *Developers* planejam realizar durante a *Sprint* para atingir a Meta da *Sprint*."(SCHWABER; SUTHERLAND, 2020, p. 13)

3.1.3.3 Incremento

O Incremento é o conceito de definição de pronto.

> "A Definição de Pronto cria transparência ao fornecer a todos um entendimento compartilhado de qual trabalho foi concluído como parte do Incremento. No momento em que um item do *Product Backlog* atende a Definição de Pronto, um incremento nasce."(SCHWABER; SUTHERLAND, 2020, p. 14)

3.2 eXtreme Programming (XP)

Um jogo digital, como a própria classificação explicita, é essencialmente um software, o que abre a oportunidade para a utilização do XP.

O eXtreme Programming (XP) foi pensado para o desenvolvimento de *software* onde os requisitos são vagos e equipes são de pequeno ou médio porte.

Para Beck (1999, p. 71), o "XP virou do avesso o processo de *software* convencional. Em vez de planejar, analisar e projetar a longo prazo, o XP explora a redução no custo de mudança de *software* para fazer todas essas atividades aos poucos durante o desenvolvimento".

Para Soares (2004), o XP é uma metodologia ágil para aplicação em pequenas e médias equipes que possuam requisitos vagos e que se modificam rapidamente. Algo que ocorre com frequência no desenvolvimento de jogos. O XP apresenta 3 principais diferenças ao ser comparado com outras metodologias:

a) *feedback* constante;
b) abordagem incremental; e
c) comunicação entre as pessoas é encorajada.

3.2.1 Regras

O XP possui 13 (treze) regras de acordo com Beck (1999), que serão descritas a seguir. Os títulos de algumas regras são autoexplicativos, porém merecem alguns apontamentos.

3.2.1.1 Planejamento

Essa regra tem como base envolver os clientes na definição dos escopos e no tempo para a entrega, de acordo com estimativas fornecidas pelos programadores, que apenas implementarão as funcionalidades nas estórias da iteração atual do ciclo.

Nesse sentido, quando se trata de jogos, a figura do cliente fica com o *game designer*, pois é a função desse membro ficar responsável pelo desenvolvimento das regras e funcionamento do jogo.

3.2.1.2 Pequenas versões

Durante o processo de desenvolvimento, a cada iteração, deve-se entregar a menor versão possível, com o maior valor agregado, pois a ideia na qual essa regra se sustenta é disponibilizar o produto o mais cedo possível no mercado.

Já numa ótica aplicada a jogos, a relevância dessa regra está intrinsecamente ligada à regra de teste na visão de jogos, pois, quanto mais cedo uma *feature* é finalizada, mais rápido é possível testá-la para saber se agrega valor ao jogo ou se possui elementos que divertem ou frustram o jogador. Dependendo do *feadback*, alterações serão necessárias.

3.2.1.3 Metáfora

Essa regra está muito relacionada à comunicação da equipe, que deve criar metáforas que expliquem o resultado do projeto ou, ainda, metáforas que expliquem o resultado de partes do projeto.

Esse tipo de pensamento é muito presente no desenvolvimento de jogos, pois, desde o início desse mercado, os desenvolvedores criavam jogos que eram representações de outro artefatos da vida real, como o jogo Space Invaders (1978), que faz uma representação de ataque alienígena à terra, que se defende a partir de uma nave.

3.2.1.4 Design simples

"Diga tudo uma vez e apenas uma vez" (BECK, 1999, p. 71). Essa regra diz respeito a ser o mais conciso e "*assertivo*"possível no código, porém, pode ser levado a outras esferas do projeto, como no caso dos jogos no GDD.

3.2.1.5 Testes

Na aplicação do XP, os testes devem ser desenvolvidos primeiro do que as novas funcionalidades. Essa regra pode até funcionar quando se trata da equipe de programação, porém, em relação às outras equipes, ela não funciona muito bem, como para arte. Entretanto, quando se fala de *design*, existem testes que podem ser feitos antes da aplicação, como em papel, em tabuleiro, entre outras técnicas.

3.2.1.6 Refatoração

Aprimoramento constante do código, porém isso pode ser traduzido para outras áreas, como refino na arte, música, mecânicas etc.

3.2.1.7 Pair Programming

Todo o desenvolvimento do código deve ser construído em duplas, essa regra ajuda no compartilhamento de conhecimento, levando ao nivelamento do conhecimento da equipe, além de fortalecer um regra de "Propriedade coletiva".

Essa é uma regra que deve ser adaptada para outras equipes, as que não são de programação. Entretanto, em alguns casos, como o da arte, é difícil fazer com que artistas compartilhem a criação da arte.

Em caso de equipes pequenas que possuem 1 ou 2 programadores, que não têm capacidade de realizar a mesma tarefa juntos, pois pode comprometer o tempo da entrega ou até mesmo em casos em que só há um programador, torna-se impossível a aplicação dessa regra.

3.2.1.8 Integração Contínua

Essa é mais uma regra que está ligada à programação, em que todo novo código passa por uma bateria de testes e, caso seja aprovado, é integrado diretamente ao projeto principal.

Para as outras equipes, além da equipe de programação, não há um paralelo que se encaixe nessa regra, pois ele trata de um imediatismo que é impraticável nessas áreas.

3.2.1.9 Propriedade coletiva

Essa regra é referente ao senso de pertencimento do código, em que qualquer um tem a liberdade de melhorá-lo a qualquer momento.

O senso que tenta se empregar nessa regra é muito alinhado ao que já é feito em estúdios de jogos, principalmente em estúdios *indies*. Há sempre um senso de pertencimento do jogo para a equipe. Por mais que uma área não seja responsável por algum aspecto do jogo, existe a liberdade de influenciar, conversar e discutir com a outra equipe a qualquer momento do projeto.

3.2.1.10 Cliente presente

O cliente tem que estar em contato com o time durante todo o desenvolvimento.

Como normalmente o *game designer* é quem ocupa a posição de cliente, essa é uma regra muito simples de adaptar.

3.2.1.11 40 horas de trabalho semanal

É exatamente o que o título diz. Essa posição defende que um corpo descansado é muito mais produtivo do que aquele que passa noites trabalhando. Caso um projeto precise de constantes horas extras, existe um problema muito grave no desenvolvimento que deve ser averiguado com urgência.

3.2.1.12 Espaço Aberto

Essa regra está ligada ao espaço físico de trabalho, o qual deve ser amplo, cujo acesso dos integrantes da equipe seja fácil aos demais colegas de trabalho.

A realidade de muitas empresas *indies* é de não possuir uma sala apropriada e todo jogo ser produzido por meio de *Home Office*. Essa situação foi passada durante o desenvolvimento dos Teodoro e Super Gattai, os quais foram completamente desenvolvidos por colaboradores do grupo da universidade e empresa jr. em regime de *Home Office*. Para nos adequarmos a essa regra, estipulamos horários de trabalho em que todos entravam em reunião online ao mesmo tempo e, quando alguém precisava se comunicar com o outro, bastava abrir o microfone e todos da equipe já acompanhavam as discussões.

3.2.1.13 Just Rules

Fazer parte de uma equipe que usa XP é se comprometer a seguir as regras, porém o time pode mudá-las a qualquer momento, contanto que todos estejam alinhados.

3.3 Kanban

O Kanban é visto como uma ferramenta empregada em diversas metodologias de gerenciamento de projetos com o intuito de criar um fluxo de trabalho.

"O Kanban, ou mais precisamente o 'sistema Kanban para desenvolvimento de *software*', representa uma implementação mais direta dos princípios de Desenvolvimento Lean de Produtos para o desenvolvimento de *software* que os métodos ágeis tradicionais." (BOEG, 2010, p. 4)

A metodologia ou método Kanban é uma das mais antigas abordadas neste trabalho. Ela surgiu logo depois a 2ª Guerra Mundial, dentro das fábricas da Toyota, como uma resposta à crise econômica que o Japão enfrentava e buscava uma estratégia de reduzir os custos e aumentar a produtividade.

Também dentro de todas metodologias deste trabalho, é uma das mais simples, como cita Silva e Anastácio (2019, p. 1019): "o Kanban é uma ferramenta simples e bastante eficaz e tem com a complementação do *Just-in-time* (no tempo certo) a combinação perfeita."

Normalmente, o Kanban é utilizado em apoio a outras metodologias por ser extremamente eficiente na organização e no acompanhamento das informações, além de ser a base para diversos aplicativos de gestão, como Trello, Click Up e diversos outros.

3.3.1 Just-in-Time

Para Guimarães e Falsarella (2008, p. 131), o *Just-in-Time* (JIT) "pode ser descrito como sendo uma metodologia que está constantemente buscando a integração da organização, através do processo mais simples para permitir que o processo de mudança, direcionado pelas necessidades da sociedade, seja atendido com maior rapidez e sem desperdício."

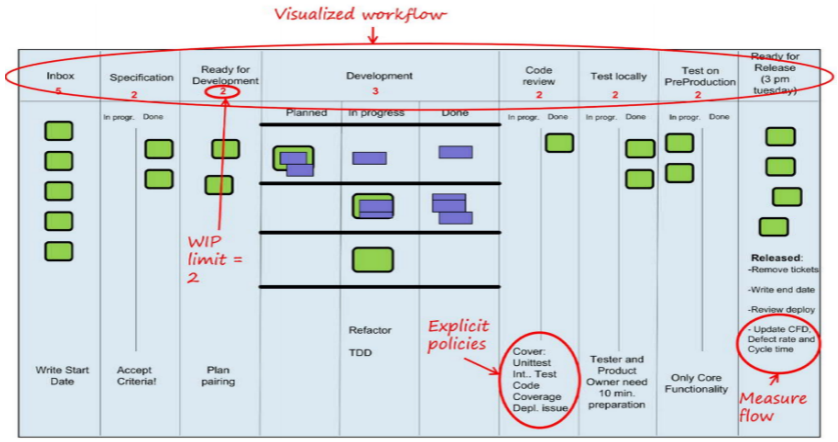
Como o JIT foi criado dentro do contexto das fábricas, para a sua aplicação em jogos, devemos pegar sua essência de integração e permissividade a mudanças para fazer sentido dentro desta visão. Os membros da equipe devem pensar bastante em todas as mecânicas, *assets*, etc., para que não haja uma grande quantidade de esforço para artefatos que não vão constar no

produto final. Isso não diz respeito aos estudos de *design* de personagem, por exemplo, mas a se a ideia de um personagem a mais é válida, se faz sentido, pois, dependendo do caso, para esse personagem extra, podem haver gastos de arte, animação dublagem etc.

3.3.2 Quadro Kanban

O quadro no Kanban é um dos artefatos mais importantes do método, pois é nele que se encontra visualmente o estado onde reside uma determinada atividade. As seções do quadro podem ser divididas de formas diferentes dependendo da necessidade da equipe ou organização que esteja utilizando. Na figura 2, encontra-se uma forma de divisão do quadro, porém existem inúmeras formas de divisão, sendo mais importante que essas seções reflitam o passo a passo das atividades em execução.

Figura 2 – Exemplo 1 de quadro Kanban.



Fonte: Boeg (2010)

3.3.3 Cartões

O cartão, assim como o quadro, é um artefato de extrema importância, pois contém as descrições das atividades que devem ser feitas ao longo do ciclo de produção ou execução de um serviço. O desenvolvimento de um produto ou execução de um serviço pode possuir diversas pequenas partes para ser completo, as quais normalmente são associadas aos cartões que serão posicionados em suas respectivas etapas no quadro Kanban. Quando uma etapa é finalizada, esse cartão é movido para a próxima seção do quadro, assim mantendo o controle de onde a atividade se encontra dentro do ciclo.

3.4 Feature-Driven Development (FDD)

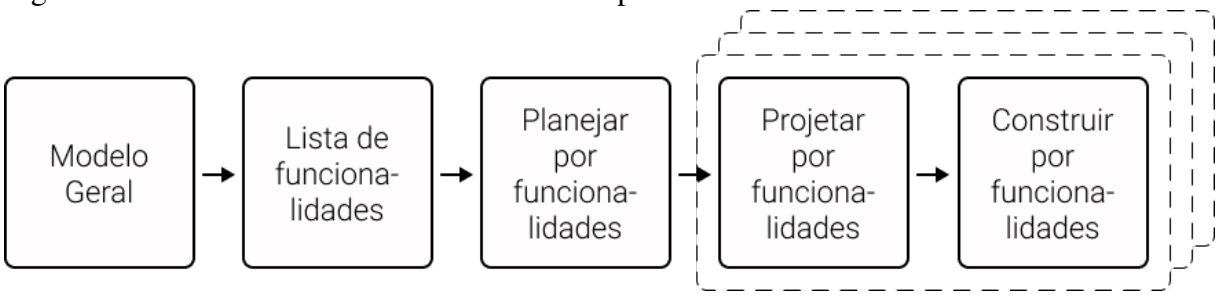
O *Feature-Driven Development* (FDD) normalmente é usado como complemento ao Scrum, adaptando a metodologia ao cenário de desenvolvimento de *softwares*.

Conforme Posvolski *et al.* (2014, p. 1081), o "FDD se mostra adequado ao desenvolvimento de jogos devido a melhorias ágeis no desenvolvimento de software que proporciona, tais como o desenvolvimento iterativo, tempo de desenvolvimento aprimorado, testes integrados e resposta rápida às mudanças de requisitos."

> "O método Feature Driven Development (Desenvolvimento Dirigido a Características) tem como objetivo promover pequenas iterações, que normalmente duram em torno de duas semanas, e resultam na entrega de uma parte do *software* funcionando." (HIGHSMITH, 2002, p. xvi)

O *Feature-Driven Development* ou FDD foi criado por Peter Coad, Jeff De Luca e Stephen Palmer. De acordo com Faccioli e Silva (2020), o FDD é um modelo de processos de projeto ágil que possibilita o desenvolvimento de software, de acordo com funcionalidades de valor para o cliente, a partir de 5 (cinco) processos, como mostrado na figura 3.

Figura 3 – Processos do Feature-Driven Development.



Fonte: Reprodução de Faccioli e Silva (2020)

A seguir, serão apresentados cada um dos processos em que o FDD se pauta.

3.4.1 Modelo Geral

Nesse processo, a equipe irá se reunir para desenvolver uma representação de alto nível do sistema, dentro do escopo e contexto propostos previamente. Após essa primeira etapa, o modelo é dividido em partes menores, mais detalhadas, de cada domínio do sistema, assim como a equipe, que também será dividida para cuidar desses domínios individuais. Posteriormente, os modelos menores são unidos novamente e os ajustes são feitos para que esses complementem uns aos outros.

3.4.2 Lista de funcionalidades

Nesse segundo processo, devem ser listadas todas as funcionalidades advindas dos modelos modelos criados no processo anterior, agrupadas por domínio de negócio. Ao final desse processo, a lista deve resultar em todos os requisitos funcionais de software.

3.4.3 Planejar por funcionalidades

O terceiro processo é simplesmente ordenar as funcionalidades pela prioridade em que serão desenvolvidas e definir as *Milestones* do projeto.

3.4.4 Projetar por funcionalidades

No quarto processo, são feitos os diagramas de classe, sequência, entre outros que irão ser utilizados no próximo processo.

3.4.5 Construir por funcionalidades

Por fim, o último processo é responsável pelo desenvolvimento, teste, pelas funcionalidades, verificação e validação das funções. Caso sejam aprovadas pelos requisitos de qualidade, são liberadas para a implantação.

3.5 Rational Unified Process (RUP)

O *Rational Unified Process* (RUP) é uma metodologia mais associada às metodologias "pesadas" ou tradicionais, que são preferencialmente aplicadas a projetos que contam com grandes equipes. Porém, em sua essência, o RUP é um método bastante customizável, o que o torna muito atrativo para os projetos voltados ao desenvolvimento de jogos com escopos pequenos e pequenas equipes. A metodologia traz como característica a divisão do problema em 4 fases bem definidas:

a) Concepção: foco na definição do escopo do projeto;
b) Elaboração: momento para a construção do plano do projeto, destacando as especificações, características e arquitetura;
c) Construção: codificação do *software*; e
d) Transição: implantação do *software* para os usuários finais.

Cada fase tem objetivos muito bem especificados, com suas próprias essências para cada etapa do problema a ser resolvido pela equipe.

> “O Rational Unified Process (RUP) é um *framework* de processo de engenharia de *software* em um formato que pode ser adaptado para uma grande variedade de projetos ou organizações. Ele fornece uma abordagem disciplinada para delegar tarefas e responsabilidades em uma organização de desenvolvimento de *softwares*.” (TAMAKI, 2007, p. 11)

Por fim, a Rational Unified Process (RUP), última metodologia abordada neste trabalho, foi desenvolvida pela Rational Software Corporation e adquirida pela IBM no início dos anos 2000. Assim como o XP e o FDD, o foco é no desenvolvimento de *softwares* e, para sua aplicação em jogos, também deve passar por adaptações.

> "O Rational Unified Process ou RUP é um processo de engenharia de *software*. Ele fornece uma abordagem disciplinada para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento. Seu objetivo é garantir a produção de software de alta qualidade que atenda às necessidades de seus usuários finais dentro de um cronograma e orçamento previsíveis."(RATIONAL SOFTWARE CORPORATION, 2001)

O RUP é organizado em 4 (quatro) fases bem definidas. Apesar de parecer um modelo "cascata"de metodologias, suas fases possuem iterações para o refino.

3.5.1 Concepção

Na fase de Concepção, os *stakeholders* devem conversar entre si e definir os objetivos do projeto, bem como o ciclo de vida. Nesta fase, é decidido se o projeto é executável ou não.

De acordo com Rational Software Corporation (2001), os objetivos desta etapa são:

a) Estabelecer o escopo do *software* do projeto e as condições de limite, incluindo visão operacional, critérios de aceitação e o que se espera encontrar no produto;
b) Discriminar os casos de usos críticos do sistema, os cenários primários de operação que conduzirão às principais compensações de design;
c) Exibir, e talvez demonstrar, pelo menos uma arquitetura candidata em alguns dos cenários principais;
d) Estimar o custo geral e o cronograma para todo o projeto;
e) Estimar os riscos potenciais (as fontes de imprevisibilidade);
f) Preparar o ambiente de suporte para o projeto.

Essa fase tem como atividades essenciais "formular o escopo do projeto", "planejar e preparar um *business case*", "sintetizar uma arquitetura para o projeto"e, por fim, "preparar o ambiente para o projeto".

3.5.2 Elaboração

Nesta fase, são levantadas as especificações do sistema e é feita a formulação de sua arquitetura, levando em consideração os riscos elencados na fase anterior.

De acordo com Rational Software Corporation (2001), os objetivos da etapa Elaboração são:

a) Garantir que a arquitetura, os requisitos e os planos sejam estáveis o suficiente e os riscos suficientemente mitigados para poder determinar de forma previsível o custo e o cronograma para a conclusão do desenvolvimento. Para a maioria dos projetos, ultrapassar esse marco também corresponde à transição de uma operação leve e rápida de baixo risco para uma operação de alto custo e alto risco com inércia organizacional substancial;
b) Tratar de todos os riscos do projeto arquitetonicamente significativos;
c) Estabelecer uma arquitetura de linha de base derivada do tratamento de cenários arquiteturalmente significativos, que normalmente expõem os principais riscos técnicos do projeto;
d) Produzir um protótipo evolutivo de componentes de qualidade de produção, bem como, possivelmente, um ou mais protótipos exploratórios e descartáveis para mitigar riscos específicos, tais como:
   – *Trade-offs* de *design* / requisitos;
   – Reutilização de componentes;
   – "Viabilidade do produto ou demonstrações para investidores, clientes e usuários finais;
e) Demonstração de que a arquitetura da linha de base oferecerá suporte aos requisitos do sistema a um custo razoável e prazo razoáveis;
f) Estabelecimento de um ambiente de apoio.

Essa fase tem como atividades essenciais "validar e definir a base da arquitetura", "refinar a visão do projeto", "criar a base de planos de iteração detalhados para a fase de construção", "refinar os casos de desenvolvimento, implementar o ambiente de desenvolvimento",

e por fim, "refinar a arquitetura e selecionar componentes".

3.5.3 Construção

Nesta terceira fase é onde, de fato, o sistema será desenvolvido seguindo todas as diretrizes estabelecidas nas fases anteriores. Vale ressaltar que o foco é colocado no gerenciamento e no controle das operações.

De acordo com Rational Software Corporation (2001), os objetivos da terceira etapa são:

a) Minimizar os custos de desenvolvimento, otimizando recursos e evitando código morto e retrabalho desnecessários;
b) Alcançar a qualidade adequada tão rápido quanto prática;
c) Alcançar versões úteis (*alfa, beta* e outras versões de teste) tão rapidamente quanto prático;
d) Concluir a análise, *design*, desenvolvimento e teste de todas as funcionalidades necessárias;
e) Desenvolver de forma iterativa e incremental um produto completo que está pronto para fazer a transição para sua comunidade de usuários. Isso implica descrever os casos de uso restantes e outros requisitos, aperfeiçoar o *design*, concluir a implementação e testar o software;
f) Decidir se o *software*, os sites e os usuários estão todos prontos para a implantação do aplicativo;
g) Atingir algum grau de paralelismo no trabalho das equipes de desenvolvimento. Mesmo em projetos menores, normalmente há componentes que podem ser desenvolvidos independentemente uns dos outros, permitindo o paralelismo natural entre as equipes (se os recursos permitirem). Esse paralelismo pode acelerar significativamente as atividades de desenvolvimento, mas também aumenta a complexidade do gerenciamento de recursos e da sincronização do fluxo de trabalho. Uma arquitetura robusta é essencial se qualquer paralelismo significativo deve ser alcançado.

A fase de construção tem como atividades essenciais a "gestão de recursos, controle e otimização de processos", "desenvolvimento e teste completos de componentes de acordo com os critérios de avaliação definidos"e, por fim, "avaliação de lançamentos de produtos em relação

aos critérios de aceitação da visão".

3.5.4 Transição

Essa última fase transição é o momento em que os testes são finalizados e o software é entregue ao cliente. Essa fase pode conter várias iterações para se adequar aos *feedback* dados pelos usuários. Quando o ciclo de vida do desenvolvimento chega ao fim, é o momento em que a fase acaba. Depois disso, um novo ciclo se inicia, sendo dedicado ao aperfeiçoamento do *software* entregue ou início do próximo projeto.

De acordo com Rational Software Corporation (2001), os objetivos da última etapa são:

a) Teste beta para validar o novo sistema em relação às expectativas do usuário;
b) Teste beta e operação paralela em relação a um sistema legado que está substituindo;
c) Conversão de bancos de dados operacionais;
d) Treinamento de usuários e mantenedores;
e) Implantação para as forças de marketing, distribuição e vendas;
f) Engenharia específica de implantação, como transição, embalagem comercial e produção, implantação de vendas, treinamento de pessoal de campo;
g) Atividades de ajuste, como correção de *bugs*, aprimoramento de desempenho e usabilidade;
h) Avaliação das linhas de base de implantação em relação à visão completa e aos critérios de aceitação para o produto;
i) Alcance da autossuficiência do usuário;
j) Alcance da concordância dos *stakeholders* de que as linhas de base de implantação estão completas;
k) Alcance da concordância dos *stakeholders* de que as linhas de base de implantação são consistentes com os critérios de avaliação da visão do projeto.

A fase de transição tem como atividades essenciais "executar planos de implantação", "finalizar o material de suporte ao usuário final", "testar o produto a ser entregue", "criar uma nota de lançamento", "obter o *feedback* dos usuários", "ajustar o produto com base no *feedback*", e por fim, "disponibilizar o produto para os usuários finais".

Godoy e Barbosa (2010) citam 2 conclusões preliminares na aplicação do *Game*

Unified Process (GUP), uma adaptação do RUP para games, nos trabalhos de Flood, em que foi notado que houve benefícios para a equipe de desenvolvimento, mesclando o RUP com o XP. Entretanto a equipe de arte não consegue lidar muito bem com as iterações do processo, bem como para os *designers* e artistas a metodologia foi comparada a uma metodologia em cascata.

O próximo capítulo versará sobre as metodologias que possibilitaram a realização da análise e conclusão dessa pesquisa.

4 METODOLOGIA

Considerando que este trabalho tem entre seus objetivos realizar uma análise dos diversos recursos propostos pelas metodologias ágeis para o desenvolvimento de jogos de pequeno escopo, é de fundamental importância compreender o contexto em que essas metodologias foram desenvolvidas e aplicadas, para fundamentar as recomendações e adequações necessárias ao desenvolvimento de jogos. Este capítulo registra o contexto em que a pesquisa foi aplicada, bem como traz informações referentes à metodologia de coleta, organização, análise e avaliação dos dados.

O presente trabalho contou com a utilização de duas metodologias. A primeira considerou os critérios de design propostos por Prodanov e Freitas (2013, p. 127), para quem o critério da pesquisa se caracteriza como "objetivo do estudo". Desta maneira, a pesquisa é classificada como "Descritiva", pois "expõe as características de uma determinada população ou fenômeno, demandando técnicas padronizadas de coleta de dados".

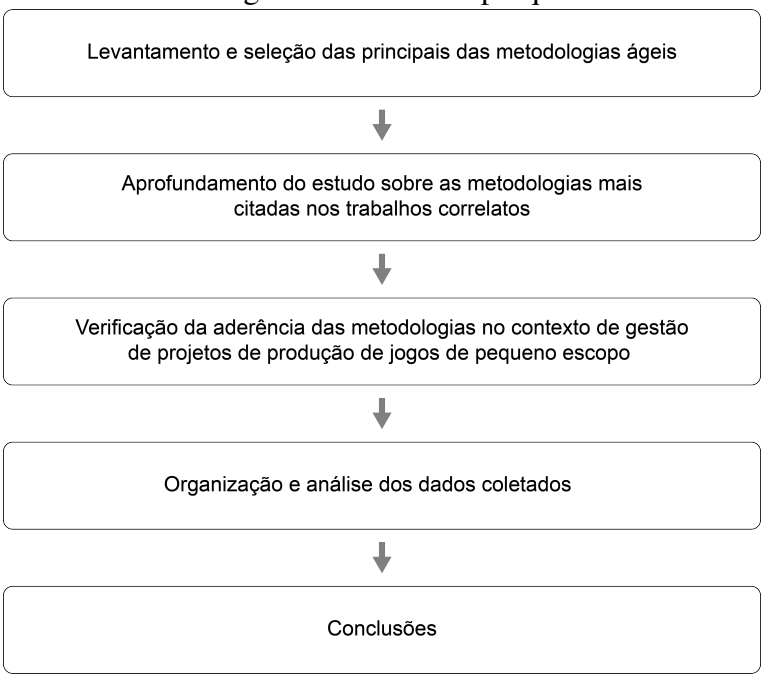
Já a segunda metodologia, proposta por Dias e Silva (2009, p. 39), refere-se à coleta de dados, para o que se adotou o método intitulado pelo autor como "Observação de Pássaros", em que não há controle dos dados, nem dos participantes:

> "Dessa forma, o pesquisador não introduz nenhuma alteração no comportamento da situação. Por outro lado, ele também não pode controlar o observado, mesmo que queira obter alguma informação adicional - somente o desenrolar normal da situação serve como material para a coleta de dados".

Além da observação, o trabalho também contou com uma pesquisa bibliográfica, realizada para proporcionar a sua fundamentação teórica. Para Gil (2010), a pesquisa bibliográfica possibilita o acesso a um número muito maior de dados e informações do que a realização de uma pesquisa a partir de fontes primárias. Escolher o tema, realizar o levantamento preliminar, buscar as melhores fontes, realizar a leitura do material e conduzir a organização lógica do assunto. Todas essas etapas foram seguidas ao longo deste trabalho para alcançar os resultados desejados com a realização da pesquisa bibliográfica.

Para o desenvolvimento da pesquisa, foi seguido um passo a passo, representado no *framework* da metodologia na figura 4.

Figura 4 – *Framework* da metodologia do trabalho de pesquisa.



Fonte: O autor.

A seguir, será abordada cada uma das etapas do *framework*.

4.1 Levantamento e seleção das principais das metodologias ágeis aplicadas em gestão de projetos de jogos

a) Para fazer o levantamento das metodologias ágeis, primeiramente foram pesquisados os trabalhos correlatos em agregadores de trabalhos acadêmicos, como SBGames, ou motores de busca, como Google e Google Scholar, utilizando as palavras-chave: "jogos", "*games*", "produção", "gestão", "gerenciamento de projeto", "metodologias ágeis"e "produtor".
b) Durante a pesquisa, foram encontrados 6 trabalhos correlatos, porém apenas 4 foram selecionados, pois os demais possuíam conteúdo similar;
c) Todas as metodologias ágeis citadas nos trabalhos correlatos foram utilizadas neste trabalho e em todos eles o Scrum esteve presente , enquanto as outras foram citadas ao menos uma vez.

4.2 Aprofundamento do estudo sobre as metodologias mais citadas nos trabalhos correlatos

a) A partir das metodologias selecionadas, um novo levantamento foi realizado em agregadores de trabalhos acadêmicos, como Google Scholar, ERIC e Periódico da CAPES, com o objetivo de aprofundar o conhecimento sobre cada uma das metodologias identificadas.
b) Seleção de textos com base nos seguintes critérios:
   – Autores criaram a Metodologia ou o termo;
   – Autores que são referências sobre o assunto;
   – Autores citados nos trabalhos correlatos.
c) Notação das suas especificidades, facilidades, dificuldades e métodos;
d) As notas foram tomadas em documento separado em forma de observações que foram reorganizados no quadro 3.

4.3 Verificação da aderência das metodologias no contexto de gestão de projetos de produção de jogos de pequeno escopo

a) A verificação da aderência das metodologias com a gestão de projetos de produção de jogos de pequeno escopo se deu a partir da revisão bibliográfica realizada no início deste trabalho, da análise dos registros encontrados nos trabalhos correlatos e da experiência adquirida como gestor de empresas e grupos especializados em desenvolvimento de jogos, como a empresa júnior do curso de Sistemas e Mídias Digitais (SMD) - The Guardian Dog Studio (TGD Studio), na empresa CosMonkeys Studio e em trabalho realizados nas disciplinas do SMD.
b) As empresas e grupos citados anteriormente possuíam a seguinte composição:
   – TGD Studio (3 produtores, 2 pessoas de marketing, 3 *game designers*, 4 artistas e 4 programadores);
   – CosMonkeys Studio (1 produtor/programador e 1 artista/*designer*);
   – Grupo das diciplinas [em geral] (1 gestor, 1 *designer*, 1 artista e 1 programador).
c) Para verificar a aderência das metodologias em projetos dedicados à produção de jogos de pequeno escopo, elaborou-se um roteiro tendo as seguintes questões

como balizadoras:

- Como essa metodologia pode auxiliar em cada um dos ciclos propostos por Chandler?
- Como essa metodologia pode atrapalhar em cada um dos ciclos propostos por Chandler?
- Quão complexa é essa metodologia para equipes sem muita experiência de gestão?
- Quais os principais pontos que facilitam a gestão de projetos de pequeno escopo?

4.4 Organização e análise dos dados coletados

a) A partir das notas tomadas, dos apontamentos dos trabalhos correlatos e da experiência acumulada como gestor de projetos, as informações foram organizadas de acordo com a estrutura proposta no Quadro 3.
b) O Quadro 3 permitiu o agrupamento das respostas de modo a evidenciar os pontos positivos e negativos de cada uma das metodologias quando aplicadas aos ciclos básicos de produção de jogos e nos ciclos básicos de produção por área, fazendo uma referência aos grupos criados por Chandler (2012) e mencionados no referencial teórico desse trabalho.
c) Para os ciclos básicos de produção de jogos, foi considerada a aplicação de cada uma das metodologias nas fases de pré-produção, produção, testes e finalização.
d) Para os ciclos de produção por área, as mesmas metodologias foram avaliadas a partir da sua aplicação nas áreas do design de jogos, artística, de programação e no trabalho do conjunto.

4.5 Conclusões

a) A análise permitiu, com base na simplicidade e similaridade com os ciclos propostos por Chandler (2012), identificar quais metodologias são as mais adequadas e recomendadas para aplicação em projetos voltados para a produção de jogos de pequeno escopo.

Quadro 3 – Quadro base para análise de cada metodologia

METODOLOGIA	-		
ETAPA	**POSITIVO**	**NEGATIVO**	**OBSERVAÇÕES GERAIS**
CICLO BÁSICO DE PRODUÇÃO DE JOGOS			
PRÉ-PRODUÇÃO	-	-	-
PRODUÇÃO	-	-	-
TESTES	-	-	-
FINALIZAÇÃO	-	-	-
CICLOS DE PRODUÇÃO POR ÁREA			
DESIGN	-	-	-
ARTÍSTICO	-	-	-
PROGRAMAÇÃO	-	-	-
CONJUNTO	-	-	-

Fonte: O autor.

Sobre o preenchimento do quadro 3, foram considerados como pontos positivos aqueles que contribuem para o melhor desempenho em cada uma das etapas ou áreas identificadas nos ciclos básico de produção de jogos e de produção por área, enquanto os pontos negativos foram aqueles que se mostraram em desacordo com as necessidades e demandas de cada uma das etapas ou áreas dos ciclos de produção.

Por fim, a coluna de "observações gerais” foi utilizada para registrar os comentários sobre diversos temas relacionados ao desenvolvimento de jogos ou sobre a metodologia, que não foram possíveis de classificar como um ponto positivo ou negativo.

Vale destacar a importância da pesquisa bibliográfica realizada, pois permitiu a construção da fundamentação teórica para o preenchimento dos quadros apresentados no próximo Capítulo - Análises e discussão dos resultados e o embasamento dos pontos teóricos e técnicos da análise.

Neste Capítulo, foram apresentadas as especificidades relacionadas à metodologia utilizada neste trabalho. A seguir, será apresentada a analise e as discussões dos resultados gerados a partir da pesquisa.

5 ANÁLISES E DISCUSSÃO DOS RESULTADOS

Neste capítulo, será utilizado o modelo proposto no Quadro 3 para a análise individual, feita pelo autor, de cada uma das metodologias apresentadas ao longo da discussão do trabalho. Para a análise, foram utilizados os critérios estabelecidos no capítulo 4. Após a apresentação dos quadros, os resultados individuais de cada um serão discutidos e, ao final, encontra-se uma análise mais global de todas as metodologias abordadas neste trabalho.

Vale destacar a importância da pesquisa bibliográfica apresentada nos Capítulos anteriores, pois ela permitiu a construção da fundamentação teórica para o preenchimento dos quadros presentes neste Capítulo e o embasamento dos pontos teóricos e técnicos da análise.

Para a analise individual das metodologias estudadas, foi utilizado como critério a quantidade de pontos positivos e negativos identificados em sua aplicação, bem como as especificidades e características que foram notadas ao longo das análises.

No Quadro 4, estão registrados os pontos positivos, negativos e as observações gerais sobre a aplicação do framework Scrum, em cada uma das etapas de desenvolvimentos de jogos. Na sequência, o Quadro 5, o Quadro 6, o Quadro 7 e o Quadro 8 trazem as análises das metodologias eXtreme Programming, Kanban, FDD e RUP, respectivamente.

Na visão mais globa, foi dada mais atenção às características qualitativas e à forma como as metodologias se correlacionam, suas construções e filosofias para a aplicação em desenvolvimento de jogos de pequeno escopo. A seguir, a analise individual de cada metodologia em seus respectivos quadros com os pontos positivos, negativos e observações gerais em cada um dos 8 ciclos descritos por Chandler (2012).

Quadro 4 – Quadro de análise do *Scrum*

METODOLOGIA	Scrum		
ETAPA	**POSITIVO**	**NEGATIVO**	**OBSERVAÇÕES GERAIS**
CICLO BÁSICO DE PRODUÇÃO DE JOGOS			
PRÉ-PRODUÇÃO	Auxilia a delimitar a quantidade de esforço a ser empregada em uma determinada área.	Esta é uma fase que possui muitas provas de conceito, modificações constantes, envolve muita arte e elementos pouco pragmáticos quando se trata da da parte de ideação do jogo.	Nessa etapa do desenvolvimento, a aplicação do Scrum não é aconselhável a times com pouca experiência, pois pode acabar gerando frustrações e problemas de design que podem atrapalhar nas fases posteriores do ciclo.
PRODUÇÃO	Auxilia em entregas constantes e gera sensação de completude.	Pode causar problema de comunicação nas equipes.	Essa fase do ciclo básico é a que mais se adéqua a utilização do *Scrum*.
TESTES	-	-	A própria natureza desse ciclo é ser repetitivo e pragmático. Então, a utilização do Scrum não altera muito as tarefas.
FINALIZAÇÃO	Auxilia muito na priorização da finalização dos elementos mais importantes, organização e previsão das atividades.	-	-
CICLOS DE PRODUÇÃO POR ÁREA			
DESIGN	Auxilia na delimitação de prazos, o que leva a um trabalho mais seletivo, reforçando os pontos fortes do *design* a ser criado.	Pode parecer um limitante da criatividade no início da aplicação para os *game designers*.	-
ARTÍSTICO	Auxilia na consistência das entregas da equipe de arte, que, por muitas vezes, dada sua natureza "artística", tende a levar "o tempo que for" para a finalização de um *asset*.	Resistência da equipe.	-
PROGRAMAÇÃO	Dentre os ciclos por área, é o mais simples de se implantar, pois as equipes que cuidam de programação já são acostumadas a utilizar diferentes metodologias no seu trabalho.	-	-
CONJUNTO	Aumenta a produtividade da equipe.	Pode apresentar diversas dificuldades caso não haja adaptação pra realidade de jogos.	Deve-se ter cuidado, pois, em determinados momentos do ciclo básico, algumas áreas possuem mais tarefas que outras, e isso pode impactar na percepção da efetividade do uso da metodologia.

Fonte: O autor.

Quadro 5 – Quadro de análise do *eXtreme Programming*

METODOLOGIA	*eXtreme Programming* (XP)		
ETAPA	**POSITIVO**	**NEGATIVO**	**OBSERVAÇÕES GERAIS**
CICLO BÁSICO DE PRODUÇÃO DE JOGOS			
PRÉ-PRODUÇÃO	Por ser uma metodologia com foco em pequenas versões, *design* simples e testes, acaba tendo um ótimo desempenho nessa fase do processo.	Algumas regras, como a de Refatoração, não são tão úteis nesse estágio, pois, em muitos casos, o código *assests*, entre outros, acaba sendo descartado e retrabalhado do zero na etapa de Produção, além da dificuldades de adequar as equipes de *design*, arte, música às regras de base estabelecidas.	O XP por si só pode funcionar nessa fase, por não ter objetivos muito concretos, pois é uma fase de muito descobrimento.
PRODUÇÃO	Dentre todos os ciclos básicos, é onde melhor se enquadra.	Assim como todos os ciclos básicos, pode ser de difícil implantação para as equipe que possuem menos pragmatismo.	Se as regras estabelecidas forem seguidas, principalmente a de "40 horas de trabalho semanal", pode evitar as práticas prejudiciais de *crunch*.
TESTES	Uma das principais caraterísticas do XP é pensar nos testes antes de qualquer desenvolvimento, então se aplica muito a este ciclo.	-	-
FINALIZAÇÃO	-	-	A aplicação nessa fase, de modo aparente, não possui impactos positivos ou negativos além dos comentados anteriormente.
CICLOS DE PRODUÇÃO POR ÁREA			
DESIGN	-	Algumas regras-base podem ser prejudiciais para a equipe de *design*, por conta da proposta de ser uma metodologia com foco em programação.	-
ARTÍSTICO	-	Algumas regras-base podem ser prejudiciais para a equipe de arte, por conta da proposta de ser uma metodologia com foco em programação.	-
PROGRAMAÇÃO	É uma metodologia pensada por e para os desenvolvedores.	-	-
CONJUNTO	Pode tornar o processo de desenvolvimento de jogos de pequeno escopo bem mais facilitado.	Pode atrapalhar em algumas equipes.	Deve sofrer várias alterações para a aplicação nas diversas equipes, pois não foi pensado para equipes multidisciplinares.

Fonte: O autor.

Quadro 6 – Quadro de análise do *Kanban*

METODOLOGIA	Kanban		
ETAPA	**POSITIVO**	**NEGATIVO**	**OBSERVAÇÕES GERAIS**
CICLO BÁSICO DE PRODUÇÃO DE JOGOS			
PRÉ-PRODUÇÃO	Nessa primeira etapa, é uma excelente ferramenta para criar um *backlog* com todas as atividades que possam ser desenvolvidas durante todo o desenvolvimento dos jogos e de fácil adesão a todas as equipes.	-	-
PRODUÇÃO	Assim como na Pré-produção, é uma ferramenta que auxilia muito na gestão das atividades.	Apenas a utilização do Kanban nessa fase pode dar a falsa impressão de organização, porém os Cartões Kanban podem ser rasos e acabar atrapalhando mais do que ajudando.	-
TESTES	-	-	Não possui uma aplicação tão detalhada para essas fase da produção.
FINALIZAÇÃO	-	-	Assim como em Teste, não há muito o que detalhar nessa fase, pois o Kanban é muito mais sobre fazer um planejamento prévio e segui-lo.
CICLOS DE PRODUÇÃO POR ÁREA			
DESIGN	-	Nos momentos inicias do desenvolvimento, pode acabar engessando o processo criativo do *designers*.	-
ARTÍSTICO	Auxilia na listagem de todos os *assets* que devem ser desenvolvidos, além de criar definições de estados para os processos.	-	-
PROGRAMAÇÃO	Assim como na arte, auxilia nos estados de desenvolvimento.	-	-
CONJUNTO	Como um todo, o Kanban sozinho pode auxiliar no processo de desenvolvimento.	Entretanto a utilização exclusiva do KANBAN pode não suprir todas as necessidades de desenvolvimento. Deve ser combinado com outra metodologia.	Pode ser uma excelente ferramenta para introduzir o conceito de Metodologias Ágeis a equipes que não possuem nenhuma experiência com essas práticas.

Fonte: O autor.

Quadro 7 – Quadro de análise do *Feature-Driven Development*

METODOLOGIA	*Feature-Driven Development* (FDD)		
ETAPA	**POSITIVO**	**NEGATIVO**	**OBSERVAÇÕES GERAIS**
CICLO BÁSICO DE PRODUÇÃO DE JOGOS			
PRÉ-PRODUÇÃO	Esta etapa se assemelha bastante com a primeira etapa de Modelo Geral do FDD, então elas acabam se complementando.	Neste trabalho, está sendo observado a aplicação com a lente para projetos de pequeno escopo, entretanto vale mencionar que, para projetos de maior complexidade, a aplicação nesta fase pode ser um entrave.	-
PRODUÇÃO	As etapas descritas pela metodologia acabam auxiliando bastante na organização das atividades e nas iterações durante esse ciclo.	Pode acabar gerando um certo ruído na separação das equipes na geração do modelo geral.	Assim como em outras metodologias, deve passar por algumas modificações para se adequar à realidade de jogos.
TESTES	As funcionalidades assim como no XP devem ser pensadas junto com os testes, onde estes devem ser desenvolvidos antes que as próprias.	-	-
FINALIZAÇÃO	A última etapa do FDD está muito ligada a este ciclo, então acabam sendo complementares.	-	-
CICLOS DE PRODUÇÃO POR ÁREA			
DESIGN	A etapa de Modelo Geral acaba auxiliando bastante como uma ferramenta para a equipe de *design*.	Pode acarretar mais tempo de acompanhamento das outras equipes pelo fato de os modelos serem ser elaborados de forma separada antes de se unirem.	-
ARTÍSTICO	-	-	Possui as mesmas características descritas pelo ciclo de *design*.
PROGRAMAÇÃO	Assim como XP e RUP, o FDD foi pensando para os desenvolvedores, que são os que acabam se beneficiando das técnicas.	-	-
CONJUNTO	Como um todo, a metodologia é a que mais se encaixa, sem grandes alterações, ao desenvolvimento de jogos de pequenos escopo.	-	-

Fonte: O autor.

Quadro 8 – Quadro de análise do *Rational Unified Process*

METODOLOGIA	*Rational Unified Process* (RUP)		
ETAPA	**POSITIVO**	**NEGATIVO**	**OBSERVAÇÕES GERAIS**
CICLO BÁSICO DE PRODUÇÃO DE JOGOS			
PRÉ-PRODUÇÃO	O RUP pode ajudar muito no processo de definir o projeto viável a ser desenvolvido, o que é um ponto crucial, ainda mais se tratando de projetos com orçamento apertado, como é o caso da maioria dos jogos de pequeno escopo.	Entretanto pode ser uma metodologia muito pesada para projetos pequenos ou para quem está tendo os primeiros contatos com desenvolvimento ágil.	-
PRODUÇÃO	-	Envolve processos muito pesados para a dinamicidade que um jogo pode ter durante o desenvolvimento.	-
TESTES	-	A fase de teste prevista pelo RUP acaba ficando para fases posteriores, o que pode prejudicar, já que, em jogos, é necessário testar os conceitos desde o início do desenvolvimento.	Vale ressaltar que os POC são muito importantes no desenvolvimento de jogos, enquanto no RUP os *Alphas Products* (uma versão inicial do produto, com o mínimo viável) possuem mais destaque.
FINALIZAÇÃO	A sua última etapa, a de *transition*, encaixa-se perfeitamente com esse ciclo e está muito alinhada às atividades que são executadas nesta fase.	-	-
CICLOS DE PRODUÇÃO POR ÁREA			
DESIGN	Ajuda bastante na construção do GDD, na organização e no acompanhamento das atividades das equipe de *design*.	-	-
ARTÍSTICO	-	As equipes de arte acabam tendo dificuldades de acompanhar as iterações do RUP.	-
PROGRAMAÇÃO	Assim como as outras metodologias com foco em desenvolvimento, essa acaba beneficiando bastante os desenvolvedores.	-	-
CONJUNTO	-	Acaba sendo uma metodologia muito pesada para todas as equipes dedicadas a projetos de pequeno escopo.	-

Fonte: O autor.

5.1 Análises e discussões dos Quadros

Nesta seção, serão apresentadas as discussões acerca de cada uma das metodologias analisadas, a partir dos resultados registrados em seus respectivos quadros, começando com o Scrum.

5.1.1 Scrum

O *framework* Scrum é uma das metodologias ágeis mais utilizadas e uma das mais frequentemente encontradas em trabalhos de diferentes áreas, por atender às necessidades da produção industrial, de agências publicitárias e de desenvolvedores de jogos. A metodologia também é muito flexível para a utilização em diversas áreas de conhecimentos multidisciplinares, o que a credencia como extremamente compatível com o desenvolvimento de jogos o qual necessita de equipes de conhecimentos e habilidades plurais.

O Scrum, dentre todas as metodologias ágeis analisadas neste trabalho, é aquela que possui mais adesão no mercado por ser um *framework* que pode ser utilizado com diversas ferramentas diferentes, ser integrado a outras metodologias e ser de fácil adaptabilidade. Sua adesão não é grande apenas no mercado de jogos, mas também em áreas diversas que vão além de produtos multimídias ou sistemas.

Para o desenvolvimento de um jogo, independentemente de seu escopo, há uma necessidade de conhecimentos diversos, como arte, *game design*, UX/UI, programação, música, entre outros, o que se adéqua perfeitamente ao pilar da multidisciplinaridade do Scrum, pois a metodologia recomenda a formação de equipes com conhecimentos diversos.

Quando se trata de estúdios *indies*, essa multidisciplinaridade acaba se sobressaindo, pois, na maioria dos casos, a equipe é formada por profissionais das diversas áreas, normalmente 1 (um) de cada, contribuindo para a composição de 1 (um) *Scrum Team*, onde cada um dos profissionais deve desempenhar suas funções para garantir as entregas periódicas do jogo.

Em se tratando dos Eventos Scrum, mais especificamente do *Sprint*, para o desenvolvimento de jogos de pequeno escopo, na minha experiência, *Sprints* de 1 (uma) a 2 (duas) semanas costumam ser muito mais benéficas ao desenvolvimento, pois, como o processo de construção de um jogo é extremamente mutável, uma *Sprint* de duração superior a isso acaba ficando obsoleta.

Alguns dos problemas da aplicação do Scrum em jogos citados por Godoy e Barbosa

(2010) vêm da natureza multidisciplinar da equipe, que, embora crie um ambiente que incentiva a criatividade, pode acabar criando também uma divisão entre as equipes de arte e programação, o que pode dificultar a comunicação eficaz. Ainda segundo o autor, existe a dificuldade do "Incremento", pois, quando tratamos de jogos de entretenimento, o foco deve ser a diversão e, para isso, não há uma técnica para descrever que esse objetivo foi cumprido.

Quando aplicada a metodologia em seus trabalhos, foi a ausência do GDD, contendo informações que vão além das técnicas, uma vez que apenas os registros encontrados no *Product Backlog*, que não supriu as necessidades do projeto. Também foi percebido o conflito nas *Daily Scrum*, em que a equipe tendia a discutir sobre o *Game Design* ao invés de inspecionar o progresso do evento em direção à sua meta.

A análise do framework Scrum, referente ao quadro 4, possibilitou a identificação de pontos positivos em 7 (sete) dos 8 (oito) ciclos apresentados por Chandler (2012). Entretanto também possibilitou a identificação de pontos negativos em 5 (cinco) dos critérios analisados. Sua construção permite a utilização em vários campos de atuação e da importância para equipes multidisciplinares como as quais os estúdios de jogos se enquadram.

A maior barreira para utilização dessa metodologias em equipes que trabalham com escopo pequeno é a sincronização das equipes, pois dentro do desenvolvimento de jogos cada uma possui data de entrega distinta e deve-se levar em consideração que equipes pequenas acabam não possuindo um produtor que tem exatamente essa função de organização, então a implantação pode ser de um nível de dificuldade alto, além de criar algumas burocracias, como as *daily meetings*, que podem ser interpretadas de forma errônea, como reuniões para discutir sobre o *design* do jogo.

5.1.2 eXtreme Programming

Assim como o Scrum, pelas especificidades do desenvolvimento de um jogo, o XP não pode ser utilizado sem adaptações. Porém, diferentemente do Scrum, que se propõe a resolver problemas mais gerais de áreas multidisciplinares, o XP tem como foco resolver problemas mais ligados à área de desenvolvimento de *softwares*.

O XP é mais conhecido por sua aplicação em desenvolvimento de *softwares*. No entanto um jogo digital não deixa de ser um *software*, porém com suas próprias particularidades aplicadas ao seu conceito. Por esse fato, algumas das regras não conseguem ser aplicadas a todas as equipes necessárias para o desenvolvimento de um jogo.

Para a utilização do *eXtreme Programming* no desenvolvimento de jogos, são necessárias muitas adaptações para todas as equipes as quais não trabalham com codificação, porém não há uma explicação como trabalhar com equipes multidisciplinares.

O XP possui um equilíbrio de 5 (cinco) para 5 (cinco) entre os pontos positivos e negativos referente à análise do quadro 5. A característica mais expressiva que a metodologia apresentou durante a análise foi exatamente seu foco para o desenvolvimento de *softwares*. Um jogo digital, de fato, é um *software*, porém exige outro pensamento no seu desenvolvimento, em que as adaptações realizadas para a utilização única dessa metodologia poderia acabar perdendo sua essência.

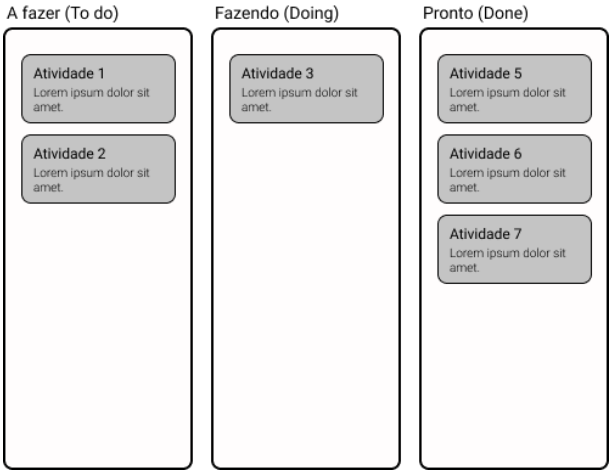
Idealmente, a adoção do XP poderia se dar como complemento à outra metodologia, como o Scrum, principalmente por parte da equipe de programação, porém vale o questionamento se equipes com escopo pequeno se adaptariam bem à utilização de 2 (duas) metodologias em conjunto, já que apenas uma traria custo operacional grande, então a adoção de 2 (duas) metodologias ágeis poderia acabar sendo inviável.

5.1.3 Kanban

Dentre todas as metodologias abordadas neste trabalho, o Kanban é o mais simples para a introdução em equipes que nunca trabalharam com alguma metodologia, pois é de fácil entendimento e aplicação. Em se tratando da aplicação do Desenvolvimento Lean, o foco do Sistema Kanban está em evitar desperdícios e resolver problemas de forma sistemática.

Como citado no referencial teórico, a metodologia é muito associada a outras metodologias, mas também é vista sendo utilizada por times iniciantes na sua forma mais simples apenas como uma quadro de "*To do*", como apresentado na figura 5. Pela sua simplicidade, é muito fácil de aplicar em qualquer projeto independentemente do escopo, para uma melhor organização.

Figura 5 – Exemplo de quadro "*To do*"no Kanban.



Fonte: O Autor.

Dentre as 5 (cinco) metodologias apresentadas neste trabalho, essa é a de maior facilidade da implantação em equipes pequenas e que possuem o escopo reduzido de pouco tempo de trabalho. Sua simplicidade e intuitividade acabam auxiliando bastante nas explicações para a equipe.

Com um balanço de 5 (cinco) pontos positivos e 3 (três) pontos negativos, referentes ao quadro 6, apresenta uma boa opção para a implantação em projetos mais simples, porém, mesmo com escopos pequenos, pode não resolver problemas de gerenciamento mais complexos que possam eventualmente aparecer.

5.1.4 Feature-Driven Development

Assim como o XP, a sua aplicação no desenvolvimento de jogos demanda pequenas alterações, pois suas diretrizes são muito voltadas para o pensamento lógico, enquanto na produção de jogos, além do pensamento lógico, ainda temos que acomodar o pensamento artístico, que segue um fluxo diferente.

Essa metodologia tem o foco no desenvolvimento de softwares e sua aplicação deve ser adaptada quando aplicada ao contexto de desenvolvimento de jogos. Assim como as outras metodologias citadas, pode ser utilizada em combinação com diferentes metodologias.

O modelo geral proposto pela metodologia FDD para jogos de escopo médio ou grande pode ser muito prejudicial, pois o jogo tende a mudar bastante quanto maior for seu escopo. Mas, tratando de projetos de escopo menor, podem auxiliar no foco que esse jogo pode tomar.

Não se pode afirmar que o FDD é a melhor metodologia. Entretanto, durante os

meus momentos de tentar encontrar soluções para o gerenciamento de desenvolvimento de jogos, cheguei a ideias que se aproximavam dessa metodologia, por conta do seu nível de complexidade suficientemente equilibrado para projetos de escopos pequenos e médios, como os que tive que desenvolver ao longo das disciplinas da Universidade, em projetos da TGD Studio e meu estúdio de desenvolvimento de jogos, por exemplo.

O FDD foi a metodologia que apresentou maior diferença entre pontos positivos e negativos, sendo 7 (sete) para os positivos e 3 (três) para os negativos. Isso se deve ao fato de a sua filosofia conseguir ser pensada para equipes multidisciplinares e para o desenvolvimento de *softwares* ao mesmo tempo, diferentemente das metodologias apresentadas. Entretanto a sua implantação em equipes pequenas pode possuir uma resistência grande e uma difícil adaptação, por ser uma metodologia que se demonstrou bastante pesada com inúmeras etapas que podem acabar desvirtuando projetos de pequeno escopo.

5.1.5 Rational Unified Process

Dentre as metodologias abordadas neste trabalho, essa foi a única que apresentou mais pontos negativos do que positivos, sendo 4 (quatro) positivos e 5 (cinco) negativos. Isso se deve ao fato de ser uma metodologia que, mesmo ágil, pesada, com um nível de complexidade alto o qual não é recomendado para a utilização em projetos com o escopo reduzido, apesar de ter várias soluções para problemas mais complexos.

5.2 Visão Geral

Em uma visão mais geral, foi notado que as metodologias acabam auxiliando bastante no desenvolvimento de jogos, entretanto algumas áreas, como arte e *game design*, acabam não tendo facilidade de aderir a algumas características propostas pelas metodologias apresentadas, pela própria natureza das suas atividades. Para que essas áreas pudessem ser contempladas, tiveram que haver diversas alterações nas metodologias. Em contraponto, as equipes de tecnologia acabam se beneficiando muito com a adesão de qualquer uma das metodologias propostas.

Outra observação é que todas as metodologias com foco em jogos pesquisadas neste trabalho, eram adaptações de outras já existentes, tentando encaixar as especificidades da produção *games* nas bases já estabelecidas dessas metodologias.

As metodologias que possuem menos passos ou etapas e são mais abertas a adap-

tações ou modificações acabam tendo mais pontos positivos para a implantação em jogos de pequeno escopo.

Algumas metodologias devem ser combinadas, pois, mesmo com algumas alterações, acabam não suprindo todas as necessidades do desenvolvimento, o que acaba criando uma complexidade a mais para equipes que estão começando a utilizar metodologias ágeis.

Dentre as metodologias pesquisadas neste trabalho, as que possuem maior potencial na utilização de projetos de pequeno escopo são a Kanban, pela sua simplicidade, e a *Feature-Driven Development*, por sua filosofia coincidir bastante com o desenvolvimento de jogos. Apesar das duas serem utilizadas como complemento do Scrum, este não é recomendado, principalmente pelo ruído de comunicação gerado por ocasião da dificuldade da separação da *Daily* Scrum e da reunião de discussão do projeto por parte de equipes menos experientes.

Neste Capítulo, foram apresentadas as análises e discussões acerca da aplicação das metodologias ágeis aplicadas no contexto de desenvolvimento de jogos de pequeno escopo, seguindo a metodologia apresentada no Capítulo 4, separando a utilização de cada uma nos ciclos proposto por Chandler (2012). A seguir, no Capítulo final, será apresentada a conclusão do texto, bem como a proposição de trabalhos futuros.

6 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho foi realizada uma análise da aplicabilidade de 5 (cinco) metodologias ágeis (Scrum, XP, Kanban, FDD e RUP) no contexto de desenvolvimento de jogos de pequeno escopo. Para tal, foi utilizada a metologia "Descritiva"proposta por Prodanov e Freitas (2013), juntamente com a técnica de coleta de dados da "Observação de Pássaros"descrita por Dias e Silva (2009).

Ao longo de toda a pesquisa, diversos autores relataram que a aplicação de metodologias ágeis em seus projetos foram benéficas, apesar das dificuldades e adaptações que tiveram de ser feitas para a implantação em seus projetos. Nas experiências de projetos executados na Universidade e na Empresa Jr. em que o próprio autor participou, tais observações se mantiveram válidas.

Na aplicação das metodologias, as equipes de programação acabaram sendo as mais beneficiadas com a adoção de metodologias ágeis, entretanto, por mais que as equipe de *design* e arte tivessem benefícios, parte do seu trabalho, que é menos objetivo, como a concepção de personagens, por exemplo, acabava sendo deixado de lado por conta das regras estabelecidas pelas próprias metodologias, principalmente as que foram criadas para resolver problemas ligados ao desenvolvimento de *softwares*.

Vale destacar que, diante da pesquisa feita, as metodologias ágeis focadas em jogos que foram encontradas eram apenas adaptações de outra metodologia já existente, o que suscita a pergunta: "será que o desenvolvimento do zero de uma metodologia ágil com foco em jogos não se faz necessário para que esses problemas de adaptações sejam resolvidos?". Já temos mais de 20 (vinte) anos desde a criação do Manifesto Ágil, muito evoluiu nessas 2 (duas) últimas décadas, o que pode tornar menos trabalhoso esse desenvolvimento devido ao extenso conhecimento acumulado.

Entretanto, diante da presente pesquisa, como supracitado no Capítulo anterior, sugerimos a utilização das metodologias ágeis Kanban e Feature-Driven Development FDD, po estarem mais alinhadas com as necessidades que um desenvolvimento de jogo de pequeno escopo exige dentro dos ciclos estabelecidos por Chandler (2012).

Existem inúmeras metodologias ágeis e sempre estão aparecendo novas para resolver problemas específicos. Por essa razão, apenas 5 (cinco), a princípio, foram escolhidas para serem estudadas neste trabalho, considerando-se como critério as metodologias mais citadas pelos autores nos trabalhos relacionados.

Também deve-se citar a dificuldade de aplicar e testar cada uma das metodologias em um trabalho de conclusão de curso de graduação, pela própria natureza dos projetos, que exigem bastante tempo para serem executados e analisados, mesmo tendo pequenos escopos.

Outra limitação do trabalho refere-se, do ponto de vista empírico, à interpretação do trabalhos correlatos e a observações das aplicações das metodologias ágeis em jogos, as quais foram feitas por apenas um sujeito com conhecimento de causa sobre os assuntos abordados e se recomenda que outros possam fazer a análise das metodologias expostas no texto.

Como possíveis trabalhos futuros, podem-se citar: A adição de análise de outras metodologias ágeis à pesquisa; a aplicação, observação e análise das metodologias ágeis, de forma controlada, em projetos para medir seus impactos; e, até mesmo, o desenvolvimento de uma nova metodologia ágil, totalmente dedicada à gestão de projetos de desenvolvimento de jogos de pequeno escopo.

REFERÊNCIAS

AMARAL, D. C.; CONFORTO, E. C.; BENASSI, J. L. G.; ARAUJO, C. d. Gerenciamento ágil de projetos: aplicação em produtos inovadores. **São Paulo: Saraiva**, p. 240, 2011.

BECK, K. Embracing change with extreme programming. **Computer**, IEEE, v. 32, n. 10, p. 70–77, 1999.

BECK, K.; BEEDLE, M.; BENNEKUM, A. V.; COCKBURN, A.; CUNNINGHAM, W.; FOWLER, M.; GRENNING, J.; HIGHSMITH, J.; HUNT, A.; JEFFRIES, R. *et al.* **The agile manifesto**. [S.l.]: Feb, 2001.

BOEG, J. Kanban em 10 passos. **Tradução de Leonardo Campos, Marcelo Costa, Lúcio Camilo, Rafael Buzon, Paulo Rebelo, Eric Fer, Ivo La Puma, Leonardo Galvão, Thiago Vespa, Manoel Pimentel e Daniel Wildt. C4Media**, 2010.

CHANDLER, H. M. **Manual de produção de jogos digitais**. 2ª edição. ed. [S.l.]: Bookman Editora, 2012.

CHANDLER, H. M. **The Game Production Toolbox**. [S.l.]: CRC Press, 2020.

DIAS, D. d. S.; SILVA, M. F. d. Como escrever uma monografia. **Relatórios Coppead**, Universidade Federal do Rio de Janeiro, 2009.

FACCIOLI, R. V.; SILVA, A. C. da. Fuidd: Desenvolvimento dirigido à funcionalidade de interface de usuário. 2020.

FERREIRA, R. B.; LIMA, F. Metodologias ágeis: Um novo paradigma de desenvolvimento de software. In: **II Workshop Um Olhar Sociotécnico sobre a Engenharia de Software–WOSES**. [S.l.: s.n.], 2006.

GIL, A. C. Como elaborar projetos de pesquisa. **Como elaborar projetos de pesquisa**, São Paulo: Atlas, v. 5, 2010.

GODOY, A.; BARBOSA, E. F. Game-scrum: An approach to agile game development. **Proceedings of SBGames**, p. 292–295, 2010.

GUIMARÃES, L. F. d. A.; FALSARELLA, O. M. Uma análise da metodologia just-in-time e do sistema kanban de produção sob o enfoque da ciência da informação. **Perspectivas em Ciência da Informação**, SciELO Brasil, v. 13, n. 2, p. 130–147, 2008.

HIGHSMITH, J. **Agile software development ecosystems**. [S.l.]: Addison-Wesley Professional, 2002.

HUIZINGA, J. **Homo Ludens—o jogo como elemento da cultura (Homo Ludens—The Cultural Influence)**. [S.l.]: São Paulo: Perspectiva.(in Portuguese), 1938.

KEOGH, B. Between triple-a, indie, casual, and diy. **The Routledge companion to the cultural industries**, p. 152–162, 2015.

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

LAUBISCH, A.; CLUA, E. Scrum4games: Uma aplicação do scrum para projetos de games focada em game design. **Simpósio Brasileiro de Jogos e Entretenimento Digital**, p. 178–187, 2010.

MENCHER, M. **So You Want To Be A Producer**. 2006. Disponível em: <https://www.gamasutra.com/view/feature/131158/so_you_want_to_be_a_producer.php>.

NAKANO, D. N.; NAKAMURA, R.; SAKUDA, L. O. Produção e operações em games: Visão geral e perspectivas. **XI SBGames–Brasília–DF–Brazil**, 2012.

POSSI, M. *et al.* **Gerenciamento de Projetos Guia do Profissional Vol. 1: Abordagem Geral e Definição de Escopo**. [S.l.]: Brasport, 2006.

POSVOLSKI, A.; TORRES, I.; CONCILIO, I.; PACHECO, B. Agigame: Proposta de uma metodologia híbrida para desenvolvimento de jogos. **Anais do XIII Simpósio Brasileiro de Jogos e Entretenimento Digital**, 2014.

PRODANOV, C. C.; FREITAS, E. C. de. **Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico-2ª Edição**. [S.l.]: Editora Feevale, 2013.

RATIONAL SOFTWARE CORPORATION. **Rational Unified Process**. 2001. Disponível em: <https://sceweb.uhcl.edu/helm/RationalUnifiedProcess/>. Acesso em: 20 jun 2021.

SALEN, K.; ZIMMERMAN, E. Rules of play: Fundamentals of game design. **MIT Press Cambridge**, 2003.

SALES, R. Proposta de método para gestão ágil da visão no desenvolvimento de jogos digitais. **Anais do XII Simpósio Brasileiro de Jogos e Entretenimento Digital**, 2013.

SANTOS, C. F. R. **Gerenciamento de Projetos-Conceitos e Representações**. [S.l.]: Rio de Janeiro: LTC, 2014.

SCHWABER, K.; SUTHERLAND, J. La guía de scrum. **Scrumguides. Org**, v. 1, p. 21, 2013.

SCHWABER, K.; SUTHERLAND, J. **O Guia do Scrum**. 2020. Disponível em: <https://scrumguides.org/scrum-guide.html>. Acesso em: 17 mai 2020.

SILVA, J. B. da; ANASTÁCIO, F. A. de M. Método kanban como ferramenta de controle de gestão. **ID on line REVISTA DE PSICOLOGIA**, v. 13, n. 43, p. 1018–1027, 2019.

SOARES, M. dos S. Metodologias ágeis extreme programming e scrum para o desenvolvimento de software. **Revista Eletrônica de Sistemas de Informação**, v. 3, n. 1, 2004.

TAMAKI, P. A. O. **Uma extensão do RUP com ênfase no gerenciamento de projetos do PMBoK baseada em process patterns.** Tese (Doutorado) — Universidade de São Paulo, 2007.

VIEIRA, L. Games independentes: plataformas de inovação das novas mídias no contexto da economia criativa. Universidade Estadual Paulista (UNESP), 2015.