Análise Comparativa entre Metodologias de Desenvolvimento de *Games*

Luiz Felipe de Lima Mitterofhe

JUIZ DE FORA

DEZEMBRO, 2016

Análise Comparativa entre Metodologias de Desenvolvimento de *Games*

Luiz Felipe de Lima Mitterofhe

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Sandro Roberto Fernandes

Coorientador: Marco Antônio Pereira Araújo

JUIZ DE FORA

DEZEMBRO, 2016

Análise Comparativa entre Metodologias de Desenvolvimento de *Games*

Luiz Felipe de Lima Mitterofhe

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Sandro Roberto Fernandes
D.Sc. em Modelagem Computacional - UERJ

Marco Antônio Pereira Araújo
D.Sc. em Engenharia de Sistemas e Computação - UFRJ

Igor de Oliveira Knop
D.Sc. em Modelagem Computacional - UFJF

Victor Ströele de Andrade Menezes
D.Sc. em Banco de Dados - UFRJ

JUIZ DE FORA

16 DE DEZEMBRO, 2016

Dedico esse trabalho ao meu avô, João Geraldo Mitterofhe (in memoriam) por ter despertado em mim a paixão pelos games.

Resumo

O desenvolvimento de *games* evoluiu muito desde seu início, na década de 70, até os dias atuais. Os *games* tornaram-se maiores, envolvendo na sua construção, profissionais de diferentes áreas de conhecimento. Como consequência, o processo de desenvolvimento de *games* tornou-se uma tarefa que possui um gerenciamento complexo, fazendo-se necessário a utilização de metodologias pertencentes a Engenharia de *Software*. Essas metodologias são utilizadas com o intuito de otimizar da melhor forma possível o processo da produção de *games*, aumentando sua qualidade e gerindo recursos de forma eficiente, sem desperdícios para não comprometer o projeto. Este trabalho apresenta de forma sistemática, metodologias utilizadas no desenvolvimento de *games* e como elas ajudam a otimizar esse processo. O objetivo deste trabalho é realizar comparações entre as metodologias encontradas, de forma a reunir e analisar suas vantagens e desvantagens na produção de *games*. Com os resultados obtidos das comparações entre as metodologias, é mostrado que para obter os melhores resultados na criação de um *game* é necessário que seu processo de desenvolvimento esteja intimamente ligado aos processos de desenvolvimento da Engenharia de *Software*.

Palavras-chave: Jogos Eletrônicos, Projeto de *Game*, Desenvolvimento de *Games*, Metodologias de Desenvolvimento, Engenharia de *Software*, Processo Ágil.

Abstract

The games development has evolved much since your beginning, in the decade of 70, to the present day. The games have become major, involving in its construction, professionals from different fields of knowledge. As a consequence, the process of games developing has become a task that has a complex management, making necessary the use of methodologies pertaining to Software Engineering. These methodologies are used in order to optimize of the best possible way the game production process, increasing their quality and managing resources efficiently without waste not to compromise the project. This work presents of a sistematic way to, methodologies used in the game development and how they help to optimize this process. The aim of this work is carry out comparisons between the methodologies, in order to gather and analyze its advantages and disadvantages in the production of games. With the results of the comparisons between the methodologies, it is shown that for the best results in the creation of a game it is necessary that your development process is closely linked to the processes of development of Software Engineering.

Keywords: Games, Game Design, Games Development, Development Methodologies, Software Engineering, Agile Process.

Agradecimentos

Agradeço a Deus, em primeiro lugar, por tudo. Principalmente, por estar sempre ao meu lado dando-me forças nos momentos difíceis da minha vida.

Agradeço aos meus pais, Adalberto e Marcia, e minha avó Zenilda (in memoriam), pelo amor, carinho, educação e dedicação que sempre me deram. Sem o apoio, incentivo e compreesão de vocês, jamais teria chegado nesse ponto da minha vida. Agradeço também ao meu irmão, Victor Hugo, pela amizade e companheirismo durante todos esses anos.

Agradeço aos meus orientadores, Sandro e Marco Antônio, pela oportunidade que me deram, em pesquisar sobre a área de desenvolvimento de *games*.

Agradeço aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e que durante esses anos, contribuíram de algum modo para o meu enriquecimento pessoal e profissional.

Agradeço aos meus colegas de curso pela amizade e, de algum modo, por terem contribuído para que eu chegasse até aqui.

A todos, muito obrigado !

"No meu cartão de visitas, sou presidente de uma empresa. Na minha mente, sou um desenvolvedor de jogos. Em meu coração, sou um gamer."

Satoru Iwata, ex-presidente da Nintendo (in memoriam)

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

AGP	*Agile Game Process*
CAPES	Comissão de Aperfeiçoamento de Pessoal do Nível Superior
GAMA	*Game Agile Methods Applied*
GDD	*Game Design Document*
GUP	*Game Unified Process*
IA	Inteligência Artificial
QA	*Quality Assurance*
RGP	*Rapid Game Process*
SB *Games*	Simpósio Brasileiro de *Games*
TI	Tecnologia da Informação
UML	*Unified Modeling Language*
XGD	*Extreme Game Development*
XP	*Extreme Programming*

1 Introdução

Um jogo eletrônico (ou *game*, como será definido ao longo deste trabalho) é um sistema codificado e interativo capaz de interpretar ações de um ou vários jogadores, através de um controle. Atualmente, é um conjunto de várias mídias: áudio, cinematográfica e digital; todas compiladas em um ambiente onde funcionam em equilíbrio. O profissional responsável por criar todo o conteúdo do *game* (ideias, regras e a experiência do *game*), conceitualmente, é conhecido por *game designer*. Existem alguns conceitos importantes sobre a definição de *games* de acordo com famosos *game designers*. Entre eles destacam-se Scott Rogers, que define um *game* como uma atividade que requer no mínimo um jogador, possui regras e possui uma condição de vitória. E também Sid Meier, que define *games* como sendo uma série de escolhas significativas (Sato, 2010; Rogers, 2013).

1.1 Motivação

Games são um dos meios de entretenimento mais utilizados nos dias de hoje, tornando parte da cultura popular mundial, influenciando cada vez mais pessoas, ao longo de mais de 40 anos desde o surgimento do primeiro *Video Game*, o Magnavox Odyssey. É uma das formas de mídia mais completas que se tem atualmente, devido ao poder em não somente divertir mas também ensinar, simular e criar. Além disso, a indústria de *games* é uma das que mais crescem no mundo, ultrapassando a indústria do cinema e da música, isso pode ser comprovado pelo aumento de cursos e empresas especializadas em desenvolvimento de *games* (Ampatzoglou e Chatzigeorgiou, 2007; Valin, 2013; BNDSGediGames, 2014).

No entanto, com o crescimento da complexidade e tamanho dos *games*, não é mais possível um único programador criar sozinho toda a codificação, *design* e testar o *game*, como ocorria nos primórdios de sua produção. Essa complexidade se deve ao próprio mercado, onde os jogadores buscam sempre *games* maiores, mais bonitos, músicas de alta qualidade e mais imersivos. Sendo necessário muitas pessoas (e de diferentes áreas

de atuação) para desenvolver um *game* de forma profissional. Segundo Petrillo (2008) e Albino, Souza e Prado (2014), para melhor gerenciar esses profissionais e as etapas envolvidas no desenvolvimento de *games*, houve uma necessidade de utilizar metodologias para criar um processo de produção de forma eficiente. Além disso, a essência dos *games* para o entretenimento é a diversão. Uma boa metodologia de *software* tem a capacidade de auxiliar os desenvolvedores em refinar essa diversão, de modo a produzir um *game* de boa qualidade e que agrade a maior quantidade de pessoas possíveis.

1.2 Justificativa

A produção de *games* é um cenário diverso, devido a sua característica de multidisciplinaridade, envolvendo profissionais de diferentes áreas. As chances de ocorrerem problemas durante o desenvolvimento são grandes e como consequência, há a possibilidade de cancelamentos ou *games* lançados com erros ou ainda com uma qualidade inferior ao esperado. A fim de minimizar esses problemas típicos da produção de *games*, é necessário a utilização de metodologias para o seu desenvolvimento. E com isso, diminuindo o tempo e custo de produção, otimizando da melhor forma possível o desenvolvimento, e aumentando a qualidade do *game* com o mínimo de erros possíveis.

1.3 Objetivos

1.3.1 Objetivo geral

Analisar as metodologias existentes para o desenvolvimento de *games*, buscando seus pontos fortes e fracos, e comparando-as. A fim de encontrar as técnicas mais indicadas ao desenvolvimento de *games*.

1.3.2 Objetivos específicos

1. Verificar quais as metodologias existentes para o desenvolvimento de *games*.
2. Analisar os pontos fortes e fracos de cada metodologia encontrada.

3. Verificar como a Engenharia de *Software* pode auxiliar o processo de desenvolvimento de *games*.

4. Comparar as metodologias encontradas na pesquisa.

5. Identificar as metodologias mais completas para o desenvolvimento de *games*.

1.4 Metodologia de Pesquisa

A presente monografia é um projeto de pesquisa científica do tipo bibliográfica e exploratória. A pesquisa foi realizada através de uma Revisão Sistemática, com base em questões que pudessem atingir os objetivos propostos neste trabalho. Além disso, foi construído o referencial teórico com a finalidade de conhecer melhor a indústria de *games*. E com isso, posteriormente, fazer a comparação entre as metodologias encontradas na pesquisa.

1.5 Estrutura do Trabalho

Este trabalho está estruturado da seguinte forma: No Capítulo 1, foi apresentado o conceito geral de *games* e, a motivação para o uso de metodologias apropriadas ao desenvolvimento de *games* para o entretenimento. Ou seja, metodologias que proporcionam à produção de *games*, o menor desperdício de tempo e custo, e máxima qualidade possível. No Capítulo 2, será exposto importantes informações sobre a indústria e etapas da produção de *games*. A partir de então, será destacado um elementar processo para o desenvolvimento de *games*, conhecido como metodologia ágil. No Capítulo 3, será apresentado uma Revisão Sistemática, com o intuito de realizar o levantamento das metodologias de produção de *games* existentes na literatura, quais as mais utilizadas e, a relação entre as áreas de Desenvolvimento de *Games* e Engenharia de *Software*. Particularmente, como a Engenharia de *Software* auxilia o processo de produção de *games*. No Capítulo 4, será apresentado com mais detalhes um estudo sobre as metodologias de desenvolvimento de *games*, encontradas na Revisão Sistemática, destacando-se a descrição de cada uma e, seus pontos altos e baixos. Em seguida, será enfatizado a comparação entre as metodologias,

em relação a critérios de comparação que representam boas práticas de desenvolvimento de *games*, com a finalidade de pesquisar os processos mais adaptados aos desafios impostos pela indústria. Ao final do capítulo, com base na análise comparativa entre as metodologias, será apresentado um modelo de referência, com o objetivo de encontrar as metodologias mais apropriadas em relação a complexidade da produção de um *game*.

2 Referencial Teórico

Segundo Lemes (2009), uma das definições mais difundidas sobre *games* afirma que, estes são jogos eletrônicos onde há uma interação entre imagens (enviadas e exibidas por um monitor) e o jogador. Contudo, *games* não são meramente uma interação ente imagens. Para Cruz e Garone (2013), *games* são um conjunto de ações e decisões possíveis de serem realizadas por um ou mais jogadores, limitados por regras e pela ambientação do *game*, com um propósito de diversão. Já em Araujo (2006) e Brathwaite e Schreiber (2009), é apresentado uma definição mais tecnológica sobre o que é um *game*: Uma atividade estruturada onde os jogadores devem tomar decisões e alcançar determinados objetivos, envolvendo estímulos físicos e/ou mentais. É importante destacar que, em geral, os *games* possuem propósitos de entretenimento porém, podem ser utilizados com outros fins, como educação, treinamento, publicidade, entre outros. Todavia, neste trabalho de monografia, serão tratados apenas os *games* com caráter de entretenimento.

2.1 Etapas do Processo de Desenvolvimento de *Games*

Segundo Barros (2007); Chagas (2009); Lavor (2009); Lemes (2009); Santos e Correia (2009); Chandler (2012) e Posvolski et al. (2014) , um *game* é composto por diversos elementos, tais como histórias, regras, interface com os jogadores, efeitos de som, música, elementos e efeitos visuais, cenários, personagens, animações, entre outros. Sua criação requer método, disciplina, ideias, roteiro, recursos tecnológicos, *softwares* específicos e a capacidade de transformar a experiência interativa do ato de jogar a mais divertida possível. Dessa forma, como qualquer outro *software*, um *game* necessita ser desenvolvido através de etapas dentro de um processo de produção, até estar devidamente preparado para ser lançado no mercado. Assim sendo, independente do tamanho da equipe, do escopo do *game*, do orçamento e do estúdio, em geral, existe uma estrutura básica para o

processo de produção de *games*. Esse processo, pode ser dividido em quatro fases principais: Concepção, Pré-Produção, Produção e Pós-Produção. Nas subseções seguintes, será apresentada um resumo de cada uma dessas etapas. Além disso, será apresentado um resumo de dois processos de gerenciamento de projetos, muito importantes para a produção de *games*, os gerenciamentos de qualidade e de riscos.

2.1.1 Fase de Concepção

O desenvolvimento de um *game* inicia-se com a criação do conceito geral (ou *game concept*). Esse conceito geral é a ideia principal que servirá como base pelo qual irá ser construído o *game*. Se o conceito for fraco ou não for totalmente definido antes do início da próxima etapa de produção, elementos importantes podem ficar ausentes e só serem descobertos quando a equipe estiver em um momento mais avançado do desenvolvimento (Araujo, 2006; Chandler, 2012).

Para Luz (2004); Petrillo (2008); Nakano, Nakamura e Sakuda (2012); Carvalho (2013) e Posvolski et al. (2014), a fase de Concepção é a atividade de idealização do *game*, sua proposta, na qual as pessoas envolvidas no desenvolvimento do projeto, definem como este deverá ser, ou seja, o seu perfil. A fase de Concepção pode ser considerada como uma primeira parte, do importante processo do desenvolvimento de *games*, conhecido como *Game Design*. Neste processo, pertencente a fase de Pré-Produção, de uma forma geral, será detalhado e aperfeiçoado a ideia gerada na fase de Concepção.

Esta fase inicia-se a partir de reuniões em que ideias são apresentadas e discutidas. Essas reuniões devem abranger temas mais amplos do que apenas a ideia central do *game*. Temas como originalidade, inovação, público-alvo, plataforma, gênero, possibilidades de mercado, objetivos do projeto e algumas funcionalidades iniciais; devem ser discutidas. Além disso, nessas reuniões devem ser incluídas também, questões do ponto de vista gerencial, tais como: A elaboração das primeiras estimativas de custos, recursos humanos, cronograma inicial do projeto, entre outros. Após tudo isso ser definido, qualquer pessoa envolvida no projeto, deve ser capaz de entender os objetivos do *game* (Chandler, 2012; Nakano, Nakamura e Sakuda, 2012; Carnasciali, 2014; Freitas, 2014).

Para Luz (2004); Lavor (2009) e Chandler (2012), quando o conceito geral estiver

suficientemente amadurecido e pronto para gerar um *game*, será necessário apresentá-lo ao *publisher*[1]. Esta apresentação dará a eles a chance de ver como a equipe planeja criar um *game* sólido a partir do conceito inicial. Se o *publisher* gostar da ideia e achar que ela é viável economicamente, aprovará o projeto. A partir de então, a equipe de desenvolvimento será reunida. Dessa forma, o projeto estará pronto para a próxima fase, a Pré-Produção.

2.1.2 Fase de Pré-Produção

A Pré-Produção é a etapa de planejamento do projeto, ou seja, é onde as informações serão reunidas mostrando como tudo será produzido. Decisões que afetarão todo o desenvolvimento do projeto, inclusive se o *game* fará sucesso ou não, serão tomadas nesse momento. Nesta etapa, as equipes de desenvolvimento são alocadas. São definidos também, as responsabilidades de cada profissional envolvido no projeto. É na Pré-Produção, que é criado o principal artefato necessário para o desenvolvimento do *game*, o GDD[2] (Barros, 2007; Slyke, 2009; Chandler, 2012).

Para Taylor et al. (2007); Petrillo (2008); Slyke (2009); Godoy e Barbosa (2010); Carvalho (2013) e Posvolski et al. (2014), a etapa da Pré-Produção de um *game* é um processo complexo, pois envolve a criação de uma experiência para o jogador. É nessa fase que serão definidos os elementos fundamentais do projeto, incluindo a definição das principais *features*[3] do *game*, o roteiro, a arte, músicas e efeito sonoros (somente o planejamento), plataformas, a linguagem de programação usada, utilização ou não de protótipos, e o cronograma e orçamento final do projeto. Além disso, nesta fase, o conceito do *game* será detalhado e melhorado.

Para Lavor (2009); Slyke (2009); Keith (2010) e Carvalho (2013), os objetivos da Pré-Produção são: Permitir que a equipe de desenvolvimento planeje cada detalhe do

[1] É o investidor, editor e publicador do *game* no mercado, geralmente.

[2] Documentação de *design*, fundamental para qualquer projeto de *games*. O GDD tem a função de auxiliar a equipe de desenvolvimento durante a produção, e deve esboçar tudo o que estará no *game* até o final do projeto. Além disso, o GDD tem o princípio de auxiliar na comunicação das pessoas envolvidas no projeto, guiando-as e mantendo toda a equipe alinhada na busca pelos mesmos objetivos (Petrillo, 2008; Lemes, 2009; Chandler, 2012; Rogers, 2013).

[3] Termo comum da indústria, que pode ser traduzido como qualquer característica ou funcionalidade do *game*. Essa funcionalidade pode ser uma mecânica, recursos artísticos, sonoros, elementos de IA, entre outros.

projeto, inclusive o cronograma, com base nas estimativas de tempo fornecidas por cada equipe de desenvolvimento, e principalmente, planejar o entretenimento do *game*, ou seja, o seu *gameplay*[4]. Portanto, a fase de Pré-Produção, sendo a etapa de planejamento do *game*, é importante para que todos os objetivos do projeto sejam alcançados ao longo de seu ciclo de desenvolvimento. Assim, quanto mais tempo for investido nela, maior será a capacidade de ter uma visão completa do *game*, o que permite a antecipação e a solução dos problemas, que irão ocorrer, de maneira mais eficiente. Após esta etapa, o produtor[5] e a equipe terão uma ideia clara do que esperar durante a fase de Produção, pois se tudo foi planejado adequadamente, toda a equipe estará pronta para a próxima etapa de desenvolvimento.

2.1.2.1 *Game Design*

"*Game Design* é o processo de imaginar um *game*" (Luz, 2004, p. 14). É criar disputas, conteúdos e suas regras. É o processo a partir do qual, são descritos as características principais do *game*, como os detalhes da jogabilidade[6], as escolhas que o jogador terá dentro do *game*, assim como as ramificações que suas escolhas vão originar, quais as condições de vitórias e derrotas, como o *game* será controlado e as informações que o jogador deverá receber (sua interface), detalhes dos personagens, comportamento dos inimigos, as características sonoras e visuais do *game*, detalhes das fases, entre outros elementos que fazem parte da experiência gerada pelo *game*. A construção e evolução do *Game Design* deve ser realizada sempre de uma forma dinâmica pois, para a produção de um *game* de boa qualidade será necessário alterar, sempre que necessário, seus elementos de *design*. Por exemplo, as mecânicas do *game*. Assim sendo, o processo de *Game Design* pode durar até o final do projeto e não apenas na fase de Pré-Produção. É o processo mais importante do desenvolvimento de um *game*. Sendo que, ele por si só, pode garantir

[4]É a experiência que o *game* proporciona ao jogador. É tudo o que acontece entre o início e o final de um *game* (Carmona, 2012).

[5]Análogo ao gerente de projetos na Engenharia de *Software*.

[6]É o conjunto das mecânicas[7] de um *game*. A jogabilidade está diretamente ligada ao modo de jogar e influencia nas reações do jogador. Todas as possibilidades e formas de interação do *game*, são determinadas através da jogabilidade (Lemes, 2009; Carmona, 2012; McEntee, 2012).

[7]É tudo o que o jogador pode fazer no *game*, como suas ações, habilidades, possibilidades de decisões, entre outros. As regras do *game* e as variedades de respostas que este dará ao jogador (seu sistema de *feedback*), também são considerados mecânicas (Lemes, 2009; Sato, 2010; Stateri, 2013).

o sucesso ou o fracasso de um *game* no mercado. (Rouse, 2005; Araujo, 2006; Santana, 2006; Brathwaite e Schreiber, 2009; Lavor, 2009; Lemes, 2009; Sato, 2010).

2.1.2.2 Prototipação

Antes do projeto passar para a fase de Produção, é necessário testar os conceitos, mecânicas, a experiência proporcionada ao jogador, entre outros. Assim, se algo não ocorrer como o esperado, não irá comprometer o projeto, pois este ainda estará na fase de Pré-Produção. Esses testes são realizados através de protótipos, muitas vezes chamados de "versão demo". Protótipos são modelos de interfaces simplificadas, com apenas alguns recursos implementados para testes rápidos, ou testes de funcionalidades críticas, que comprometeriam o projeto, caso fossem inseridos em uma versão estável do *game*. Dessa forma, a essência do uso de protótipos está em não implementar o *game* até que o planejamento do *gameplay* esteja bem resolvido, testado e validado, e com isso, diminuindo o tempo e custos do projeto. É importante destacar que, a prototipagem é um método de concepção comumente utilizado na produção de *games* (Ollila, Suomela e Holopalnen, 2008; Sato, 2010; Cruz e Garone, 2013; Medeiros et al., 2013).

Para Araujo (2006); Ollila, Suomela e Holopalnen (2008); Sato (2010); Medeiros et al. (2013) e Carnasciali (2014), a prototipação funciona com um canal de comunicação eficiente, de forma a transmitir as ideias do projeto a todos os membros da equipe. Este tipo de comunicação, também permite resgatar dados que seriam pouco perceptíveis através da maioria dos métodos de teste, como a verificação da experiência que o *game* está proporcionando, através do *Playtest*[8]. É importante ressaltar que, o uso de protótipos não é necessariamente exclusivo da etapa de Pré-Produção, podendo estar presente ao longo de todo o desenvolvimento do *game*.

2.1.3 Fase de Produção

Se tudo for planejado adequadamente e aprovado pelo produtor, e pelo *publisher*, durante a etapa de Pré-Produção, são grandes as possibilidades da fase de Produção ser bem sucedida. Ou seja, o *game* será criado dentro do prazo e sem estourar os custos

[8]Mais detalhes sobre *Playtests*, serão vistos na seção 2.1.5.

do projeto. A etapa de Produção começa após o conceito e os requisitos do *game* serem definidos nas fases anteriores. O objetivo desta etapa é executar o planejamento do *game*, criado durante a Pré-Produção (Keith, 2010; Chandler, 2012; Medeiros et al., 2013; Albino, Souza e Prado, 2014; Posvolski et al., 2014).

Para Santana (2006); Petrillo (2008); Godoy e Barbosa (2010); Sato (2010); Chandler (2012); Carvalho (2013) e Albino, Souza e Prado (2014), esta fase deve ser cuidadosamente executada e gerenciada, através de processos capazes de prever e lidar com possíveis falhas ou alterações no *game*. Uma das atividades desta etapa, é a conversão de conceitos, mecânicas e outras *features* documentadas no GDD, em uma lista de pendências a serem implementadas. Outra importante atividade desta etapa é a criação da arquitetura do *game* e sua divisão em módulos, de forma a otimizar o processo de produção, diminuindo custo e tempo. E com isso, no final do processo, esses módulos são integrados de forma coesa e eficiente. É importante ressaltar que, na etapa de Produção, muitos elementos devem ser gerenciados, principalmente aqueles relacionados a arte, *design*, programação e testes. As equipes devem ser guiadas pelo produtor, pelo cronograma e pela documentação gerada na Pré-Produção. Dessa forma, a fase de Produção, é o ponto em que o projeto começa a se parecer com um *game*.

Segundo Santana (2006) e Chandler (2012), outra importante função da etapa de Produção está relacionada a garantia de qualidade. Nesta fase, as equipes de programação e QA devem trabalhar juntas para a rápida identificação e correções de eventuais *bugs*[9], de forma a não permitir o acúmulo de erros e, muito menos, a identificação tardia de *bugs* críticos, que possam inviabilizar o projeto. É importante destacar que, neste momento do projeto, uma versão executável e estável do *game* deve estar em funcionamento durante toda a etapa. Com o passar do tempo, serão geradas várias versões do *game*. Cada versão é resultado de refinamentos das versões anteriores. Esses refinamentos são, em geral, correções de *bugs*, melhorias gráficas, melhorias no áudio, novas *features* (se houver necessidade), entre outros. Esse processo ocorre até que se tenha a versão final do projeto, que na verdade, é o *game* propriamente dito. Quando este chega a sua versão final, é realizado os últimos ajustes. Após isso, o *game* estará pronto para ser lançado no

[9]Erros no *design* do *game*, código, arte, som ou escrita (Brathwaite e Schreiber, 2009).

mercado.

2.1.4 Fase de Pós-Produção

Após o código do *game* ter sido aprovado para lançamento, algumas atividades devem ser realizadas antes do encerramento oficial da produção do *game*. Uma das principais atividades desta etapa é a geração de um plano de arquivamento do código e *assets*[10] do projeto. Esse arquivamento é feito através de um kit de fechamento. Um kit de fechamento deve conter toda a documentação do projeto, código-fonte, *assets* artísticos, e tudo mais que foi usado e criado para desenvolver o *game*. Os kits de fechamento são necessários para a reutilização do código e *assets* em outros projetos. Como por exemplo, na sequência de um *game* (Chandler, 2012; Carvalho, 2013; Posvolski et al., 2014).

Além de atividades de *marketing* e publicidade, uma das principais tarefas desta etapa, é a construção de um documento que possui a finalidade de aprendizagem com a experiência do projeto atual. Este documento, conhecido como *Post-mortem*, possui a finalidade de registrar tudo o que ocorreu de bom e ruim durante a produção do *game*, com o intuito de evoluir o processo de produção da empresa. Assim sendo, quando o *Post-mortem* for desenvolvido e o kit de fechamento estiver finalizado e devidamente testado, o projeto do *game* está oficialmente encerrado (Godoy e Barbosa, 2010; Chandler, 2012).

2.1.5 Garantia de Qualidade

Segundo Chandler (2012), a principal motivação do processo de QA é criar um plano de testes para o *game* e validá-lo em relação a esse plano. O plano de testes é desenvolvido de acordo com os *assets* e *features* do projeto. Dessa forma, todos os elementos passados à equipe de QA devem estar atualizados para a criação de planos de testes apropriados e também atualizados, de modo a refletir as últimas alterações no *game* e no GDD. O produtor e os líderes devem trabalhar juntos ao departamento de QA para tornar disponíveis todas as informações necessárias à criação de planos de testes precisos.

[10]São os ativos do *game*, ou seja, todo e qualquer elemento utilizado no projeto e que será, de alguma forma, apresentado para o jogador, como: Arquivos gráficos 2D, modelos 3D, efeitos sonoros, músicas, textos e diálogos, mapas, entre outros. Portanto, é tudo que contribui para a aparência visual do *game* (Davis, 2009).

O processo de testes é crucial no desenvolvimento de *games*. É quando a equipe de QA verifica se tudo funciona corretamente, e se não há nenhum *bug* fatal, comprometendo o lançamento do *game*. Os testes são contínuos durante a fase de Produção, já que o departamento de QA verificará as *builds*[11], e novas funcionalidades à medida que ficarem disponíveis. Após a versão beta do *game* ser finalizada, a equipe de desenvolvimento dedicará, principalmente, às correções de *bugs* e a criação de novas *builds* para serem testadas pela equipe de QA. Os testes são parte integrante do desenvolvimento de *games*. Dessa forma, os testadores de QA são fundamentais em todo o ciclo de produção e, seu líder deve ser incluído em decisões de todos os níveis sobre o projeto (Chandler, 2012; Nakano, Nakamura e Sakuda, 2012).

De acordo com Araujo (2006); Santana (2006); Petrillo (2008); Brathwaite e Schreiber (2009); Brauwers (2011) e Chandler (2012), durante a etapa de Produção, a equipe de programação libera versões jogáveis do *game*. Contudo, dentre essas versões, três se destacam, sendo importantes para o projeto. Essas *builds* são denominadas Alpha, Beta e *Gold Master*. A versão Alpha é a primeira versão estável e, completamente jogável do *game*. Onde os principais elementos de jogabilidade, narrativas e *assets* já estão implementados (cerca de 40% a 50% do *game* concluído). No entanto, nem todo o conteúdo está incluso. Sendo que, as *features* já implementadas, ainda podem possuir *bugs* e, em geral, o *game* ainda se encontra desbalanceado. Já a versão Beta, apresenta todos os níveis, *assets*, recursos sonoros e interatividade do *game* implementados. Mas ainda podem conter *bugs*, inclusive *crash bugs*[12]. Desssa forma, a versão Beta é significativamente mais estável do que a versão Alfa, com menos *bugs* e um sistema mais balanceado. E por fim, a versão *Gold Master* é a última *build* do *game*. Nesse momento, o *game* está com a maioria dos principais *bugs* corrigidos, recursos áudio-visuais polidos, mecânicas refinadas, e o sistema balanceado. Ou seja, o *game* está pronto para ser lançado no mercado.

É importante destacar também, um tipo especial de teste de *software*, específico da indústria de *games*, conhecido como *playtest*, ou teste de jogabilidade. O *playtest* é realizado sobre uma versão, geralmente Beta, do *game*. Neste tipo de teste, potenciais

[11] Versões compiladas e executáveis do *game*.

[12] Tipo de erro que impede o jogador de progredir no *game*. Os *crash bugs* são graves pois, podem travar o *game* ou, gerar mensagens de erros durante o mesmo (Chandler, 2012).

jogadores deverão jogar o *game* em questão, com a finalidade de que a equipe possa avaliar se a experiência projetada para o *game* foi alcançada. É no *playtest* que os desenvolvedores analisam a aceitação do *game* e a reação dos jogadores. Dessa forma, as sugestões e críticas desses usuários podem ser encaminhadas diretamente aos desenvolvedores, melhorando ainda mais a qualidade do *game* (Reis, Nassu e Jonack, 2002; Araujo, 2006; Petrillo, 2008; Medeiros et al., 2013).

2.1.6 Gerenciamento de Riscos

Pode-se definir riscos como tudo aquilo que pode dar errado em um projeto, causando impactos financeiros a empresa desenvolvedora. Por exemplo: Um membro-chave da equipe sair no meio da produção, um fornecedor externo não cumprir com sua data final de entrega, identificação de um *crash bug* na versão *Gold Master*, entre outros. O gerenciamento de riscos é fundamental em qualquer tipo de projeto. Dessa forma, esse processo deve ser contínuo, ou seja, executado durante todo o desenvolvimento do *game*. É também, de suma importância que o produtor saiba quais são os riscos para o projeto, antes mesmo deste ser iniciado, ainda na fase de Pré-Produção. Além disso, uma forma eficaz de fazer esta avaliação inicial, é envolver todos os *stakeholders*[13] do projeto, preparando toda a equipe para eventuais ocorrências dos riscos já identificados. Após a identificação, é necessário priorizá-los para uma posterior estratégia de mitigação, pois os impactos causados ao projeto e a probabilidade de ocorrerem, varia de risco para risco. É importante destacar que, quando há um bom planejamento e gerenciamento de riscos, os membros da equipe de desenvolvimento podem se dedicar à conclusão do *game* de forma mais tranquila, porque os riscos já foram previstos. O gerenciamento de riscos, para o desenvolvimento de *games*, é dividido em duas etapas: avaliação e análise dos riscos, e controle dos riscos. A Tabela 2.1, apresenta os procedimentos básicos que o gerente do projeto deve realizar durante estas etapas (Chandler, 2012).

[13]É qualquer pessoa ou organização que tenha interesse, ou seja afetado pelo projeto. Como por exemplo: O gerente do projeto, o analista de sistema, o programador, o investidor, os usuários, entre outros.

Etapa 1 - Avaliação e Análise de Riscos
Identificar quais os riscos que possam afetar o projeto.
Analisar a probabilidade de cada risco ocorrer e o impacto que ele terá sobre o projeto.
Priorizar (através de uma escala) cada risco, começando com os de maior impacto.
Etapa 2 - Controle de Riscos
Criar um plano de gerenciamento que neutralize ou remova os riscos do projeto.
Implementar os planos propostos para eliminar os riscos.
Monitorar a evolução do projeto, de modo a não permitir qualquer possibilidade de ocorrência dos riscos identificados. Principalmente os riscos de maior impacto ao projeto.

Tabela 2.1: Procedimentos para avaliação e controle de possíveis riscos que possam ocorrer durante o projeto.

Durante todo o projeto, qualquer novo risco identificado, deve ser priorizado e controlado imediatamente, através dos procedimentos de avaliação, análise e controle de riscos. Além disso, uma importante atividade a ser realizada é a classificação dos riscos, de acordo com seus respectivos impactos. Em projetos de *games*, geralmente, essa classificação varia de 1 a 4. Somado a isso, riscos que possuem classificação 1 ou 2 devem ser acompanhados com atenção especial, devido ao impacto maior no projeto, caso ocorram (Chandler, 2012).

2.2 Características da Indústria de *Games*

O processo de desenvolvimento de *games* sempre esteve cercado de elementos particulares que o tornaram mais complexo do que a produção tradicional de *softwares*. Projetos com mais de 100 colaboradores, com custos superiores a dezenas de milhões de dólares são comuns. Muitos desses projetos ultrapassam o orçamento e/ou não conseguem permanecer no cronograma. Aliado a isso, à necessidade dos desenvolvedores em criar, a todo novo *game*, uma experiência diferente para os jogadores. Consequentemente, a produção de *games* é cercada de muitos riscos. É por esta razão que a utilização de me-

todologias de desenvolvimento de *games*, um bom planejamento e boa gestão, tendem a diminuir a grande quantidade de riscos, comuns da indústria (Petrillo, 2008; Keith, 2010; Sales, 2013). Nas subseções seguintes, serão apresentados alguns dos desafios mais relevantes para a indústria de *games* e, frequentemente relatados em *Post-mortems* disponíveis pelas empresas.

2.2.1 Multidisciplinaridade

Os *games* tornaram-se complexos, não sendo mais possível um pequeno número de pessoas desenvolvê-los. A grande quantidade de profissionais envolvidos no processo, aliada a necessidade da separação de várias áreas dentro do desenvolvimento de um *game* (para atender aos anseios cada vez maiores dos jogadores), deram origem a uma das principais características da indústria, a multidisciplinaridade. Todas as áreas envolvidas na criação de *games* (artes, gerenciamento, modelagem, programação, sonorização, QA, *marketing*, entre outros) carecem de profissionais altamente especializados. Essa especialização tornou-se um padrão para a criação profissional de *games* (Taylor et al., 2007; Petrillo, 2008; Ampatzoglou e Stamelos, 2010; Carnasciali, 2014). Contudo, essa multidisciplinaridade também trouxe problemas antes inexistentes à indústria, relacionados a gestão de projetos de *games*. De acordo com Godoy e Barbosa (2010); Silva (2010); Carvalho (2013) e Sales (2013), a multidisciplinaridade aliada a grande quantidade de profissionais necessários na criação de *games* cada vez maiores, geram um grande desafio para a gestão desse tipo de projeto, onde qualquer erro de planejamento pode ser fatal, e consequentemente, a perda de grande quantidade de capital investido no *game*.

2.2.2 Comunicação

Nos *Post-mortems* disponíveis pelas empresas, um dos principais problemas da indústria de *games* estão relacionados a falhas na comunicação, principalmente por causa da multidisciplinaridade existente na indústria. Isso se deve pois, profissionais de uma determinada área tem dificuldades de explicar questões técnicas para profissionais de outras áreas. Essa quebra na comunicação gera sérios problemas para o projeto, inclusive, a ocorrência de cancelamentos, devido a falhas no entendimento e implementação dos

requisitos. Falhas na comunicação também acarretam na produção de *games* com uma péssima qualidade ou ainda, bem diferentes do que foram inicialmente projetados. É de responsabilidade do produtor, ser o elo entre todos os profissionais envolvidas no projeto, gerenciando a comunicação da equipe, de forma eficiente, para que o *game* seja finalizado da melhor forma possível e com uma boa qualidade (Petrillo, 2008; Godoy e Barbosa, 2010; Park, 2010; Carvalho, 2013; Sales, 2013).

2.2.3 Cronograma Otimista

Demandas extremas e a pressão do mercado, problemas típicos em projetos de *games*, fazem com que as empresas desenvolvedoras trabalhem com prazos curtos, sendo difícil criar um cronograma real para o projeto, gerando sempre atrasos e frustrando os jogadores (Ampatzoglou e Stamelos, 2010; Sales, 2013; Albino, Souza e Prado, 2014). De acordo com Petrillo (2008), na maioria dos *Post-mortems* abertos pelas empresas de *games*, são descritos o quanto os projetos são afetados por falhas no desenvolvimento dos cronogramas. Estimativas otimistas, levam os desenvolvedores a trabalharem sem se preocuparem com prazos nas fases iniciais do projeto. E no final, quando notam que o *game* não ficará pronto na data prevista, os prazos são prolongados e os custos do projeto aumentam, afetando inclusive a credibilidade do estúdio. Ou seja, o cronograma otimista é um problema gerencial, pois o produtor deve conhecer todas as áreas do desenvolvimento de *games* e, com isso, ser capaz de estimar prazos corretos de acordo com as tarefas estabelecidas para cada um dos membros da equipe. Dessa forma, o problema de cronogramas otimistas é um dos que mais afetam os projetos de desenvolvimento de *games*.

2.2.4 Desenvolvimento do GDD

O GDD é um documento de grande importância, se usado como uma ferramenta para auxiliar o produtor, na comunicação da visão geral do *game* e de suas *features*, a todos os envolvidos no projeto. O problema, frequentemente relatado nos *Post-mortems* abertos, é quando o produtor delega a função de comunicação totalmente ao GDD. Neste caso, este ficará tão detalhado, que se torna cansativo sua leitura. E com isso, a equipe começará a ter ideias ambíguas sobre a visão contida nesse documento. Com o passar do

tempo, os profissionais acabarão construindo outro *game*, diferente do concebido originalmente, pois o documento indicou a direção errada a ser tomada. Além disso, a visão geral do *game* pode mudar ao longo do tempo. Nesse caso, se as alterações estiverem apenas nos documentos, dificilmente elas chegarão a todos os envolvidos no projeto, comprometendo-o seriamente. Portanto, depender exclusivamente do GDD para compartilhar a visão do *game*, compromete negativamente o projeto pois, muitas das informações que o produtor inseriu no documento serão perdidas ao longo do processo de desenvolvimento. Conversas diárias, reuniões significativas e muito planejamento, são situações ideais para compartilhar a visão do *game* (Keith, 2010; Stateri, 2013).

2.2.5 *Feature Creep*

De acordo com Petrillo (2008); Keith (2010); Brauwers (2011); Chandler (2012) e Albino, Souza e Prado (2014), uma das maiores razões para falhas em projetos de *games* é desenvolver um escopo pouco definido, ou seja, sem objetivos fortemente estabelecidos. Com isso, a equipe não possui uma visão geral de como o *game* deverá estar ao final do seu desenvolvimento. Sem a referência da visão do *game*, a equipe começará um processo, sem limites, de inserção de novas funcionalidades, aumentando vertiginosamente o tamanho do projeto. Esta prática é conhecida na indústria de *games* como *Feature Creep*, ou funcionalidade estranha. Esse crescimento ou acréscimo desenfreado de recursos, que não estavam no escopo original do projeto, definidos nas etapas de Concepção e Pré-Produção, acarreta em quebra do cronograma, aumento de *bugs* para inserir o novo recurso, e na maioria dos casos, em um aumento substancial do custo do projeto. Quando novas *features* são incorporadas, também novos riscos são estabelecidos, pois o incremento de novas funcionalidades requer mais testes e, possíveis problemas de integração.

2.2.6 *Crunch Time*

Crunch Time é o termo comumente utilizado na indústria, para períodos de extrema sobrecarga de trabalho, como jornadas de 12 a 16 horas ininterruptas por dia. O *crunch time*, geralmente, ocorre nas semanas que antecedem a data final de lançamento do *game*, gerando uma grande deterioração do ambiente de trabalho. O *crunch time* é

um dos problemas mais frequentes da indústria de *games* (e em projetos de *softwares* em geral), sendo relatados pela maioria dos *Post-mortems* abertos (Petrillo, 2008; Godoy e Barbosa, 2010; Keith, 2010; Brauwers, 2011; Albino, Souza e Prado, 2014). O *crunch time* é mais uma das consequências de má gestão, e metodologias pobres e inapropriadas, utilizadas em projetos de *games*.

2.2.7 Outros Desafios

É importante destacar que, nos *Post-mortems* abertos pelas empresas, alguns problemas típicos da indústria de *games* são relatados porém, de uma forma menos frequente e menos relevantes, do que os desafios citados anteriormente. Esses problemas são: Inserção tardia de profissionais no projeto, escopo irreal ou ambicioso, criação de uma experiência para os jogadores, grande quantidades de *bugs* encontrados no final do projeto, testes realizados tardiamente, orçamento extrapolado e problemas com os *softwares* utilizados no desenvolvimento do *game* (Petrillo, 2008; Godoy e Barbosa, 2010; Brauwers, 2011; Sales, 2013; Albino, Souza e Prado, 2014).

2.3 Processos Ágeis

Com o passar do tempo, novas necessidades foram surgindo à indústria tradicional de *software*. Essas necessidades possuem relação com o aumento das exigências e expectativas dos clientes, principalmente no que tange a mudanças rápidas nas funcionalidades do sistema. O que é bom para o cliente hoje, pode não ser tão bom amanhã. Esse princípio fez com que surgisse uma nova forma de desenvolver *softwares*, para atender a um mercado influenciado por mudanças cada vez mais aceleradas. Tais mudanças ocorrem, em geral, pelo avanço cada vez mais rápido da tecnologia, pressões constantes por inovações, sistemas e riscos cada vez maiores, concorrência acirrada no mundo dos negócios, entre outros. Os métodos tradicionais de desenvolvimento de *softwares* são insuficientes para o sucesso desse novo tipo de projeto. Com isso, foram surgindo novas abordagens para atender a projetos com essa característica (flexibilidade a mudanças). Uma dessas abordagens é conhecida como processos ou metodologias ágeis (Petrillo, 2008;

Machado, 2009; Pressman, 2011; Albino, Souza e Prado, 2014).

2.3.1 Conceito

Para Petrillo (2008); Machado (2009); Godoy e Barbosa (2010); Sales (2013) e Albino, Souza e Prado (2014), o desenvolvimento ágil é um processo de *software* que promove adaptação, fortalecimento do trabalho em equipe, auto-organização, entregas rápidas e adoção de boas práticas, alinhando o desenvolvimento com as necessidades dos clientes. A proposta do desenvolvimento ágil é aumentar a capacidade de criar e responder a mudanças, reconhecendo que está nas pessoas o principal elemento para guiar um projeto ao sucesso. Um aspecto importante do desenvolvimento ágil é visualizar *softwares* como sistemas adaptativos. Esses sistemas devem ser descentralizados, nos quais indivíduos independentes utilizam formas de interação auto-organizáveis, guiados por um conjunto de regras simples. Além disso, é importante ressaltar que, desenvolver *softwares* de forma ágil exige um alto grau de disciplina e organização.

2.3.2 Características

As principais características das metodologias ágeis são; flexibilidade, colaboração e o desenvolvimento iterativo e incremental. Assim sendo, o desenvolvimento ágil, de uma forma dinâmica, combina ciclos iterativos curtos (de uma a quatro semanas) com as necessidades do cliente, ou seja, implementação dos requisitos de maior prioridade para o mesmo. Assim, no final de cada iteração, o cliente pode priorizar novamente as funcionalidades desejadas para o próximo ciclo (ou seja, na próxima iteração), descartando ou alterando funcionalidades originalmente planejadas ou ainda, adicionando novas. A abordagem iterativa ocorre ao longo de todo o desenvolvimento do projeto, minimizando riscos e sempre buscando respostas rápidas à mudanças que possam ocorrer. (Sommerville, 2007; Petrillo, 2008; Godoy e Barbosa, 2010; Keith, 2010; Mikoluk, 2013).

Segundo Sommerville (2007); Keith (2010) e Carnasciali (2014), na abordagem ágil, a implementação e testes são desenvolvidos em paralelo, no mesmo ciclo iterativo. O *software* não é disponibilizado para o cliente, em sua versão final, no término do desenvolvimento. O sistema é desenvolvido e entregue ao cliente de forma incremental, ou

seja, em partes onde são implementadas apenas as funcionalidades de maior prioridade e necessidade para o cliente, naquele momento. Ao final de cada ciclo iterativo, novas funcionalidades previamente planejadas e priorizadas são liberadas para o cliente, através de uma nova versão do *software*. Isso é feito até o término do projeto, quando todas as funcionalidades requeridas pelo cliente estão incluídas na última versão do sistema. É importante destacar que, o cliente participa do planejamento de todos os ciclos iterativos. Propondo novas funcionalidades, aumentando ou diminuindo prioridades, removendo requisitos, ou seja, ele avalia (junto ao gerente do projeto) cada incremento a ser implementado no sistema.

A principal característica de uma equipe ágil é sua capacidade de lidar com mudanças no projeto de forma eficiente. Para tanto, a abordagem ágil propõe que as equipes de desenvolvimento sejam pequenas, para potencializar a comunicação e cooperação entre os profissionais. A gestão de um projeto ágil, é focada menos na administração e controle rigoroso das atividades e mais na comunicação e motivação dos desenvolvedores. E com isso, equilibrando valores como regras, hierarquia e independência. Além disso, na abordagem ágil, todos os membros da equipe sabem o que devem fazer e o que os outros estão fazendo. Sabem também, quais os atuais impedimentos que estão atrapalhando o progresso do projeto, bem como, os objetivos do mesmo (Pressman, 2011; Carnasciali, 2014).

3 Revisão Sistemática da Literatura

Uma importante área dentro do desenvolvimento de *games* é o estudo de metodologias para a sua produção. Essa área, ainda é pouco explorada, devido a quantidade de empresas que desenvolvem *games* sem qualquer tipo de metodologia, o que demonstra uma falta de profissionalização da indústria. Outro fator a ser destacado é a estreita relação da área de desenvolvimento de *games* com a Engenharia de *Software*, já que mais de 70% da indústria de *games* utilizam metodologias da Engenharia de *Software* (BNDSGediGames, 2014).

De acordo com Leite (2006) e Galeote (2010), metodologia é uma forma de utilizar um conjunto organizado e coerente de regras ou boas práticas com o intuito de alcançar um objetivo. Sua utilização possui como finalidade, evitar ao máximo a subjetividade na execução de tarefas em um processo de produção. Toda metodologia deve fornecer um roteiro dinâmico e interativo para a construção de *softwares*, em especial *games*, sempre com o propósito de aumentar a qualidade do produto e seu processo de produção. Outra importante característica na utilização de metodologias a ser destacada, é sua utilização para a prevenção de falhas no desenvolvimento de *softwares*. Dentre essas falhas em projetos de sistemas de TI, ressaltam-se: Complexidade subestimada; arquitetura pouco definida; testes insuficientes; falhas na comunicação e inconsistências na definição, planejamento e execução do projeto.

A importância de utilizar uma metodologia na área de desenvolvimento de *software*, é justamente obter o máximo em excelência para o sistema desenvolvido, gerando um alto custo-benefício para quem o desenvolveu. As vantagens de usar uma boa metodologia são: Desenvolver um *software* de boa qualidade (satisfazendo todos os requisitos pré-estabelecidos) e de maneira eficiente, ter todo o processo documentado de forma clara e precisa, não extrapolar prazos e orçamentos, estar preparado para a ocorrência de riscos, e saber controlar mudanças sem comprometer o projeto (Gervazoni, 2005).

Assim sendo, com a falta de uma literatura clássica sobre metodologias para o desenvolvimento de *games* e com o intuito de conhecer as metodologias existentes, as

mais importantes, as mais utilizadas e quais os impactos de sua utilização no processo de produção de *games*; foi realizada uma Revisão Sistemática sobre metodologias para o desenvolvimento de *games* e como a Engenharia de *Software* pode auxiliar este processo de produção.

3.1 Escopo e Questões

O escopo para a aplicação da Revisão Sistemática, na presente monografia, realizou-se através da utilização de metodologias de desenvolvimento de *softwares* (tradicionais e ágeis) aplicadas ao desenvolvimento de *games*. É importante ressaltar que, a Revisão Sistemática dessa monografia teve como finalidade, obter informações que ajudaram a encontrar e analisar o maior número de trabalhos primários relevantes e reconhecidos na área de metodologias para o desenvolvimento de *games*. Além de auxiliar na busca das seguintes questões de pesquisa:

Q1.1) Quais as metodologias para o desenvolvimento de *games* existentes na literatura?
Q1.2) Quais as metodologias para o desenvolvimento de *games* são mais utilizadas?
Q2) Como a Engenharia de *Software* pode auxiliar o processo de desenvolvimento de *games*?

3.2 Especificação das Questões da Pesquisa

Para cada questão da pesquisa foi necessário analisar ainda outros itens relacionados ao escopo e especificidade. Segundo Kitchenham (2007), é recomendado considerar as questões de pesquisa a partir da estrutura PICOC. Segundo essa estrutura, os itens a serem considerados por questão da pesquisa, são apresentados na Tabela 3.1:

Q1.1 e Q1.2	
Intervenção	Trabalhos que apresentem ocorrências de metodologias sendo usadas para o desenvolvimento de *games*, e quais as mais utilizadas.
Controle	Artigos, periódicos, monografias, dissertações de mestrado e teses de dou-

	torado que abordem o tema "Quais as metodologias de desenvolvimento de *games* existentes e quais as mais utilizadas".
Efeito	Auxiliar no processo de desenvolvimento de *games*.
Problema	Como utilizar a metodologia para otimizar o processo de desenvolvimento de *games* e aumentar a qualidade destes.
Aplicação	Entender como uma metodologia, específica para a produção de *games*, pode aumentar a qualidade destes durante todo o seu processo de desenvolvimento.
Q2	
Intervenção	Trabalhos que apresentem ocorrências do uso de metodologias da Engenharia de *Software* aplicadas ao desenvolvimento de *games*.
Controle	Artigos, periódicos, monografias, dissertações de mestrado e teses de doutorado que abordem o tema "Como a Engenharia de *Software* pode otimizar o processo de desenvolvimento de *games*, aumentando a qualidade destes".
Efeito	Auxiliar no processo de desenvolvimento de *games*.
Problema	Como utilizar a metodologia para otimizar o processo de desenvolvimento de *games* e aumentar a qualidade destes.
Aplicação	Entender como a Engenharia de *Software*, através de uma metodologia, irá otimizar o processo de desenvolvimento de *games*.

Tabela 3.1: Itens para a estruturação das questões da pesquisa.

3.3 Critérios para a Seleção de Fontes Primárias

Os critérios utilizados para a realização da Revisão Sistemática para a presente monografia foram escolhidos de acordo com as questões de pesquisa Q1.1, Q1.2 e Q2, apresentadas anteriormente. Além disso, esses critérios foram avaliados de forma a obter um conjunto relevante de estudos primários, com o intuito de descobrir o Estado da Arte sobre metodologias de desenvolvimento de *games* e a utilização da Engenharia de

Software na produção eficaz e eficiente de *games*. A Tabela 3.2, expõe os critérios levados em consideração e que contribuíram da melhor forma possível para a construção desta monografia.

Critério	**Descrição**
Seleção de fontes	Foi fundamentada em bases de dados eletrônicas incluindo conferências, periódicos e artigos.
Palavras-chave	*games* / jogos eletrônicos / jogos digitais;
	game design;
	game development / desenvolvimento de *games*;
	techniques / técnicas;
	method / *methodology* / metodologias / métodos / metodológicos;
	software engineering / engenharia de *software*;
	agile / ágeis.
Idioma	Português, Inglês e Espanhol.
Métodos de busca	As fontes foram acessadas via *Web*. No contexto dessa revisão não foi considerada a busca manual.
Listagem de fontes	Portal Periodicos CAPES
Artigos	Teórico e estudos experimentais.
Critérios de inclusão e exclusão de artigos	O material estava disponível na *Web*;
	O material considerava estudos de desenvolvimento de *games*;
	O material estava disponível integralmente;
	O material continha no título e no resumo alguma relação com o tema da monografia;
	Materiais relacionados à Engenharia de *Software*, possuíam alguma aplicação na área de desenvolvimento de *games*;
	Materiais duplicados foram excluídos.

Tabela 3.2: Critérios para a Revisão Sistemática.

3.4 Construção da *String* de Busca

Para a criação da *string* de busca, foram selecionadas palavras-chave em português, de acordo com o tema da monografia e as questões da pesquisa. A partir das palavras-chave em português, elas foram traduzidas para o inglês e acrescentadas à *string* de busca para incrementar o valor da pesquisa. Como cada base de dados possui milhares de artigos (e nesse trabalho utilizou-se o Portal Periodicos CAPES que agrega 256 bases, sem contar revistas internacionais e periódicos), foi necessário restringir o escopo da pesquisa sobre metodologias de desenvolvimento de *games*. Essa restrição teve como base os já mencionados critérios de inclusão e exclusão e os operadores lógicos AND e OR (que foram utilizados juntamente com as palavras-chave). Na Tabela 3.3, são apresentadas as *strings* de busca para as questões Q1.1 e Q1.2, e Q2.

String para Q1.1 e Q1.2
(técnica* OR technique* OR metodol* OR método* OR method*) AND ((“game design” OR “game development”) OR (desenvolvimento (“jogo* eletrônico*” OR “jogo* digital*” OR game*)))
String para Q.2
(agile OR ágil OR ágeis OR “software engineering” OR “engenharia de software”) AND (técnica* OR technique OR metodol* OR método* OR method*) AND (“game design” OR “game development” OR “jogo* eletrônico*” OR “jogo* digita*” OR game*)

Tabela 3.3: *Strings* de busca utilizadas na pesquisa.

Para as questões Q1.1 e Q1.2, após a utilização de suas *strings* de busca, foram encontrados um total de 673 artigos. Para a questão Q2, após a utilização de sua string de busca, foram encontrados um total de 226 artigos. Na primeira filtragem, baseada nos critérios de inclusão, exclusão e na leitura do título e do resumo de cada artigo advindo da busca; foram selecionados 25 artigos para Q1.1 e Q1.2, e 21 artigos para Q2. Na segunda filtragem, baseada na leitura criteriosa de cada artigo advindo da primeira filtragem, foram selecionados 6 artigos para Q1.1 e Q1.2, e 4 artigos para Q2. Na Tabela 3.4, é apresentado um resumo da quantidade de estudos primários, extraídos pelas *strings* de busca, antes e depois das filtragens, segundo os critérios de inclusão e exclusão.

Filtragens	**Quantidade de Estudos Primários para Q1.1 e Q1.2**	**Quantidade de Estudos Primários para Q2**
Sem filtro	673	226
Primeira filtragem	25	21
Segunda filtragem	6	4

Tabela 3.4: Quantidade de estudos primários obtidos antes e depois de cada filtragem.

É importante ressaltar que, apesar da Revisão Sistemática desta monografia, possuir menos estudos do que o Material Complementar[14], esses estudos representam para a presente monografia um importante ponto de partida para o início e continuidade da pesquisa. A maior parte do referencial bibliográfico da presente monografia, não pertencentes às bases do Portal de Periódicos da CAPES, advêm de artigos da SB *Games*, *sites* de produção de *games* (como o Gamasutra) e de livros sobre desenvolvimento de *games* ou *softwares*.

3.5 Resultados

Após a realização da Revisão Sistemática, pode-se compreender como uma metodologia é importante para a criação de um *game* de boa qualidade. Além disso, foi verificado um aumento da pesquisa na área de desenvolvimento de *games*, particularmente, pesquisas envolvendo metodologias para a produção de *games* (Perani, 2008; Petrillo, 2008; BNDSGediGames, 2014). Apesar do aumento da importância acadêmica, através da quantidade de artigos publicados, não é possível afirmar que exista uma literatura clássica na área de desenvolvimento de *games*. Segundo o relatório BNDSGediGames (2014) e, os estudos primários obtidos através da pesquisa, no Portal de Periódico da CAPES, verificou-se que uma grande parte dos trabalhos acadêmicos ainda estão relacionados a *serious games*[15], e uma menor parte estão relacionados a *games* para o entretenimento. O que é um indício de que a área de desenvolvimento de *games* ainda tem muito a ser explorada, fortalecendo-se cada vez mais no meio acadêmico e consequentemente no meio profissional.

[14] Conjunto de estudos científicos (que fazem parte da referência bibliográfica de um trabalho acadêmico), não presentes na Revisão Sistemática.

[15] *Games* com motivações educacionais.

3.5.1 Respostas para Q1.1

A pesquisa demonstrou que existem várias metodologias para o desenvolvimento de *games*. Algumas, desenvolvidas por empresas e outras por estudos acadêmicos, contudo, todas elas são originadas da Engenharia de *Software*. Das metodologias encontradas, uma grande parte utiliza processos ágeis e uma menor parte baseia-se em metodologias tradicionais como o Modelo em Cascata ou são híbridas (mesclando metodologias tradicionais com processos ágeis). Foram encontradas um total de 12 metodologias relevantes. A seguir, na Tabela 3.5, são listadas essas metodologias e a presença, em sua estrutura, de características ágeis ou tradicionais ou ambas:

Nome	**Ágil**	**Tradicional**
GWP	Não	Sim
Modelo proposto por Sid Meier	Sim	Sim
Modelo proposto por Ed Logg	Não	Sim
Modelo proposto por Rollings e Morris	Não	Sim
RGP	Não	Sim
Game Scrum	Sim	Não
XGD	Sim	Não
GUP	Sim	Sim
AgiGame	Sim	Sim
SUM	Sim	Sim
GAMA	Sim	Não
AGP	Sim	Sim

Tabela 3.5: Metodologias encontradas na Revisão Sistemática.

3.5.2 Respostas para Q1.2

Não foi possível descobrir, exatamente, quais as metodologias de desenvolvimento de *games* são mais utilizadas pelas empresas. Esse fato é compreensível, pois a indústria de *games*, por sua competitividade e seu caráter corporativo, geralmente tornam ina-

cessíveis seus dados internos de projetos a pesquisadores. Outro fato que explica a falta de informações sobre quais as metodologias mais utilizadas é que, apesar das metodologias da Engenharia de *Software* aplicadas ao desenvolvimento de *games* serem um campo de grande interesse, há poucos indícios sobre os avanços neste domínio (Petrillo, 2008; Ampatzoglou e Stamelos, 2010).

Apesar de não se ter um número exato em relação a quantidade de metodologias utilizadas pelas empresas, foi possível obter os tipos de metodologias mais relevantes, de uma forma geral. De acordo com Petrillo (2008) e Godoy e Barbosa (2010), o uso de metodologias ágeis tornaram-se comuns na produção de *games*. No entanto, tais metodologias devem ser adaptadas a realidade da equipe e as particularidades desta indústria. Apesar disso, existem ainda poucas metodologias ágeis que abordem especificamente os problemas e desafios encontrados no desenvolvimento de *games*. É importante destacar que as principais falhas na produção de *games* são motivadas por questões gerenciais, causando atrasos no cronograma e aumentando os custos já elevados desse tipo de projeto. Dessa forma, o uso de metodologias focadas em desenvolvimento de *games*, em especial metodologias iterativas, podem evitar ou minimizar esses problemas. Essa constatação explica o aumento do uso das metodologias iterativas e como consequência, a popularização de técnicas ágeis na indústria de *games*. A metodologia *Scrum*, juntamente com a metodologia XP, são bastante citadas em artigos científicos, envolvendo o desenvolvimento de *games*, e sendo usadas como base para a criação de metodologias mais específicas ainda, em relação a esse tipo de projeto.

Além das metodologias ágeis, na produção de *games* também são utilizadas metodologias tradicionais, principalmente o Modelo em Cascata. Esta metodologia possui uma grande importância histórica para a indústria de *games*, pois para a redução de riscos crescentes, que ocorriam em meados da década de oitenta e início da década de noventa, foi largamente adotada pela indústria e se tornou um padrão para desenvolver *games*. Além disso, é utilizada até hoje, porém de forma adaptada, e não como a metodologia em Cascata original da Engenharia de *Software*. É importante destacar que, a maior parte da indústria de desenvolvimento de *games* usam ou já usaram, de alguma forma, o Modelo em Cascata, como base para o processo de gerenciamento e desenvolvimento em

seus projetos. Por esta razão, o Modelo em Cascata, ainda é considerado como o processo tradicional e padrão da indústria de *games* (McGuire, 2006; Petrillo, 2008; Keith, 2010; Brauwers, 2011; Cruz e Garone, 2013; Albino, Souza e Prado, 2014).

Portanto, as metodologias para o desenvolvimento de *games* surgiram de adaptações de metodologias da Engenharia de *Software*. A seguir, na Tabela 3.6, é apresentado um resumo das metodologias, em geral mais utilizadas pela indústria de *games*, e suas respectivas motivações, respondendo a questão Q1.2.

Metodologias baseadas no Modelo em Cascata
O Modelo em Cascata foi a primeira metodologia, utilizada pelas empresas, para desenvolver *games*.
Foi largamente utilizada e se tornou o processo tradicional e padrão da indústria.
Grande parte das empresas de desenvolvimento de *games* utilizam ou já utilizaram de alguma forma esta metodologia.
É usada até hoje, de forma adaptada, por ser simples e possuir menos riscos, apesar de suas desvantagens em relação a outros tipos de abordagens.
Metodologias baseadas no *Scrum*
Entre as metodologias ágeis de desenvolvimento de *games* é considerada a mais importante e como consequência, a mais utilizada.
São metodologias que se adaptam facilmente a projetos de desenvolvimento de *games*, devido a sua flexibilidade para realizar mudanças de requisitos e ênfase em aspectos gerenciais, sendo esse último um dos principais problemas da indústria.
Metodologias baseadas no *Scrum* estão se tornando comuns na indústria de *games*.
A maioria dos artigos científicos e outros trabalhos acadêmicos, relacionados a metodologias de desenvolvimento de *games*, citam técnicas adaptadas ao *Scrum*.
Metodologias baseadas no XP
Metodologias baseadas no XP estão se popularizando na indústria de *games*, devido ao seu escopo dar ênfase a aspectos de desenvolvimento e, assim como o *Scrum*, adaptar-se facilmente ao cenário desse tipo de projeto.

Os princípios do XP auxiliam o desenvolvimento de *games*, fazendo com que conhecidos problemas da indústria sejam minimizados ou até mesmo eliminados. Dessa forma, aumentando a qualidade dos *games*.
A maioria dos artigos científicos e outros trabalhos acadêmicos, relacionados a metodologias de produção de *games*, citam técnicas adaptadas ao XP. Só perdendo em quantidade, para as referências relacionadas à metodologias baseadas no *Scrum*.

Tabela 3.6: Resumo das metodologias mais utilizadas para o desenvolvimento de *games*.

3.5.3 Respostas para Q2

A Engenharia de *Software* não somente auxilia o desenvolvimento de *games*, como é fundamental para o processo atual de produção da indústria, principalmente em relação a gerir uma grande quantidade de profissionais, chegando a pouco mais de cem pessoas em algumas empresas, e de diferentes áreas de conhecimento. Dessa forma, empresas de médio a grande porte utilizam metodologias da Engenharia de *Software* adaptadas para o desenvolvimento de *games*. Sem essas metodologias, seria inviável comercialmente, gerir um projeto tão complexo e peculiar, com todos os seus problemas e desafios, como é o caso de um projeto de desenvolvimento de *games* (Petrillo, 2008; Park, 2010).

Grande parte dos estudos obtidos pela pesquisa desta monografia, relacionados a importância da Engenharia de *Software* no processo de desenvolvimento de *games*, dizem respeito as dificuldades para produzir *games* e a forma como a Engenharia de *Software* ajuda a tratar os desafios da indústria. Para Taylor et al. (2007) e Petrillo (2008), *games* são um tipo relativamente novo de *software* e como tal, a forma em que são projetados e desenvolvidos ainda está em evolução. Por isso, seu processo de desenvolvimento ainda apresenta resultados variáveis, sobretudo em relação as taxa de falhas serem maiores do que as de sucessos, devido a utilização de metodologias imaturas ou não adaptadas a realidade da indústria. *Games* são *softwares* complexos, e boa parte dessa complexidade ocorre devido a grande quantidade de pessoas envolvidas no processo. Outro motivo para a ocorrência dessa complexidade é a longa duração de um projeto de *games*, de dois a quatro anos, geralmente, para ser concluído. Dessa forma, para lidar com essa

complexidade em desenvolver *games* e gerir melhor esse tipo de projeto, é necessário a utilização de práticas sólidas e comprovadas da Engenharia de *Software*.

Segundo Taylor et al. (2007) e Ollila, Suomela e Holopalnen (2008), outro importante benefício proporcionado pela Engenharia de *Software* está na utilização das metodologias ágeis de desenvolvimento. Estes processos não possuem os rigores dos modelos tradicionais de *softwares* e são indicados a projetos com requisitos flexíveis, complexos e algumas vezes nebulosos, como é o caso de projetos de *games*. A maioria das técnicas ágeis incentivam práticas de desenvolvimento iterativo e incremental (motivo pelo qual, o uso desse tipo de técnica se popularizou na indústria de *games*), suporte à comunicação e valorização das pessoas ao invés de processos, entre outros. Os processos ágeis possuem tamanha importância ao desenvolvimento de *games* que em Albino, Souza e Prado (2014), é citado a relação entre *games*, diversão e flexibilidade. A essência dos *games* para o entretenimento é a diversão. Encontrar esta diversão exige testes, reflexão e adaptações constantes. Não é possível criar *games* de boa qualidade sem adaptar sua jogabilidade, a partir dos resultados obtidos ao longo de todo o seu processo de produção. Desta forma, o desenvolvimento de *games* tem caráter fundamentalmente ágil.

Outros benefícios que a área de Desenvolvimento de *Games* obtém através da Engenharia de *Software*, é em relação às etapas do seu processo de produção e em relação à gerência de projetos, mais especificamente ao gerenciamento de riscos e de qualidade. Apesar do desenvolvimento de *games* e *softwares* tradicionais terem diferenças, principalmente em relação à suas naturezas, entretenimento e negócios, eles possuem o mesmo processo de *software* e consequentemente, passam pelas mesmas etapas de desenvolvimento da Engenharia de *Software*. Existem quatro atividades fundamentais que são comuns a todos os processos de *softwares*: Especificação, desenvolvimento, validação e evolução de *software*. É importante ressaltar que, as atividades envolvidas no processo de desenvolvimento de *games*, no geral, são similares às envolvidas no processo do desenvolvimento tradicional de *software*. Isto se deve ao fato de que, todo *game* é um *software* com características específicas. Dessa forma, todo *game* deve ser resultado de um projeto de *software*. Assim sendo, deve passar por todas as etapas da Engenharia de *Software*, a fim de se tornar um produto de boa qualidade, e consequentemente rentável para a empresa

que o desenvolveu. O desenvolvimento de *games* possui quatro etapas básicas, independente da metodologia utilizada. São elas: Concepção (ou *Game Concept*), Pré-Produção, Produção e Pós-Produção (Barros, 2007; Sommerville, 2007; Chandler, 2012; Carvalho, 2013; Albino, Souza e Prado, 2014). Cada uma dessas fases possuem relações com as etapas clássicas da Engenharia de *Software*.

Na estrutura abaixo, é apresentado os principais benefícios que a Engenharia de *Software* traz para o desenvolvimento de *games*, respondendo a questão Q2 (Barros, 2007; Sommerville, 2007; Petrillo, 2008; Pressman, 2011; Chandler, 2012; Carvalho, 2013; Posvolski et al., 2014).

1. **Processo do Desenvolvimento de *Games*:** Metodologias ágeis aplicadas ao processo de desenvolvimento de *games*.

    1.1. **Processo análogo à Engenharia de *Software*:** Metodologias ágeis de desenvolvimento de *software*.

    1.2. **Benefícios proporcionados pela Engenharia de *Software* neste processo:**

        1.2.1. Melhor gestão de uma quantidade considerável de profissionais, e pertencentes a diferentes áreas de atuação.

        1.2.2. Melhor gestão em relação a projetos de natureza complexa, ou seja, projetos com uma duração que pode passar de quatro anos.

        1.2.3. Auxílio para tratar os principais problemas e desafios da indústria de *games*.

        1.2.4. Melhora na comunicação entre os membros da equipe.

        1.2.5. Respostas rápidas a mudanças que irão ocorrer durante o projeto, sem comprometer o tempo e o custo do mesmo.

        1.2.6. Processo de desenvolvimento iterativo e incremental.

        1.2.7. A qualidade do *game* é testada durante todo o projeto.

2. **Etapa do Desenvolvimento de *Games*:** *Game Concept*.

    2.1. **Etapa análoga à Engenharia de *Software*:** Engenharia de Requisitos.

2.2. **Benefícios proporcionados pela Engenharia de *Software* nesta etapa:**

2.2.1. Facilidade da comunicação e planejamento entre os membros da equipe de desenvolvimento e os clientes, para a escolha da ideia central do *game*.

2.2.2. Criação de estimativas mais realistas para os recursos (custo, pessoal, prazos, ferramentas e equipamentos) do projeto.

2.2.3. Auxiliar o desenvolvimento do projeto com o mínimo de desperdício de tempo e recursos.

3. **Etapa do Desenvolvimento de *Games*:** Pré-Produção.

3.1. **Etapa análoga à Engenharia de *Software*:** Engenharia de Requisitos e Arquitetura de *Software*.

3.2. **Benefícios proporcionados pela Engenharia de *Software* nesta etapa:**

3.2.1. Seleção e agrupamento dos melhores recursos que o *game* deverá ter.

3.2.2. Auxílio na definição das restrições do projeto.

3.2.3. Possibilidade de avaliar se o desenvolvimento do *game* ou os recursos escolhidos são viáveis economicamente para a empresa.

3.2.4. Possibilidade de testar os recursos escolhidos para o *game*, antes de implementá-los, através da construção de protótipos.

3.2.5. Facilidade na comunicação entre a equipe de desenvolvimento através dos diagramas da UML.

3.2.6. Facilidade na criação de documentos que explicitam os recursos, objetivos e outras atividades que serão realizadas no projeto.

3.2.7. Definir (de forma eficiente) a arquitetura do *game* e a interação entre os seus principais elementos (som, arte e código) para a construção do *game* na fase posterior do projeto.

4. **Etapa do Desenvolvimento de *Games*:** Produção.

4.1. **Etapa análoga à Engenharia de *Software*:** Desenvolvimento e Validação de *Software*.

4.2. **Benefícios proporcionados pela Engenharia de *Software* nesta etapa:**

4.2.1. Construção do *game* de forma otimizada, ou seja, sem perda de desempenho durante a sua execução.

4.2.2. Facilidade na manutenção do código do *game*.

4.2.3. Facilidade na realização de testes, correções de erros e garantia de qualidade.

5. **Etapa do Desenvolvimento de *Games*:** Pós-Produção.

5.1. **Etapa análoga à Engenharia de *Software*:** Evolução de *Software*.

5.2. **Benefícios proporcionados pela Engenharia de *Software* nesta etapa:**

5.2.1. Auxílio na atividade de análise do impacto de mudanças no *game*, após este já ter sido lançado no mercado.

6. **Processo do Desenvolvimento de *Games*:** Garantia de Qualidade.

6.1. **Processo análogo à Engenharia de *Software*:** Gerenciamento de Qualidade.

6.2. **Benefícios proporcionados pela Engenharia de *Software* neste processo:**

6.2.1. Garantir que todos os testes sejam realizados.

6.2.2. Guiar a equipe de testes na busca de defeitos, deficiências ou incompatibilidades no projeto.

6.2.3. Garantir que o *game* seja lançado sem defeitos.

6.2.4. Garantir que todos os recursos do *game* estejam implementados.

7. **Processo do Desenvolvimento de *Games*:** Gerenciamento de Riscos.

7.1. **Processo análogo à Engenharia de *Software*:** Gerenciamento de Riscos.

7.2. **Benefícios proporcionados pela Engenharia de *Software* neste processo:**

7.2.1. Identificação dos riscos que possam afetar o projeto.

7.2.2. Auxílio na análise da probabilidade de cada risco ocorrer e o impacto que ele terá sobre o projeto.

7.2.3. Priorização dos riscos de maior impacto, mas sem deixar de lado os de menor impacto.

7.2.4. Criação de um plano de gerenciamento que neutralize ou diminua os riscos.

Portanto, a partir das respostas obtidas às questões Q1.2 e Q2, é observado que sem a utilização de processos e técnicas da Engenharia de *Software* não é possível desenvolver um *game* profissionalmente. Ou seja, um projeto profissional de *game* é um projeto de *software*. Outro ponto a ser destacado é que, programar um *game* sem o mínimo de planejamento é perda certa de recursos e tempo investido. Uma comprovação desse fato é que todas as metodologias encontradas na pesquisa, são baseadas em metodologias da Engenharia de *Software*. Assim sendo, de acordo com Araujo (2006), Barros (2007), Petrillo (2008), Brauwers (2011) e Albino, Souza e Prado (2014), conclui-se que utilizar a Engenharia de *Software* para desenvolver *games* não é mais uma opção, é uma regra.

4 Comparação entre Metodologias de Desenvolvimento de *Games*

Com base nos capítulos anteriores, é visto que para um projeto de *games* ter sucesso é crucial um alto nível de gerenciamento e planejamento, em relação ao processo de desenvolvimento. Um bom gerenciamento aborda o planejamento e coordenação do começo ao fim do projeto, identificando as exigências do cliente, cumprindo cronogramas, custos e padrões de qualidade. Sem um bom planejamento, o desenvolvimento de *games* se torna imprevisível. A imprevisibilidade gerada pela falta de uma metodologia de produção, conhecida como *Code Like Hell*, *Fix Like Hell*, faz o desenvolvimento de *games* basear-se exclusivamente em implementar e corrigir *bugs*, até o final do projeto. Essa falta de uma metodologia e estruturação dentro do processo de desenvolvimento de *games*, traz consigo vários problemas, tais como: Aumento de custos, aumento da ocorrência de riscos deste tipo de projeto, altíssimos níveis de *crunch times*, perda de profissionais ao longo do projeto, atrasos no cronograma, cancelamentos, entre outros. Por isso, é essencial a utilização de metodologias de *softwares* para a produção de *games* (Chagas, 2009; Lavor, 2009; Albino, Souza e Prado, 2014; Posvolski et al., 2014).

4.1 Descrição das Metodologias

4.1.1 GWP

4.1.1.1 Descrição

O Modelo em Cascata ou *Waterfall* foi o primeiro modelo criado para o desenvolvimento de *software*, por volta de 1970. Posteriormente, também foi uma das primeiras técnicas de *design* aplicadas a projetos de *games*. É uma metodologia tradicional e linear, onde há uma subdivisão das etapas de produção de *softwares*. Essas etapas são executadas sequencialmente e dependem totalmente da conclusão da fase anterior para a atual

ser iniciada (Araujo, 2006; Petrillo, 2008; Velasquez, 2009; Cruz e Garone, 2013; Mikoluk, 2013; Freitas, 2014).

Segundo Keith (2009); Machado (2009) e Preisz (2012), houve a necessidade de adaptação do Modelo em Cascata para a indústria, devido a incompatibilidade da rigidez do processo linear e sequencial na produção de qualquer projeto de *games*. Uma abordagem puramente em Cascata, fatalmente levaria a produção de um *game* ao fracasso iminente, e nenhuma empresa sobreviveria a constantes fracassos em seus projetos. A adaptação do Modelo em Cascata para o processo de criação de *games* é conhecida como *Game Waterfall Process*. Esta adaptação possui certa flexibilidade inexistente no modelo tradicional. Dessa forma, com o GWP, não é necessário que uma etapa da produção de *games* seja iniciada apenas quando uma etapa anterior acabar.

4.1.1.2 Vantagens e Desvantagens

A Tabela 4.1, apresenta vantagens e desvantagens na utilização da metodologia GWP (Araujo, 2006; McGuire, 2006; Barros, 2007; Petrillo, 2008; Brauwers, 2011; Cruz e Garone, 2013; Mikoluk, 2013; Posvolski et al., 2014).

Vantagens	**Desvantagens**
Possui estimativas de tempo e custo com maior precisão.	Equipe focada em apenas uma determinada parte do projeto.
Produção de uma documentação completa, o que é útil para orientar a equipe, principalmente com muitos colaboradores.	Testes realizados tardiamente, aumentando consideravelmente a quantidade de *bugs* no *game*.
A ênfase do GWP é o plano de projeto e, portanto, antes de iniciar qualquer tipo de desenvolvimento é preciso haver um plano e visão claros do que será realizado. Isso se deve ao fato de que, o GWP requer um planejamento antecipado e extenso.	Problemas como *feature creeps* e cronogramas otimistas são potencializados, pois as decisões mais importantes do projeto são tomadas quando as incertezas são maiores (principalmente no início da fase de Pré-Produção).
É uma das metodologias mais simples de	Maior ocorrência de períodos de *crunch*

Vantagens	Desvantagens
serem utilizadas.	*times*.
O projeto torna-se mais seguro, devido a característica do GWP de ser orientado a documentação. Por exemplo, se um *game designer* sair da empresa, como o GWP exige um extenso planejamento e documentação, o novo *game designer* pode orientar-se através da documentação, seguindo o plano de desenvolvimento.	A elaboração de documentos detalhados, os chamados *Design Bible*, nas fases iniciais do projeto, podem levar ao escopo irreal e ambicioso, que é um dos problemas da indústria de *games*.
Com o GWP, o projeto é inicializado rapidamente.	Nessa abordagem, a integração do *game* é feita de forma única e tardia. Somente no final do projeto e, com pouco tempo para testes.
Possui gerenciamento de riscos.	Possui como princípio, reunir todos os requisitos necessários no início do projeto. O que é ruim para um projeto de *games*, onde as funcionalidades estão sempre mudando.
Possui um processo de prototipação.	O *feedback*, para a equipe saber se o *game* é bom ou não, ocorre em uma fase tardia do processo, o que pode ser fatal para o projeto.
	Dificuldades na comunicação, principalmente entre programadores e artistas. A consequência direta desse problema, é a produção de um *game* de qualidade inferior.
	Pouca flexibilidade para alterações de *features* a qualquer momento do projeto.

Vantagens	Desvantagens
	Recursos planejados e já implementados são descartados, ocorrendo perda de esforço e desperdício de tempo e custo no projeto.

Tabela 4.1: Vantagens e desvantagens na utilização do GWP.

4.1.2 Modelo proposto por Sid Meier

4.1.2.1 Descrição

Sid Meier é um dos mais respeitados *game designers* do ocidente. Em seus quase vinte anos de experiência, ele trabalhou em vários tipos de projetos de *games*. Sid Meier é mais conhecido por desenvolver *games* como, Pirates, Railroad Tycoon, Covert Action e Civilization. Sua abordagem apresenta um desenvolvimento ágil com a utilização de protótipos, na fase de Pré-Produção. Com isso, é reduzido o tempo e custos do projeto, pois com a prototipação, os membros da equipe tem uma clara visão de como deverão desenvolver o *game*, reduzindo a possibilidade de grandes alterações, principalmente no final do projeto (Rouse, 2005; Kabashi e Saabi, 2008; Johnson, 2009).

4.1.2.2 Vantagens e Desvantagens

A Tabela 4.2, apresenta vantagens e desvantagens na utilização da abordagem criada por Sid Meier (Rouse, 2005; Kabashi e Saabi, 2008; Johnson, 2009).

Vantagens	Desvantagens
Utiliza a prototipação no início do projeto, para testar ideias e analisar a diversão do *game*.	Possui uma eficácia pouco comprovada, principalmente a longo prazo.
Este modelo utiliza uma abordagem de desenvolvimento ágil onde a comunicação é espalhada de forma verbal (comunicação	Dá ênfase na Pré-Produção do *game*, o que pode comprometer outras fases do desenvolvimento, atrasando ou inviabilizando o

Vantagens	Desvantagens
face a face).	lançamento do *game*.
É inteiramente focado na qualidade do *game*, em relação ao seu conteúdo, ou seja, no refinamento constante do seu entretenimento.	Testes realizados tardiamente, apenas no final do projeto.
Utiliza o modelo iterativo.	Possui uma escassa documentação.
	Utiliza uma abordagem, em que o *game designer* também é o programador-chefe, o que pode ser prejudicial para o projeto, principalmente se o *game* for grande e a equipe constituída por muitas pessoas. A visão do *game* ficará limitada a apenas uma pessoa.
	Não é adaptada a grandes mudanças no projeto, caso seja necessário.
	Apesar de Sid Meier apresentar seu modelo como sendo ágil. Essa metodologia é híbrida, com mais elementos de processos tradicionais (como o Modelo em Cascata) do que elementos ágeis de desenvolvimento.

Tabela 4.2: Vantagens e desvantagens na utilização do modelo proposto por Sid Meier.

4.1.3 Modelo proposto por Ed Logg

4.1.3.1 Descrição

Ed Logg é um programador e *game designer*. Ex-funcionário da Atari, trabalhou em diversos *games* famosos dos anos 70 e 80, como Asteroids, Centipede, Gauntlet 1 e 2, Super Breakout, Millipede, Xybots, entre outros. Trabalhou também, nos anos 90,

em adaptações de *games* de *arcade* para consoles, como é o caso do *game* San Francisco Rush. Durante os projetos em que trabalhou, Ed Logg desenvolveu uma metodologia para facilitar o gerenciamento dos *games* que produzia. Sua metodologia utiliza a abordagem tradicional, semelhante ao Modelo em Cascata porém, partes de cada etapa da produção do *game* ocorrem em paralelo, e ainda, utilizando protótipos durante as fases de implementação. O modelo proposto por Ed Logg utiliza uma visão *top down*, onde é imaginado o *game* já pronto, e no decorrer do processo de produção, este é dividido em pequenas partes conceituais, que em seguida, são implementadas (Rouse, 2005; Kabashi e Saabi, 2008).

4.1.3.2 Vantagens e Desvantagens

A Tabela 4.3, apresenta vantagens e desvantagens na utilização da abordagem criada por Ed Logg (Rouse, 2005; Kabashi e Saabi, 2008).

Vantagens	**Desvantagens**
Utiliza o modelo de prototipação entre a Pré-Produção e a Produção do *game*.	Não há informações sobre o uso de *play-tests*.
Possui forte documentação, influenciada pelo Modelo em Cascata.	É basicamente o Modelo em Cascata, onde as etapas da produção do *game* podem ocorrer em paralelo.
	A implementação de recursos, que no final do projeto podem não ser aproveitados, pode gerar perda de tempo, custo e atrasos no projeto.
	Não há uma preocupação com testes durante a implementação. Isso é um forte indício de que, assim como ocorre no Modelo em Cascata, os testes são realizados apenas no final do projeto.
	Apresenta uma fase de Pré-Produção su-

Vantagens	Desvantagens
	perficial, possibilitando o surgimento de sérios problemas de gerenciamento e planejamento durante todo o desenvolvimento do *game*, acarretando em índices altos de cancelamentos de projetos ou um produto final de baixa qualidade.
	O desenvolvimento *top down* traz riscos ao projeto. Pois no final da produção do *game*, recursos já implementados podem ter de serem descartados, gerando um desperdício de tempo e custo. Ou ainda, recursos necessários ao projeto são detectados apenas no final do mesmo, acarretando em possíveis atrasos e aumento de custos na produção do *game*.
	Não há uma fase de Pós-Produção, principalmente a geração do documento de *Post-mortem*.
	Grande possibilidade de ocorrências de *crunch time*, por ser baseada no Modelo em Cascata
	Por ser baseada no Modelo em Cascata, é inflexível. Principalmente quanto a inserção ou alteração de *features* durante o projeto.
	Não há um planejamento das áreas não programadoras.

Tabela 4.3: Vantagens e desvantagens na utilização do modelo proposto por Ed Logg.

4.1.4 Modelo proposto por Rollings e Morris

4.1.4.1 Descrição

Andrew Rollings e David Morris são dois *game designers* com mais de 10 anos de experiência na indústria. A metodologia proposta por eles é uma abordagem iterativa e incremental, com características tradicionais (pois ainda possuem as fases clássicas do desenvolvimento de *games*), e é baseada no conceito de desenvolvimento em camadas: A arquitetura de *Hardware* e a de *Software*. A arquitetura de *Hardware*, representa a interface gráfica, efeitos sonoros e interface de entradas. A arquitetura de *Software* é específica para cada tipo de *game*, dependendo do gênero do mesmo. Na verdade, esse modelo é um *framework* genérico reutilizável, que pode ser usado para desenvolver qualquer tipo de *game*. (Rollings e Morris, 2004; Kabashi e Saabi, 2008).

4.1.4.2 Vantagens e Desvantagens

A Tabela 4.4, apresenta vantagens e desvantagens na utilização da abordagem criada por Rollings e Morris (Rollings e Morris, 2004; Kabashi e Saabi, 2008).

Vantagens	**Desvantagens**
É um modelo de desenvolvimento iterativo e incremental.	Implementação de *features* e componentes que podem nunca serem utilizados.
Possui uma alta capacidade de reuso para novos projetos.	O modelo é extremamente dependente do *framework*.
A produção do *game* ocorre de forma relativamente curta (se o *framework* estiver pronto).	O *framework* pode induzir na equipe uma descrença no gerenciamento e planejamento do projeto, comprometendo o mesmo.
Há uma preocupação em desenvolver soluções para problemas, antes destes ocorrerem, durante o projeto.	Não há uma clara estruturação em relação a atividades importantes do desenvolvimento de *games*, como: *Game Design*, arte e QA.

Vantagens	Desvantagens
	Para projetos futuros, mesmo com o *framework* finalizado, não é garantia que o *game* será concluído em um tempo pequeno. Pois, isso irá depender do gênero do *game* e de quão estável estará o *framework*.
	Nos primeiros projetos de uma empresa que utiliza esse modelo, ela terá que desenvolver o *framework* que será a base de seus *games* e isso irá gastar muito do tempo e orçamento do projeto.
	Apesar de ser um modelo iterativo, possui forte influência do Modelo em Cascata. Como por exemplo a divisão do desenvolvimento em várias etapas, e de forma sequencial.
	Incentivo à inserção de *features* sem o devido planejamento.

Tabela 4.4: Vantagens e desvantagens na utilização do modelo proposto por Rollings e Morris.

4.1.5 RGP

4.1.5.1 Descrição

Rapid Game Process é um modelo de desenvolvimento iterativo que se baseia na metodologia de fábrica de *software*. O termo fábrica de *software* refere-se ao método de produção de sistemas de TI, com técnicas semelhantes às de um padrão de fábrica, como na indústria automotiva. O RGP é na verdade, considerado uma fábrica de *softwares*, projetado para produzir um conjunto de componentes reutilizáveis e ferramentas que

evoluem ao longo do desenvolvimento dos *games* de uma empresa. Baseado no modelo de Rollings e Morris, o RGP usa a abordagem de duas camadas para o desenvolvimento de *games*, com a camada mais interna representando um núcleo reutilizável e a camada externa representando o *design* do *game* (Kabashi e Saabi, 2008).

4.1.5.2 Vantagens e Desvantagens

A Tabela 4.5, apresenta vantagens e desvantagens na utilização da abordagem RGP (Kabashi e Saabi, 2008).

Vantagens	**Desvantagens**
Alta capacidade de reuso para projetos futuros.	O processo de desenvolvimento fica limitado ao *framework*.
O tempo do projeto pode se tornar curto.	Toda a equipe deve possuir experiência, para não utilizar o *framework* de forma indevida e comprometer o projeto.
É flexível, devido a facilidade em incorporar novas características, funcionalidades e módulos ao projeto.	O tempo de desenvolvimento só se torna curto quando a empresa já é experiente, e já lançou vários *games* utilizando essa abordagem.
A prototipagem é feita em paralelo com a fase de Produção.	O código do *game* se torna mais genérico, e consequentemente, mais difícil de desenvolvê-lo.
A equipe possui uma visão melhor de como está o projeto, através da abordagem de prototipação.	Essa metodologia se baseia no método de Rollings e Morris, herdando todas as suas desvantagens.
O conhecimento de todas as etapas do projeto fica disponível para todas as pessoas envolvidas com a produção do *game*.	Projetos iniciais da empresa demorarão mais para serem finalizados. Pois o *framework* ainda não estará pronto.
Maior especificação do projeto.	O RGP é mais focado no *framework* do que no conteúdo do *game*.

Vantagens	Desvantagens
Facilidade no lançamento do *game* para outras plataformas.	Novos desenvolvedores devem aprender sobre as bibliotecas do *framework*. Gerando atrasos no projeto e maior ocorrência de *bugs*.
Possui um gerenciamento de riscos.	

Tabela 4.5: Vantagens e desvantagens na utilização da metodologia RGP.

4.1.6 *Game Scrum*

4.1.6.1 Descrição

O *Game Scrum* é uma metodologia ágil baseada em duas outras técnicas ágeis, o *Scrum* e o XP. A metodologia utiliza o *Scrum* devido ao seu enfoque no gerenciamento de projetos, na sua flexibilidade para ser usada em uma ampla gama de projetos de *games*, e na definição (bem específica) dos papeis para cada membro da equipe e o que cada um vai realizar para a conclusão do projeto. Já, a utilização do XP é devido, ao seu enfoque na engenharia do processo de produção de *software* pois, suas técnicas e princípios foram criados para executar tarefas de forma eficiente. O *Game Scrum* é uma metodologia de desenvolvimento iterativo e incremental, que tem como princípio a aceitação de mudanças de requisitos durante o processo de produção do *game*, de uma forma segura e planejada. Além disso, o *Game Scrum*, através dos processos *Scrum* e XP, utiliza práticas de um constante aprimoramento, permitindo que o processo de desenvolvimento seja sempre ajustado, de forma a atender as necessidades mutáveis do projeto, através da prototipação. É importante destacar que, o *Game Scrum* baseia-se mais no *Scrum* do que no XP, e também é considerado uma adaptação do ciclo de vida do desenvolvimento clássico de *games* para um contexto ágil (Barros, 2007; Petrillo, 2008; Velasquez, 2009; Godoy e Barbosa, 2010; Brauwers, 2011; Chandler, 2012; Carvalho, 2013; Medeiros et al., 2013; Albino, Souza e Prado, 2014).

4.1.6.2 Vantagens e Desvantagens

A Tabela 4.6, apresenta vantagens e desvantagens na utilização da metodologia *Game Scrum* (Araujo, 2006; McGuire, 2006; Barros, 2007; Keith, 2007; Petrillo, 2008; Godoy e Barbosa, 2010; Brauwers, 2011; Chandler, 2012; Carnasciali, 2014; Posvolski et al., 2014).

Vantagens	**Desvantagens**
Equipe qualificada, motivada e coesa.	Possui pouca ou nenhuma documentação.
É simples de ser colocada em prática.	Possui as mesmas desvantagens das metodologias *Scrum* e XP.
Equipes auto-organizáveis. Os produtores não precisam interromper o trabalho da equipe a todo o instante, para verificar o progresso do *game*. Cada membro da equipe controla como o seu trabalho é feito, facilitando a divisão das tarefas em uma lista de pendências. Cada membro da equipe faz seu próprio tempo.	Equipes atuais de desenvolvimento de *games* podem exceder 70 pessoas em tamanho. Com o *Game Scrum*, a equipe não deve exceder 10 pessoas. Como resultado, tem-se 7 equipes focadas em áreas diferentes de um *game*, podendo gerar problemas de dependência e comunicação.
Os eventos do *Game Scrum* são rotineiros e minimizam a necessidade de reuniões desnecessárias, pois estes tem curtas durações, com um tempo fixo para ocorrerem.	É difícil no início para as equipes assumirem a responsabilidade envolvida com o nível de compromisso que o *Game Scrum* demanda.
O emprego do *Game Scrum* torna o projeto como um todo mais visível, para que decisões de senso comum possam ser tomadas, através das reuniões para a remoção de impedimentos.	Possui ciclos de iterações mais longos se comparados a metodologia XGD (que será visto na seção 4.1.7), o que pode atrasar as entregas dos *releases* e como consequência, atrasar o projeto.
Bugs são corrigidos durante a implementação do *game* e não em uma fase es-	Mesmo com as boas práticas do *Game Scrum*, este modelo não é capaz de resol-

Vantagens	Desvantagens
pecífica de testes como em outras metodologias. Dessa forma, o *game* é testado em vários momentos e por diferentes pessoas, resultando em um *feedback* valioso que pode ser tomado como base para futuras decisões.	ver todos os problemas da área de desenvolvimento de *games*. Pode haver ainda, a necessidade de inclusão ou remoção de *features*, detectadas no final do projeto, comprometendo-o negativamente.
É um processo iterativo e incremental. Característica fundamental em projetos de *games*.	Equipes com menos pessoas nem sempre é o ideal para um projeto de *games*.
Permite que todos os profissionais envolvidos com o desenvolvimento do *game* estejam cientes do que está sendo produzido em todos os momentos do processo, exercendo suas atividades em uma mesma direção e com um mesmo propósito.	O *Game Scrum*, assim como o *Scrum*, evita o planejamento antecipado de requisitos. Isto pode ser prejudicial, pois a equipe não terá uma ideia inicial de como será o *game*.
Planejamento ágil. O planejamento deve ser distribuído ao longo de todo o projeto e não concentrado em uma etapa específica.	A metodologia se mostra limitada sobre a visão de *Game Design*, pois é mais focada no gerenciamento de desenvolvimento (implementação e testes).
É focada na priorização de tarefas. Ao longo do projeto, são realizadas várias priorizações que permitem a equipe de desenvolvimento trabalhar sempre no que é mais importante primeiro.	
É focada na comunicação direta. O *Game Scrum* incentiva a comunicação face a face sempre que possível e necessário. Em projetos de *games*, discussões periódicas a res-	

Vantagens	Desvantagens
peito do conceito, da diversão e dos resultados dos testes são fundamentais.	
Adaptabilidade e flexibilidade para projetos com requisitos altamente mutáveis.	
Redução do desperdício de trabalho.	
Maior crença no sucesso do projeto.	
Utilização do processo de prototipação na fase de Pré-Produção do *game*.	
Permite respostas rápidas a mudanças e a resolução de problemas encontrados durante o projeto.	
Possui um gerenciamento descentralizado.	
Preocupação com o gerenciamento de riscos.	
Auxilia no controle do escopo, reforçando que o projeto seja o mais simples possível.	

Tabela 4.6: Vantagens e desvantagens na utilização do *Game Scrum*.

4.1.7 XGD

4.1.7.1 Descrição

Extreme Game Development, ou XGD, é uma metodologia ágil, exclusiva para o gerenciamento e desenvolvimento de *games*, baseada totalmente no processo ágil conhecido como XP. Seu objetivo está em como adaptar o modelo XP à produção de *games*, gerenciando a criação de elementos audio-visuais, integrando as áreas não programadoras e avaliar elementos específicos de *games* (como a jogabilidade), e não apenas dar ênfase a programação como faz o XP. A metodologia XGD, por ser ágil, adota o desenvolvimento iterativo e incremental, aliado a um constante acompanhamento da situação do projeto através da prototipação evolutiva, lançando versões jogáveis e estáveis do *game*, ao final de

cada ciclo de iteração (Demachy, 2003; Barros, 2007; Kasperavicius et al., 2008; Petrillo, 2008; Machado, 2009; Velasquez, 2009; Godoy e Barbosa, 2010; Brauwers, 2011; Freitas, 2014).

4.1.7.2 Vantagens e Desvantagens

A Tabela 4.7, apresenta vantagens e desvantagens na utilização da metodologia XGD (Demachy, 2003; Araujo, 2006; Barros, 2007; Kasperavicius et al., 2008; Petrillo, 2008; Machado, 2009; Godoy e Barbosa, 2010; Brauwers, 2011; Freitas, 2014).

Vantagens	**Desvantagens**
Os membros da equipe ganham um *feedback* rápido em relação ao avanço do projeto.	A metodologia não aborda claramente como trabalhar com o fator multidisciplinar entre as equipes desenvolvedoras de *games*. Áreas como *Game Design*, Arte, entre outras não programadoras, não são mencionadas pela metodologia.
Equipe qualificada, motivada e coesa.	Através de testes unitários automatizados, não é possível concluir que a arte é ruim ou a música não é adequada ou que o *game* não é divertido.
Maior crença no sucesso do projeto.	Possui prazos mais curtos, se comparado a projetos que utilizam metodologias clássicas.
Focada no produto.	Falta de uma documentação necessária ao processo de desenvolvimento de *games*, o que pode gerar problemas de ambiguidade no entendimento de tarefas, e com isso alterar o escopo do projeto.
Incentiva um maior controle de versão.	Possui pouco ou nenhum planejamento de riscos.

Vantagens	**Desvantagens**
Incentiva o uso de boas práticas de programação.	É uma metodologia que exige muito da presença do cliente. O que pode ser um problema, caso este não participe ativamente do projeto.
Possibilidade de sempre verificar a motivação da equipe através das reuniões *stand-up* diárias.	Nem sempre a priorização de tarefas é o ideal. Uma funcionalidade problemática, de alta prioridade, pode travar o processo de desenvolvimento do *game*, impedindo outras *features* de serem produzidas.
A comunicação entre os membros da equipe se torna mais fluida e fácil de ser realizada.	
Preparada para lidar com frequentes mudanças de requisitos.	
Auxílio do controle do escopo, reforçando que o projeto seja o mais simples possível.	
Diminuição do problema de *crunch time*.	
Melhor integração da equipe, pois é focada em uma comunicação eficiente e eficaz.	
Enfatiza uma estreita colaboração entre o *publisher* e o desenvolvedor.	
Desenvolvimento iterativo e incremental.	
Enfoque na garantia de qualidade.	
Gerenciamento descentralizado.	

Tabela 4.7: Vantagens e desvantagens na utilização do XGD.

4.1.8 GUP

4.1.8.1 Descrição

Game Unified Process, ou GUP, é uma metodologia híbrida que combina alguns aspectos da abordagem tradicional RUP com técnicas do modelo ágil XP para o desenvolvimento de *games*. Essa metodologia adota o desenvolvimento iterativo e incremental aliado à divisão de fases, característico do Modelo em Cascata. A metodologia GUP surgiu para resolver os problemas de ineficiência, gerados pelo Modelo em Cascata, para criar *games*, e a necessidade de incluir no processo de produção o desenvolvimento iterativo. Foi a primeira tentativa de adaptar os processos ágeis na indústria de *games*, com o objetivo de atingir um aumento na qualidade e melhorias no gerenciamento de projetos (Araujo, 2006; Barros, 2007; Petrillo, 2008; Godoy e Barbosa, 2010; Brauwers, 2011; Carvalho, 2013).

4.1.8.2 Vantagens e Desvantagens

A Tabela 4.8, apresenta vantagens e desvantagens na utilização da abordagem GUP (Araujo, 2006; Barros, 2007; Petrillo, 2008; Godoy e Barbosa, 2010; Brauwers, 2011; Carvalho, 2013).

Vantagens	Desvantagens
O desenvolvimento de conteúdos se dá de forma iterativa.	Dificuldades para equipes iniciantes em adotar a metodologia.
Documentação curta, porém clara e objetiva. Caso a utilização do XP seja realizada de forma correta, dentro da metodologia GUP.	Não possui um gerenciamento para áreas não programadoras.
Possui as boas práticas da metodologia XP, como o foco em testes, comunicação entre os membros da equipe e a possibilidade do uso da programação em pares.	A documentação extensiva do RUP pode causar problemas de comunicação à equipe de desenvolvimento, devido ao alto nível de detalhamento que o GDD pode possuir.

Vantagens	Desvantagens
Possui no desenvolvimento de *software*, um processo não linear.	Desenvolvimento antecipado de conteúdo, gerando problemas ao projeto, caso uma grande alteração seja necessária.
É focado na criação de uma concepção válida para o *game*.	Não é focada na mudança de requisitos. Característica fundamental para qualquer projeto de *games*.
Utilização de testes contínuos, executados desde as fases iniciais do projeto.	Não possui um gerenciamento de riscos.
	Adota parcialmente as práticas ágeis.
	Possui um gerenciamento centralizado.
	A metodologia carece de mais estudos, com a finalidade de medir sua eficiência para o desenvolvimento de *games*.

Tabela 4.8: Vantagens e desvantagens na utilização do GUP.

4.1.9 *AgiGame*

4.1.9.1 Descrição

AgiGame é uma metodologia híbrida, exclusiva para a produção de *games*, que adota os princípios ágeis aliado a divisão clássica das etapas do ciclo de vida de um *software*, abordadas pelo Modelo em Cascata. Seu objetivo é adaptar a divisão de fases do Modelo em Cascata a uma abordagem ágil; aproveitar os benefícios do desenvolvimento iterativo e incremental; reagir rapidamente a necessidade de mudanças no projeto e; utilizar os valores ágeis para otimizar o processo de produção de *games*, de forma a sempre refinar o entretenimento do *game* até o final do projeto.

4.1.9.2 Vantagens e Desvantagens

A Tabela 4.9, apresenta vantagens e desvantagens na utilização da abordagem *AgiGame* (Posvolski et al., 2014).

Vantagens	**Desvantagens**
É criada uma lista de *features*, cada uma cuidadosamente planejada e projetada.	Não há uma preocupação com as áreas não programadoras.
Dividir a produção do *game* em pequenos blocos de funcionalidades, facilitando e agilizando o processo de implementação, testes e manutenção.	Não há uma preocupação com a prototipação durantes as fases de desenvolvimento de um *game* utilizando essa metodologia.
Os *bugs* são detectados pela equipe de QA e corrigidos durante a implementação, e não em uma fase posterior a mesma.	Há a possibilidade da existência de pouca ou nenhuma documentação sobre o projeto.
Realização de reuniões rápidas e diárias.	Não há indícios da existência de uma fase de Pós-Produção, e consequentemente, da criação do documento de *Post-mortem*.
Possui gerenciamento de riscos, sobre cada *feature* que fará parte do *game*.	A metodologia carece de mais estudos, com a finalidade de medir sua eficiência para o desenvolvimento de *games*.
Acompanhamento da opinião do mercado sobre o *game* que está sendo desenvolvido.	
Desenvolvimento iterativo e incremental.	
Possui ferramentas para saber se o cronograma e o escopo do *game* estão de acordo com sua complexidade.	
Focado em responder de forma rápida a mudanças que podem ocorrer no projeto.	
Focado em QA, de modo a refinar a experiência do *game* até o final do projeto.	

Tabela 4.9: Vantagens e desvantagens na utilização da metodologia *AgiGame*.

4.1.10 SUM

4.1.10.1 Descrição

SUM é uma metodologia que segue o padrão ágil de desenvolvimento, baseando-se em características das metodologias *Scrum* e XP. Os objetivos desta metodologia são, desenvolver *games* de qualidade em um curto espaço de tempo, e administrar os custos e riscos do projeto de uma forma eficiente, obtendo alta produtividade. A metodologia SUM utiliza a abordagem *Scrum* devido a sua capacidade de flexibilização para a implementação de novos recursos, respostas rápidas a mudanças de requisitos e a sua estrutura de iteração. Já a utilização da abordagem XP se dá, devido ao seu curto ciclo de iteração. É importante ressaltar que, o ciclo de vida dos *games* produzidos com a metodologia SUM são baseados no desenvolvimento iterativo e incremental. Esse desenvolvimento é executado em fases e de forma sequencial. (Acerenza et al., 2009).

4.1.10.2 Vantagens e Desvantagens

A Tabela 4.10, apresenta vantagens e desvantagens na utilização da abordagem SUM (Acerenza et al., 2009).

Vantagens	**Desvantagens**
Utiliza o desenvolvimento iterativo e incremental.	Não há um gerenciamento das partes não programadoras.
É flexível, pois possui facilidade em adaptação à mudanças frequentes de requisitos.	Foi desenvolvida para o mercado uruguaio, assim sendo, ela é limitada a esse país, podendo não se adaptar a outros mercados.
Gerenciamento eficiente dos recursos e prazos do projeto.	Não é focada em QA, ou seja, possui uma fase de testes específica, e realizada no final do projeto.
Forte participação do cliente.	Não possui um refinamento da experiência do *game*, durante as iterações do projeto.
Possui como um dos seus principais obje-	É limitada quanto ao escopo do *game*. É

Vantagens	Desvantagens
tivos, extrair a máxima produtividade da equipe de desenvolvimento.	uma metodologia criada, preferencialmente, para a produção de *Advergames*[16].
Possui um gerenciamento de riscos.	Nada é mencionado sobre o grau de detalhamento do GDD ou sobre a sua existência, utilizando essa metodologia.
Possui uma etapa de Pós-Produção que se preocupa em melhorar cada vez mais o processo de produção de *games*.	A metodologia carece de mais estudos, com a finalidade de medir sua eficiência para o desenvolvimento de *games*.

Tabela 4.10: Vantagens e desvantagens na utilização da metodologia SUM.

4.1.11 GAMA

4.1.11.1 Descrição

Game Agile Methods Applied, ou GAMA, é uma metodologia ágil, baseada em fases de desenvolvimento, e aplicada exclusivamente a produção de *games*. O GAMA é uma abordagem que foi desenvolvida para minimizar os principais problemas da indústria e, criar *games* de boa qualidade sem desperdício de tempo e custo.

4.1.11.2 Vantagens e Desvantagens

A Tabela 4.11, apresenta vantagens e desvantagens na utilização da abordagem GAMA (Petrillo, 2008; Brauwers, 2011).

Vantagens	Desvantagens
É inteiramente voltada ao gerenciamento de projetos aplicado ao desenvolvimento de *games*.	Não possui uma estrutura de *playtest* com jogadores finais, o que é fundamental na indústria de *games*.

[16] *Games* com propósitos publicitários.

Vantagens	**Desvantagens**
É focada em QA. Os testes ocorrem durante toda a fase de Produção do *game*, e não apenas em uma etapa específica, como em outras metodologias.	Não utiliza o processo de prototipação do *game*, antes da sua implementação.
É baseada nas metodologias ágeis XP e *Scrum*, reunindo importantes benefícios dessas duas abordagens para a produção de *games*. Como o desenvolvimento iterativo e incremental, automação de testes, importância da comunicação face a face, gerenciamento descentralizado, forte controle de versão, entre outros.	Não possui em sua estrutura, uma parte específica de suporte a adição, alteração ou remoção de *features* no projeto. A metodologia possui um suporte limitado no que diz respeito a adaptar o *game* ao longo do projeto.
Apresenta uma forte estrutura na etapa de Concepção do projeto.	Não possui um gerenciamento multidisciplinar.
Boas práticas de programação como a refatoração e o uso de padrões de projetos.	Não há uma estruturação na metodologia sobre o gerenciamento de riscos.
Diminuição dos problemas de escopo irreal e cronograma otimista.	Nada é mencionado sobre a fase de Pós-Produção do *game*.
Possui uma estrutura para melhorar o processo de produção da empresa.	A documentação pode ficar simples demais, afetando a comunicação, principalmente aos profissionais não programadores.
Frequente refinamento do entretenimento do *game*, através das reuniões diárias e das reuniões de início e fim de cada iteração.	A metodologia carece de mais estudos, com a finalidade de medir sua eficiência para o desenvolvimento de *games*.
Monitoramento constante do projeto.	
O cliente faz parte da equipe.	

Tabela 4.11: Vantagens e desvantagens na utilização do GAMA.

4.1.12 AGP

4.1.12.1 Descrição

Agile Game Process, ou AGP, é uma metodologia ágil, para o desenvolvimento de *games*, baseada nos modelos ágeis *Scrum* e XP. Seu objetivo principal está em criar *games* com melhor qualidade, tendo como foco o gerenciamento do projeto, e dessa forma, atendendo as exigências do cliente. Outro objetivo da metodologia é maximizar a satisfação dos membros da equipe em desenvolver *games*, minimizando ao máximo problemas como *crunch times*, através de um eficiente gerenciamento e otimização da comunicação entre todos os envolvidos no projeto. A motivação da escolha do processo ágil XP pela abordagem AGP, deve-se aos seus ciclos iterativos curtos e suas práticas eficientes de desenvolvimento (como refatoração, automação de testes, integração contínua, programação em pares, entre outros), para projetos de difícil previsão e frequentes mudanças de requisitos, como é o caso do desenvolvimento de *games*. (Araujo, 2006).

4.1.12.2 Vantagens e Desvantagens

A Tabela 4.12, apresenta vantagens e desvantagens na utilização da abordagem AGP (Araujo, 2006).

Vantagens	**Desvantagens**
Desenvolvimento iterativo e incremental.	Não possui um processo específico de gerenciamento de riscos.
Maximiza a comunicação entre todos os envolvidos no projeto, através das práticas ágeis.	Apesar da possibilidade de adaptação, é uma metodologia voltada exclusivamente para *games* do tipo *Advergames*.
É voltada ao gerenciamento de projetos para *games*, pois dá ênfase na característica mutável de requisitos, durante todo o desenvolvimento.	Apesar de se basear em processos ágeis, principalmente na metodologia *Scrum*, ainda possui um desenvolvimento linear baseando-se em geração de versões alfa, beta e *gold*.

Vantagens	Desvantagens
Possui as boas práticas das metodologias XP e *Scrum*.	Não possui um gerenciamento multidisciplinar.
Orientado a testes.	Não utiliza o processo de prototipação do *game*, antes da sua implementação.
Gerenciamento descentralizado.	A documentação pode ficar simples demais, afetando a comunicação, principalmente aos profissionais não programadores.
Possui *feedback* antecipado. Principalmente em relação ao conteúdo do *game*, ou seja, o seu *gameplay*.	A metodologia se mostra limitada sobre a visão de *Game Design*, pois é mais focada no gerenciamento de desenvolvimento (implementação e testes).
Diminuição de períodos de *crunch times*, aumentando a satisfação da equipe em participar do projeto.	Nada é mencionado sobre a diminuição de problemas conhecidos da indústria como, atrasos no cronograma e aumento de custos no projeto.
Forte participação do cliente.	A metodologia carece de mais estudos, com a finalidade de medir sua eficiência para o desenvolvimento de *games*.

Tabela 4.12: Vantagens e desvantagens na utilização do AGP.

4.2 Critérios para a Análise Comparativa

Existem características que toda metodologia deve possuir, de modo a guiar a equipe de desenvolvimento em produzir *games* de boa qualidade e diminuindo ao máximo os principais problemas da indústria como atrasos no cronograma, extrapolação de custos, entre outros. Nesta seção, tais características serão condensadas na forma de critérios, representando as boas práticas necessárias à indústria, com a motivação de auxiliar na comparação entre as metodologias identificadas na presente monografia, de forma à en-

contrar as técnicas mais apropriadas à produção de *games*. A Tabela 4.13, descreve os critérios que serão utilizados para a comparação entre as metodologias.

Desenvolvimento ágil.
A agilidade é uma característica necessária à produção de *games*, para a minimização dos problemas da indústria e aumento da qualidade do *game* através de benefícios como motivação da equipe, crença no sucesso do projeto, foco no produto, adaptação à mudanças frequentes de requisitos, entre outros (Petrillo, 2008).
Desenvolvimento iterativo e incremental.
O desenvolvimento iterativo e incremental é fundamental para a produção de *games*. Com o processo iterativo e incremental, mudanças de requisitos durante toda a produção de um *game*, são vistas como algo importante e necessário, e não como uma falha de planejamento. Os principais benefícios alcançados pelo processo iterativo e incremental são: Geração de valor antecipado para o cliente; aumento do controle de versão do produto; detecção antecipada de erros; entre outros (Araujo, 2006; Petrillo, 2008).
Gerenciamento descentralizado.
O gerenciamento descentralizado gera equipes auto-organizáveis, que possuem autonomia em tomar importantes decisões para o projeto sem depender da burocracia existente nos processos tradicionais, onde apenas uma pessoa toma as decisões críticas do projeto. Outro benefício do gerenciamento descentralizado é a melhor distribuição da visão do projeto a todos os membros da equipe de desenvolvimento, tornando a produção de *games* um processo mais colaborativo, aumentando a motivação da equipe e crença no sucesso do projeto (Araujo, 2006).
Gerenciamento da fase de Concepção.
Um maior planejamento e gerenciamento da etapa de Concepção é fundamental para a produção de um *game*. A ideia central do *game* é responsável pelo seu sucesso ou fracasso no mercado. Dessa forma, um bom gerenciamento dos conceitos gerados nesta fase, é necessário para a construção de *games* de boa qualidade.
Gerenciamento da subetapa de *Game Design*.
Uma das principais atividades dentro da produção de *games*, o *Game Design* é a essência

de qualquer *game*, independente do gênero, plataforma ou tecnologia utilizada. Todas as outras atividades do projeto, como codificação, arte e sonorização, são desenvolvidas com base na criação de um bom *Game Design*. Desse modo, para construir *games* de boa qualidade é fundamental um alto nível de gerenciamento desta atividade, com atenção especial aos requisitos como jogabilidade, *gameplay*, entre outros (Petrillo, 2008).
Gerenciamento de tempo.
Atrasos no cronograma são um dos problemas mais recorrentes da indústria de *games*. Um bom gerenciamento sobre os prazos de cada atividade e constante monitoramento do tempo para sua conclusão, é uma forma de minimizar os atrasos que ocorrem na maioria dos projetos.
Gerenciamento de custos.
Outro problema bastante comum da indústria, a extrapolação dos custos de produção, deve ser tratado com atenção especial, através de um bom gerenciamento sobre cada requisito que será desenvolvido no *game* e, um profundo planejamento sobre a inserção de novas funcionalidades, remoção ou alteração daquelas que já estavam previstas no projeto. E sempre baseando-se no quanto será gasto para realizar estas alterações.
Gerenciamento de riscos.
Em um projeto de *games*, a quantidade de riscos e o impacto que eles podem causar são altos. Dessa forma, é de suma importância que, durante a produção de um *game*, exista um processo de gerenciamento de riscos, de forma a listá-los, monitorá-los e atualizá-los durante todo o projeto, de modo a minimizar ao máximo suas ocorrências (Araujo, 2006)
Gerenciamento da multidisciplinaridade.
Uma falha que ocorre em quase todas as metodologias encontradas no Capítulo 2, devido a sua inexistência. O gerenciamento de outras disciplinas, que não sejam somente programação e testes, como a área artística, sonora, modelagem, *Game Design*, entre outras; gera um controle maior sobre a qualidade do *game*. Além disso, de acordo com Araujo (2006), o gerenciamento da multidisciplinaridade tem a função de integrar, de forma eficiente, profissionais de diferentes áreas de atuação, potencializando a comuni-

cação entre todos os participantes do projeto.
Contínua verificação da qualidade do *game*.
Quando o processo de produção de um *game* é focado na garantia de qualidade, os testes são realizados de forma contínua, em todas as iterações da fase de desenvolvimento, em paralelo a programação e na mesma iteração (e não apenas em uma etapa separada do desenvolvimento do *game*, ao final do projeto). Dessa forma, os *games* podem ser lançados com uma menor quantidade de *bugs*, e com uma maior qualidade em relação a elementos como jogabilidade, *gameplay*, entre outros (Barros, 2007).
Contínua verificação da qualidade da produção do *game*.
É essencial que uma empresa desenvolvedora tenha a preocupação em estar sempre refinando seu processo de produção, com a finalidade de aumentar cada vez mais a qualidade de seus *games*. A existência de ferramentas que auxiliem os produtores nesta tarefa, potencializam ainda mais a melhoria da qualidade do processo de produção de uma empresa de *games*.
Flexibilidade.
É a principal característica que um processo de desenvolvimento de *games* deve possuir. Ser totalmente adaptado a mudanças de requisitos, durante todo o projeto, sem comprometê-lo e, com a mínima perda de tempo e custo (Araujo, 2006). A flexibilidade em um processo de produção dá ao produtor novas possibilidades para refinar o entretenimento de um *game*.
Práticas de automação de testes.
A automação de testes é uma importante ferramenta que traz vários benefícios à equipe de desenvolvimento, tais como: Eficiência em escrever códigos mais estáveis, capacidade maior de manutenção, *feedback* e detecção de *bugs* de forma antecipada, testes realizados quantas vezes forem necessários, entre outros (Petrillo, 2008).
Refinamento do *Gameplay*.
Elementos como diversão, surpresa, desafios e imersão fazem parte do entretenimento de um *game*, ou seja, do seu *gameplay*. Dessa forma, todo o processo de desenvolvimento de *games* deve possuir atividades para que o seu conteúdo seja continuamente avaliado

e refinado. Gerando assim, a melhor experiência possível aos jogadores quando jogarem o *game*. Isso deve ser feito durante todo o projeto. Caso, o processo de produção possua ferramentas para esse refinamento, a detecção da necessidade de mudanças para tornar um *game* o mais interessante possível, serão potencializadas e otimizadas.
O cliente faz parte da equipe.
A participação constante do cliente para um projeto, em especial de *games*, é importante pois, quaisquer dúvidas em relação ao projeto serão sanadas imediatamente. Evitando com isso, o desperdício de tempo (pela espera do esclarecimento do cliente) ou desperdício de custo (por funcionalidades implementadas de forma equivocada, sem o consentimento do cliente), acarretando em um *game* de qualidade inferior.
Agrupamento de *features*.
Em um projeto de *games*, a quantidade de funcionalidades a serem implementadas é muito grande, chegando a milhares de *assets* a serem projetados. Com isso, é fácil que a equipe perca o foco, em relação a qual requisito será implementado para gerar valor ao *publisher*. Portanto, com o intuito de diminuir problemas como atrasos no cronograma e períodos de *crunch times*, é de fundamental importância listar, agrupar e priorizar cada funcionalidade a ser implementada, de acordo com os anseios do *publisher* (Barros, 2007; Brauwers, 2011).
Playtesting.
O *playtesting* auxilia a equipe de desenvolvimento no refinamento do entretenimento do *game*, além de melhorias em elementos como o balanceamento, jogabilidade, entre outros (Brauwers, 2011). O principal benefício do *playtesting* é o real *feedback* de potenciais jogadores, que representam o público-alvo do *game* que está sendo desenvolvido.
Suporte a comunicação.
A comunicação é o principal elo que integra todos os membros da equipe de desenvolvimento (inclusive o *publisher*), trabalhando de forma coesa e focados em um mesmo objetivo. É importante que a metodologia utilizada, possua práticas de forma a potencializar a comunicação de todos os envolvidos no projeto. O enfoque à comunicação

na produção de *games* trazem benefícios, tais como: Aumento da qualidade do *game*, distribuição otimizada e eficiente da visão do projeto aos membros da equipe, *feedback* antecipado, entre outros (Araujo, 2006; Barros, 2007; Petrillo, 2008).
Integração contínua.
A integração contínua é necessária em um processo de desenvolvimento de *games*, de forma a evitar o surgimento de *bugs* críticos no final do projeto, melhorar a visão do projeto a todos os membros da equipe e diminuir atrasos no cronograma. Sua principal motivação é, integrar os módulos do *game* ao final de cada iteração, através de uma versão estável e jogável, durante toda a fase de Produção (Barros, 2007).
Suporte a diferentes tipos de reuniões.
Reuniões são de extrema importância em qualquer projeto de *software*, em especial, projetos de *games*. Pois, é através delas que o projeto é iniciado, o planejamento é realizado, questões fundamentais são discutidas, entre outros. Um processo de desenvolvimento de *games* deve estar apto a incentivar a prática de diversos tipos de reuniões, e durante toda a produção do *game*.
Prototipação.
A motivação e importância da prototipação no processo de desenvolvimento de *games*, é poder testar ideias, elementos de jogabilidade, formas de balancear o *game*, entre outros; ainda na etapa de Pré-Produção, ou seja, quando a implementação do *game* ainda não começou (Brauwers, 2011). Dessa forma, é possível avaliar itens que ainda estão no papel (ou seja, apenas foram planejados). De modo que, caso não se adaptem ao *game*, simplesmente são descartados do projeto, antes de serem tomadas decisões definitivas.
Processo de desenvolvimento dividido em fases.
A importância de um processo de desenvolvimento de *games* ser dividido em etapas, está na facilidade em organizar e planejar o projeto de uma forma eficaz. Desse modo, com um projeto melhor planejado, diminui a ocorrência de problemas de escopo ambicioso, cronograma irreal, implementação de funcionalidades a qualquer momento do projeto (sem nenhum tipo de organização), entre outros.

Fluxo de desenvolvimento não linear.
Assim como é importante a divisão em fases, para um processo de desenvolvimento de *games* ser eficiente e eficaz, é fundamental que suas etapas não estejam engessadas (ou seja, uma etapa só pode iniciar ao término da etapa anterior). Portanto, o fluxo de desenvolvimento de um *game* deve ser flexível, sendo suas fases executadas de forma paralelas, uma das outras, de acordo com critérios estabelecidos pelas empresas e pelo produtor.
Constante monitoramento do projeto.
É necessário que uma metodologia de produção de *games* possua ferramentas e incentive práticas de um contínuo monitoramento do progresso do projeto, durante todo o desenvolvimento. Pois, caso ocorram problemas, através do monitoramento constante, o produtor poderá responder de forma rápida a estes impedimentos, colocando o projeto de volta na direção correta.

Tabela 4.13: Critérios para a análise comparativa das metodologias de desenvolvimento de *games*.

4.3 Análise Comparativa

As metodologias encontradas nesta monografia, apresentam falhas que resultam, no geral, em altas taxas de fracassos para projetos de *games*. Esses fracassos vão desde *games* lançados com altas quantidades de *bugs*, *games* com uma qualidade bem inferior ao esperado, até o cancelamento do projeto. As metodologias atuais, em sua grande maioria, ainda estão presas ao paradigma "codificar, consertar". Para um projeto ser bem sucedido e gerar um *game* de boa qualidade, a metodologia escolhida deve ter como o seu principal fundamento o planejamento minucioso de todas as etapas de um processo de desenvolvimento de *games*, o gerenciamento de todas as disciplinas envolvidas no projeto, e flexibilidade diante aos desafios da indústria. E não apenas preocupar-se com a programação, como ocorre em algumas metodologias para a produção de *games*.

4.3.1 Comparação

A Tabela 4.14, apresenta a legenda dos códigos relacionados a cada critério apresentados na tabela de comparação.

Código	Critério
C01	Desenvolvimento ágil
C02	Desenvolvimento iterativo e incremental
C03	Gerenciamento descentralizado
C04	Gerenciamento da fase de Concepção
C05	Gerenciamento da subetapa de *Game Design*
C06	Gerenciamento de tempo
C07	Gerenciamento de custos
C08	Gerenciamento de riscos
C09	Gerenciamento da multidisciplinaridade
C10	Contínua verificação da qualidade do *game*
C11	Contínua verificação da qualidade da produção do *game*
C12	Flexibilidade
C13	Práticas de automação de testes
C14	Refinamento do *Gameplay*
C15	O cliente faz parte da equipe
C16	Agrupamento de *features*
C17	*Playtesting*
C18	Suporte a comunicação
C19	Integração contínua
C20	Suporte a diferentes tipos de reuniões
C21	Prototipação
C22	Processo de desenvolvimento dividido em fases
C23	Fluxo de desenvolvimento não linear

C24	Constante monitoramento do projeto

Tabela 4.14: Relação entre código e seus respectivos critérios.

A Tabela 4.15, apresenta a legenda dos códigos relacionados a cada metodologia utilizada na tabela de comparação.

Código	**Metodologia**
M01	GWP
M02	Modelo Proposto por Sid Meier
M03	Modelo Proposto por Ed Logg
M04	Modelo Proposto por Rollings e Morris
M05	RGP
M06	*Game Scrum*
M07	XGD
M08	GUP
M09	*AgiGame*
M10	SUM
M11	GAMA
M12	AGP

Tabela 4.15: Relação entre código e suas respectivas metodologias.

A Tabela 4.16, apresenta a comparação entre as metodologias encontradas na Revisão Sistemática realizada, através de critérios que representam práticas, de forma a produzir *games* com boa qualidade e gerenciando eficientemente os problemas da indústria. É importante destacar que, como cada critério representa uma boa prática, quanto mais critérios forem atendidos por uma metodologia, mais preparada ela estará para produzir *games*.

	M01	**M02**	**M03**	**M04**	**M05**	**M06**	**M07**	**M08**	**M09**	**M010**	**M011**	**M012**
C01		X				X	X	X	X	X	X	X

	M01	M02	M03	M04	M05	M06	M07	M08	M09	M010	M011	M012
C02		X		X	X	X	X	X	X	X	X	X
C03						X	X		X	X	X	X
C04											X	
C05												
C06									X	X		
C07										X		
C08	X				X	X			X	X		
C09												
C10						X	X	X	X		X	X
C11						X					X	
C12		X		X	X	X	X	X	X	X		X
C13						X	X	X			X	X
C14		X		X	X	X	X	X	X	X	X	X
C15						X	X		X	X	X	X
C16						X	X		X	X	X	X
C17												
C18						X	X		X	X	X	X
C19				X	X	X	X	X	X		X	X
C20						X	X		X		X	X
C21	X	X	X		X	X	X					
C22	X	X	X		X				X	X	X	X
C23		X				X	X	X	X		X	
C24						X	X		X		X	

Tabela 4.16: Comparação entre metodologias de desenvolvimento de *games*.

A Figura 4.1, apresenta o gráfico sobre o percentual de critérios que cada metodologia atende.

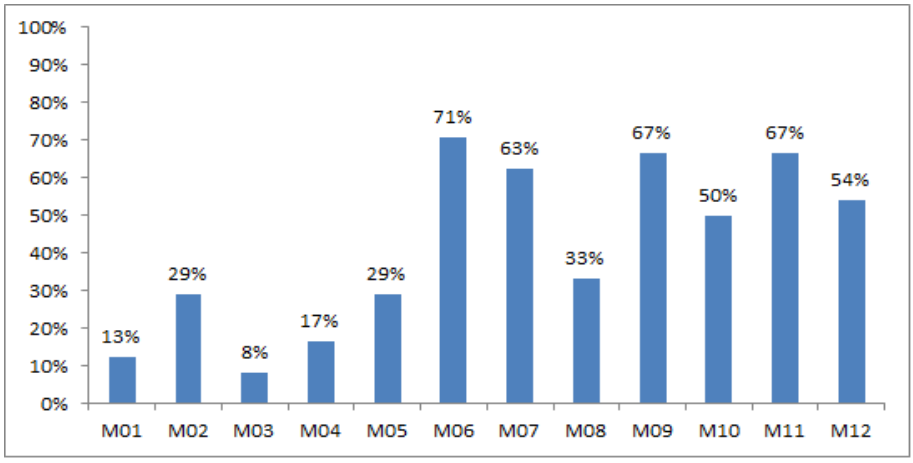

Figura 4.1: Percentual da quantidade de critérios atendidos por cada metodologia.

A Figura 4.2, apresenta o gráfico sobre o percentual de metodologias pertencentes a cada critério.

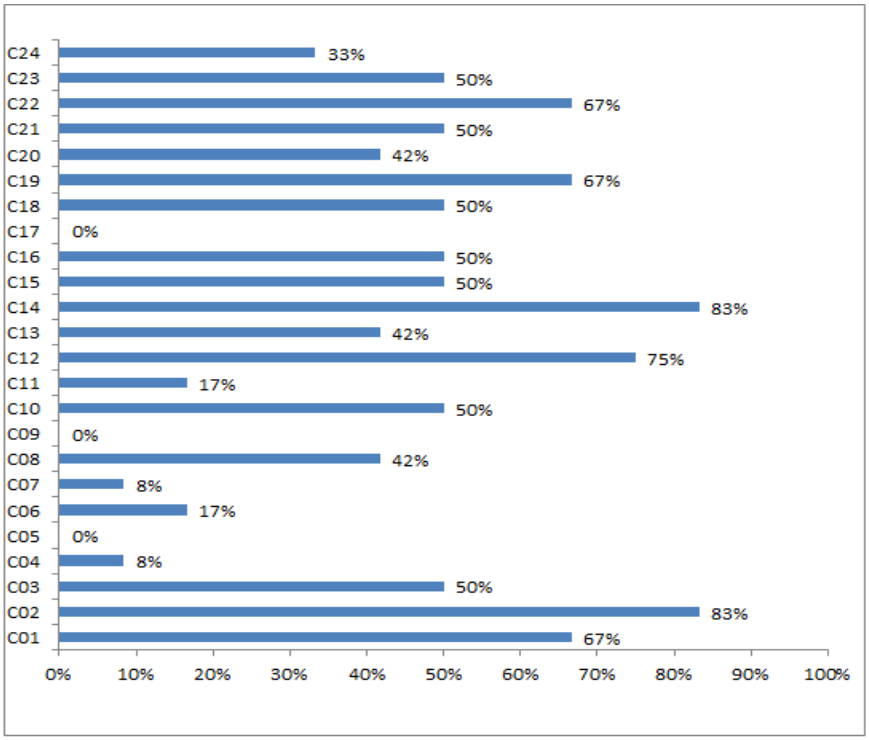

Figura 4.2: Percentual da quantidade de metodologias pertencentes a cada critério.

4.3.2 Análise Por Metodologia

4.3.2.1 GWP

Apesar da importância histórica que a metodologia GWP possui para a indústria, além da inclusão de práticas que perduram até hoje, em toda produção profissional de *games*, como gerenciamento de riscos, divisão do desenvolvimento em fases e a inclusão do processo de prototipação; a metodologia se encontra obsoleta em relação a produção atual de *games*, devido aos poucos critérios de comparação atendidos pelo GWP. A inexistência de critérios como o desenvolvimento iterativo e incremental, contínua verificação da qualidade, integração contínua e o refinamento do *gameplay*; tornam a metodologia menos adaptada à produção de *games*. Esta constatação se agrava ainda mais pelas características da metodologia de centralização gerencial, não ser flexível a mudanças de requisitos e principalmente, ao seu fluxo de desenvolvimento apresentar-se de forma linear; contribuindo para a perda da qualidade do *game* durante o projeto.

4.3.2.2 Modelo proposto por Sid Meier

O modelo proposto por Sid Meier possui importantes critérios como agilidade, desenvolvimento iterativo e incremental, flexibilidade, refinamento do *gameplay*, entre outros. O que indica a preocupação com a qualidade do *game* em relação ao seu conteúdo e adaptação a mudanças de requisitos. Contudo, a metodologia possui problemas, evidenciados pela inexistência de muitos critérios de comparação em seu fluxo de desenvolvimento. Por exemplo: Possui um gerenciamento centralizado, não possui um gerenciamento de riscos, não é focada em QA (não atende aos critérios C10 e C11), não possui suporte a comunicação, entre outros. Isso mostra a defasagem da metodologia, pois a inexistência dos critérios mencionados anteriormente, fazem com que os problemas da indústria sejam potencializados ao produzir *games* com essa metodologia.

4.3.2.3 Modelo proposto por Ed Logg

O modelo proposto por Ed Logg, é um dos mais defasados para a produção profissional de *games*. Os únicos critérios atendidos por essa metodologia (a prototipação e a divisão do desenvolvimento em fases) não são suficientes para otimizar a criação de *games*. A inexistência de quase todos os critérios de comparação ao fluxo da produção de *games*, utilizando o modelo de Ed Logg, faz com que os problemas da indústria fiquem ainda mais intensos e frequentes.

4.3.2.4 Modelo proposto por Rollings e Morris

O modelo proposto por Rollings e Morris possui elementos importantes para a produção de *games* relacionados a qualidade de seu conteúdo, através da inclusão de critérios exclusivos da indústria, como a flexibilidade e o refinamento do *gameplay*. A metodologia também é baseada no desenvolvimento iterativo e incremental, um importante aspecto para a produção de *games*. Entretanto, a não inclusão de importantes critérios como gerenciamento descentralizado, gerenciamento de riscos, contínua verificação da qualidade do *game* e do processo de produção; mostram que a metodologia não é focada na garantia de qualidade, e consequentemente, *games* criados com esse processo de produção fiquem aquém do esperado pelo mercado. Adicionado a isso, a possibilidade

do lançamento do *game* com *bugs*, comprometendo a imagem da empresa perante o seu público-alvo. Outro aspecto negativo da metodologia, é o fato de seu fluxo de desenvolvimento ser linear, baseado no obsoleto Modelo em Cascata. O que é uma característica incompatível com a indústria de *games* nos dias de hoje, sendo um dos seus elementos fundamentais, o fluxo não linear das etapas de produção durante o ciclo de desenvolvimento do projeto.

4.3.2.5 RGP

O RGP inclui em seu processo de desenvolvimento, importantes critérios, tidos como elementares para a criação de *games*, tais como: Gerenciamento descentralizado, gerenciamento de riscos, prototipação, além de ser baseado no desenvolvimento iterativo e incremental. Outro ponto positivo para a metodologia é o fato dela estar focado em gerar conteúdo de valor ao *game* e estar preparada para adaptar-se a mudanças de requisitos do projeto, sempre que necessário, com base na presença de critérios como a flexibilidade e o refinamento do *gameplay*. Entretanto, a ausência de alguns critérios fundamentais para a indústria, como a contínua verificação da qualidade e do processo de produção, trazem a possibilidade do *game* ser lançado com *bugs* e sem funcionalidades previamente planejadas, ou seja, diminuindo a qualidade do mesmo. Outro ponto negativo, e que torna a metodologia defasada, é a inexistência do critério de agilidade. O RGP é uma metodologia tradicional, baseada no Modelo em Cascata, ou seja, incorpora em seu fluxo de produção, atividades incompatíveis à indústria profissional de *games*. Como por exemplo, a existência de uma etapa específica e tardia de busca e correção de *bugs*, potencializando problemas como atrasos do cronograma e extrapolação dos custos do projeto.

4.3.2.6 *Game Scrum*

O *Game Scrum* é uma das metodologias mais adaptadas à produção atual de *games*, devido a incorporação de grande quantidade de critérios de comparação (mais de 70% dos critérios, de acordo com a Figura 4.1). Além da metodologia ser inteiramente ágil, é baseada no desenvolvimento iterativo e incremental. Outra importante característica para o desenvolvimento de *games* é que, o *Game Scrum* possui um gerenciamento des-

centralizado e dá ênfase ao gerenciamento de riscos, diminuindo conhecidos problemas da indústria devido a um maior planejamento, antes e depois de qualquer atividade realizada no projeto. A metodologia também incorpora em seu fluxo de produção, os critérios exclusivos da indústria de *games* como flexibilidade e, refinamento do *gameplay*; gerando um *game* de valor, tanto para o *publisher*, quando para o público-alvo.

Outro importante grupo de critérios atendidos pela metodologia, são aqueles relacionados a QA (C10, C11 e C24). A presença destes critérios indicam uma constante preocupação na entrega de um *game* de alta qualidade e na constante melhoria do processo de produção da empresa, sempre com o objetivo de criar *games* cada vez melhores. Contudo, a metodologia *Game Scrum* possui pontos negativos devido a inexistência de alguns critérios de comparação (tais como os gerenciamentos de multidisciplinaridade, de tempo e de custos do projeto, bem como a prática de *playtesting*) que, caso fossem incorporados ao seu fluxo de produção, poderiam otimizar ainda mais o desenvolvimento de *games*, de forma a melhorar a qualidade do mesmo.

4.3.2.7 XGD

O XGD é uma metodologia compatível em relação a produção profissional de *games*, devido ao fato de atender a mais de 60% dos critérios de comparação. Desses critérios, destacam-se sua agilidade, seu gerenciamento descentralizado, ênfase na comunicação, ênfase em QA, possui um desenvolvimento iterativo e incremental, e principalmente, está adaptada a mudanças de requisitos. Entretanto, o XGD possui pontos falhos que podem neutralizar os benefícios trazidos pelas boas práticas, em relação aos critérios de comparação que a metodologia atende. Esses pontos tem relações com a falta de alguns critérios inexistentes na metodologia. Destes critérios, destacam-se a falta de um gerenciamento de riscos e o suporte a multidisciplinaridade, podendo com isso afetar o projeto negativamente, aumentando a ocorrência de atrasos no cronograma e aumento de custos, além da possibilidade do *game* se tornar diferente do inicialmente planejado, devido a não existência de um gerenciamento específico das disciplinas artísticas e de *design*.

4.3.2.8 GUP

O GUP possui apenas alguns elementos importantes à indústria como o desenvolvimento iterativo, ênfase na qualidade do *game*, o refinamento do *gameplay* e é adaptado a mudança de requisitos. Porém, os critérios de comparação não preenchidos pela metodologia chegam a quase 70%. A falta desses critérios (possui um gerenciamento centralizado, sem suporte a comunicação e a multidisciplinaridade, inexistência de um gerenciamento de riscos, entre outros), fazem do GUP uma metodologia defasada e não indicada ao desenvolvimento profissional de *games*.

4.3.2.9 *AgiGame*

A metodologia *AgiGame*, junto com o *Game Scrum*, é uma das técnicas mais adaptadas a produção de *games*. Sendo focado inteiramente na produção ágil, o *AgiGame* possui importantes práticas como priorização de *features*, reuniões de planejamento, cliente como parte da equipe e gerenciamento descentralizado. Estas práticas fazem com que diminuam vários problemas tradicionais da indústria, com a ocorrência de *crunch times*, escopo ambicioso, cronograma otimista, entre outros. O *AgiGame* é uma das únicas metodologias focadas em um específico gerenciamento do cronograma do projeto. Desta forma, qualquer possibilidade de atrasos nos prazos de entregas são minimizados, devido a prática de um constante monitoramento do projeto, incentivada pela metodologia. Além disso, o *AgiGame* também é focado em QA, práticas de gerenciamento de riscos e na reestruturação do projeto, caso requisitos sejam alterados, durante o mesmo. É importante destacar que, a metodologia falha em alguns pontos como: Inexistência de um processo de prototipação, inexistência de práticas de *playtesting*, não há um gerenciamento específico da multidisciplinaridade e não possui práticas para melhorar a produção da empresa.

4.3.2.10 SUM

SUM é uma das metodologias intermediárias, encontradas neste trabalho. Ou seja, não possui nem muitos pontos fortes ou fracos. A técnica SUM atende a exatamente 50% dos critérios de comparação, e mesmo tendo a desvantagem de ter sido criada para um específico mercado de *games* (o uruguaio), os critérios existentes nesta metodologia

são importantes. Destes critérios, destacam-se o gerenciamento descentralizado, utilização de práticas e valores ágeis (como agrupamento de *features*, cliente como parte da equipe, e suporte a comunicação), desenvolvimento iterativo e incremental, e principalmente, estar preparada para mudanças frequentes de requisitos durante a produção do *game*. É importante destacar que, a metodologia SUM é a única de todas as técnicas pesquisadas que possui os três tipos de gerenciamento, vistos na apresentação dos critérios de comparação e fundamentais ao desenvolvimento de *games*. Sendo estes: Os gerenciamentos de tempo, de custos e de riscos.

A presença destes gerenciamentos, fazem a metodologia SUM destacar-se dentre as outras. Contudo, a inexistência de alguns critérios como a falta de um processo de prototipação, inexistência do gerenciamento da multidisciplinaridade e falta de práticas de *playtesting*, fazem da SUM uma metodologia menos adaptada à indústria do que outras já analisadas. Em relação aos pontos fracos da SUM, o que mais se destaca é o seu processo de produção ainda linear e principalmente, o fato de não ser focada em QA. Sendo que a SUM, ainda faz uso da já defasada etapa de testes ao final da produção do *game*.

4.3.2.11 GAMA

O GAMA é mais uma das metodologias ágeis, compostas por muitos dos critérios de comparação (quase 70%), que representam boas práticas para a produção de *games*. Dos critérios atendidos pelo GAMA, os responsáveis pelo destaque desta metodologia são: Ser inteiramente focada em QA, possuir um gerenciamento descentralizado, refinar o *gameplay* durante toda a produção do *game*, possuir suporte a comunicação e, ferramentas para um constante monitoramento do projeto. Além disso, o GAMA é a única metodologia (das doze técnicas encontradas) que possui em seu fluxo de desenvolvimento, um específico gerenciamento para a etapa de Concepção. Reforçando ainda mais a distribuição da visão do *game* para todos os envolvidos no projeto. Entretanto, dos critérios não preenchidos pela metodologia, a falta de alguns deles podem afetar a qualidade do *game*, tais como: Não possuir gerenciamento de riscos; não possuir um gerenciamento para a multidisciplinaridade; inexistência de um processo de prototipação e principalmente; ser limitada quanto a flexibilidade, ou seja, não possui uma estrutura para inserção, alteração

ou remoção de requisitos, de forma segura (sem comprometer os prazos e custos do projeto).

4.3.2.12 AGP

O AGP, junto com a metodologia SUM, é mais uma técnica intermediária encontrada na pesquisa desta monografia, para a produção de *games*. Apesar de seu escopo ter sido criado para o desenvolvimento de *Advergames*, os critérios incluídos no processo de produção do AGP podem ser adaptados para a criação de qualquer tipo de *game*. Desses critérios destacam-se o desenvolvimento iterativo e incremental, gerenciamento descentralizado e o enfoque à práticas ágeis. Outra importante característica da metodologia é a presença dos critérios de refinamento do *gameplay* e adaptabilidade para a mudança de requisitos a qualquer instante do projeto. Por ser uma metodologia intermediária, a presença de vários critérios em seu processo de produção (aproximadamente 54% dos critérios) e seus benefícios, acabam por ser ofuscados pela ausência de uma quantidade considerável de critérios (aproximadamente 46%). Dos critérios não pertencentes a metodologia, alguns destacam-se pela sua importância a produção de *games*, como o gerenciamento de riscos, a prototipação, gerenciamento das áreas não programadoras, gerenciamento de tempo e custos, e principalmente, um fluxo de desenvolvimento não linear. Este último, inexistente ao AGP, faz com que a ocorrência de atrasos e extrapolações do orçamento sejam maiores, principalmente no que tange a busca e correção de *bugs* ao final do projeto.

4.3.3 Análise Geral

Com base na Tabela 4.16 e nas Figuras 4.1 e 4.2, é visto que cinco das metodologias, atendem a mais de 50% dos critérios que representam boas práticas em desenvolver *games*. Uma metodologia atende a exatos 50%, e a outra metade das técnicas não possuem nem 40% dos critérios em seus processos de produção. É importante ressaltar que, a metade das metodologias que não possuem 40% dos critérios são processos já defasados quanto ao desenvolvimento de *games* e sua utilização não consegue acompanhar de forma eficiente, os desafios da indústria. As outras seis metodologias (que possuem no mínimo

50% dos critérios) são processos ágeis, baseadas no desenvolvimento iterativo e incremental. Possuem um gerenciamento descentralizado, são focadas na verificação contínua da qualidade do *game*, dão ênfase na otimização da comunicação e, principalmente, são flexíveis a mudanças frequentes de requisitos (com exceção da abordagem GAMA). Dessa forma, consequentemente, essas seis metodologias estão mais preparadas para a produção de *games*, devido a forma como tratam as características da indústria. Esse fato, reforça o que foi apresentado no Capítulo 3, ou seja, que o desenvolvimento profissional de *games* é inerentemente ágil.

Em relação aos critérios que representam boas práticas para a produção de *games*, destacam-se aqueles especificamente ágeis como: Desenvolvimento ágil, desenvolvimento iterativo e incremental, gerenciamento descentralizado, cliente como parte da equipe, suporte a comunicação, entre outros. Esse grupo de critérios estão presentes entre 50% a 83% das metodologias. Isso indica a importância cada vez maior em incluir nas metodologias o desenvolvimento ágil e seus valores. Sendo este um dos principais anseios que a indústria necessita, produzir *games* de forma ágil. Outro grupo de critérios que se destacam são os relacionados a qualidade do *game*: Contínua verificação da qualidade, contínua verificação da qualidade da produção, automação de testes e prototipação. Esses critérios estão menos presentes nas metodologias do que o primeiro grupo, entre 17% a 50%. Isso demostra, uma das principais falhas dos processos de produção de *games*, ou seja, a não priorização da garantia de qualidade. Dessa forma, na maioria das metodologias, ainda impera a já defasada etapa de testes no final da produção, o que, como já mencionado, causa sérios danos a qualidade do *game*.

Outro importante grupo, é o de critérios exclusivos em um processo de desenvolvimento de *games*: Gerenciamento da multidisciplinaridade, flexibilidade, refinamento do *gameplay* e *playtesting*. Esse grupo possui uma grande variação de 0% a 83%, para critérios incluídos nas metodologias. Os critérios de flexibilidade e refinamento do *gameplay* são bastante recorrentes entre as metodologias. O que indica uma preocupação de quase todas as técnicas com o conteúdo do *game* e a característica peculiar da indústria, em alterar ou adicionar requisitos a qualquer momento do projeto. Já os critérios, *playtesting* e gerenciamento da multidisciplinaridade, não pertencem a nenhuma metodologia

de produção de *games*. Isso indica, o preocupante despreparo das metodologias atuais em lidar com as diferentes áreas do desenvolvimento de *games* e também, na especialização da garantia de qualidade, em relação aos testes com potenciais jogadores, com o intuito de aumentar a qualidade do *game* produzido.

O último grupo a ser destacado, é o dos critérios gerenciais: Gerenciamento da concepção do *game*, gerenciamento específico do processo de *Game Design*, gerenciamentos de tempo, custos e riscos, e o constante monitoramento do projeto. Esse grupo possui uma baixa aderência entre as metodologias, variando de 0% a 42%. Os critérios de gerenciamento de riscos e o constante monitoramento do projeto são os mais presentes nas metodologias. Contudo, suas inclusões entre as técnicas, respectivamente em 33% e 42%, ainda são porcentagens baixas, em relação ao grau de importância que esses critérios apresentam aos processos de produção de *games*. É importante destacar que, a baixa porcentagem dos critérios de gerenciamento de custos e de tempo (8% e 17%), comprova o aumento cada vez maior dos atrasos de cronograma e dos custos do projeto. Entretanto, o que mais se destaca nesse grupo, é a inexistência dos critérios de gerenciamento da concepção do *game* (com exceção da abordagem GAMA) e do processo de *Game Design*. Esses critérios são importantes, pois representam a essência do *game*, ou seja, o que irá garantir o lucro da empresa com sua venda. Um específico gerenciamento desses critérios potencializará a criação e desenvolvimento de conteúdos de boa qualidade para o *game*, aumentará a visão da equipe sobre o projeto e, contribuirá para a diminuição dos atrasos no cronograma e períodos de *crunch times*.

4.4 Modelo de Referência para a Escolha de Metodologias

De acordo com Bethke (2003) e Chandler (2012), a produção de *games* pode ser dividida em três tipos: Baixa complexidade, média complexidade e alta complexidade. Uma produção de baixa complexidade desenvolve *games* mais simples, com poucos recursos financeiros e tempo inferior a outros tipos de produções. Dois exemplos a serem

citados, são os *games* casuais[17], em particular, os *games* Angry Birds e Candy Crush. Já uma produção de média complexidade desenvolve *games* que não são tão simples, quanto os *games* casuais. Contudo, não estão relacionados a grandes produções de *games*. Em geral, produções de média complexidade possuem orçamentos e cronogramas superiores à produções de baixa complexidade. Dois exemplos a serem citados, são os *games* Limbo e Horizon Chase. E por fim, uma produção de alta complexidade desenvolve *games* AAA, ou seja, *games* com histórias profundas, mecânicas complexas, recursos gráficos de última geração, com orçamentos milionários e tempo de produção que passam de dois anos. Dois exemplos que podem ser citados, são os *games* God of War e The Last of Us, resultado de produções altamente complexas.

Para reforçar os resultados alcançados pela análise comparativa (principalmente no que tange à descoberta das metodologias mais indicadas ao desenvolvimento de *games* para o entretenimento), é importante destacar também as metodologias mais apropriadas aos três tipos de produções de *games* existentes na indústria. Para isso, é utilizado uma árvore de decisão que possui o intuito de auxiliar a descoberta das metodologias mais indicadas em relação aos diferentes tipos de produções de *games*.

Cada item presente na árvore de decisão (complexidade, orçamento, equipe, tempo e *playtest*), advém da sua importância fundamental para o desenvolvimento de *games*. Esses elementos são tão importantes que, caso sejam mal gerenciados, podem gerar a inviabilidade de um projeto, como mencionado nos Capítulos 2 e 3. Além disso, problemas relacionados a esses itens, ocorrem de forma frequente nos *Post-mortems* disponíveis pelas empresas. É importante ressaltar ainda que, dos itens da árvore de decisão, apenas o *playtest* pode ser descartado (devido a seu custo) para projetos de pequena ou média complexidade, sem comprometer a qualidade final do *game*. A Tabela 4.17, ilustra os itens, descrições e valores presentes na árvore de decisão (Taylor et al., 2007; Petrillo, 2008; Chandler, 2012; Mirabello, 2014).

Itens	**Descrição**	**Valores**
PG	Projeto de *games*.	

[17] *Games* jogados por pessoas que dispõem de pouco tempo para jogar (Chacon, 2015). Geralmente, são *games* com enredos inexistentes ou superficiais, possuem poucos recursos gráficos e jogabilidade simples.

Itens	Descrição	Valores
Complexidade	Tamanho da produção de um *game*.	CP1 (alta complexidade), CP2 (média complexidade) e CP3 (baixa complexidade).
Orçamento	Custo total para a produção de um *game*.	O1 (orçamento alto), O2 (orçamento médio) e O3 (orçamento baixo).
Equipe	Tamanho da equipe de desenvolvimento para a produção de um *game*.	E1 (equipe com uma grande quantidade de pessoas), E2 (equipe média) e E3 (equipe com uma baixa quantidade de pessoas).
Tempo	Tempo médio para a produção de um *game*.	T1 (superior a 2 anos), T2 (de 1 a 2 anos) e T3 (até 1 ano).
Playstest	Presença do processo de *playtest* na produção de um *game*.	P1 (Possui um processo de *playtest*) e P2 (Não possui um processo de *playtest*).

Tabela 4.17: Legenda para os itens presentes na árvore de decisão.

A Figura 4.3, apresenta a árvore de decisão com os possíveis caminhos para o desenvolvimento de *games*, no que tange a escolha mais apropriada em relação às características do tipo de produção que se tem. É importante destacar que, a árvore de decisão da Figura 4.3, reflete apenas situações onde é possível finalizar um projeto com sucesso, ou seja, apto a lançar o *game* no mercado. Situações que geram projetos *Death Match*[18] não são expostos nesta árvore de decisão. Além disso, a abordagem utilizada não apresenta casos em que se tem folgas de produção. Como por exemplo: Tem-se um orçamento de 4 milhões de dólares mas, o produtor irá gastar apenas 300 mil dólares para desenvolver o *game*. Ou ainda, Tem-se tempo e pessoas para desenvolver um *game* AAA contudo, decide-se desenvolver um *game* casual. É importante destacar que, essas folgas,

[18]Projetos fadados ao fracasso, ou seja, não podem ser finalizados.

além de serem um desperdício de recursos da empresa, são falhas de planejamento. Outra motivação para situações de folgas de produção não aparecerem na árvore de decisão é que, esses casos são raros na indústria de *games* (que trabalha sempre com prazos e custos no limite da produção). Adicionado a isso, nenhum *Post-mortem* apresentado pelas referências pesquisadas neste trabalho, mencionam folgas de produção.

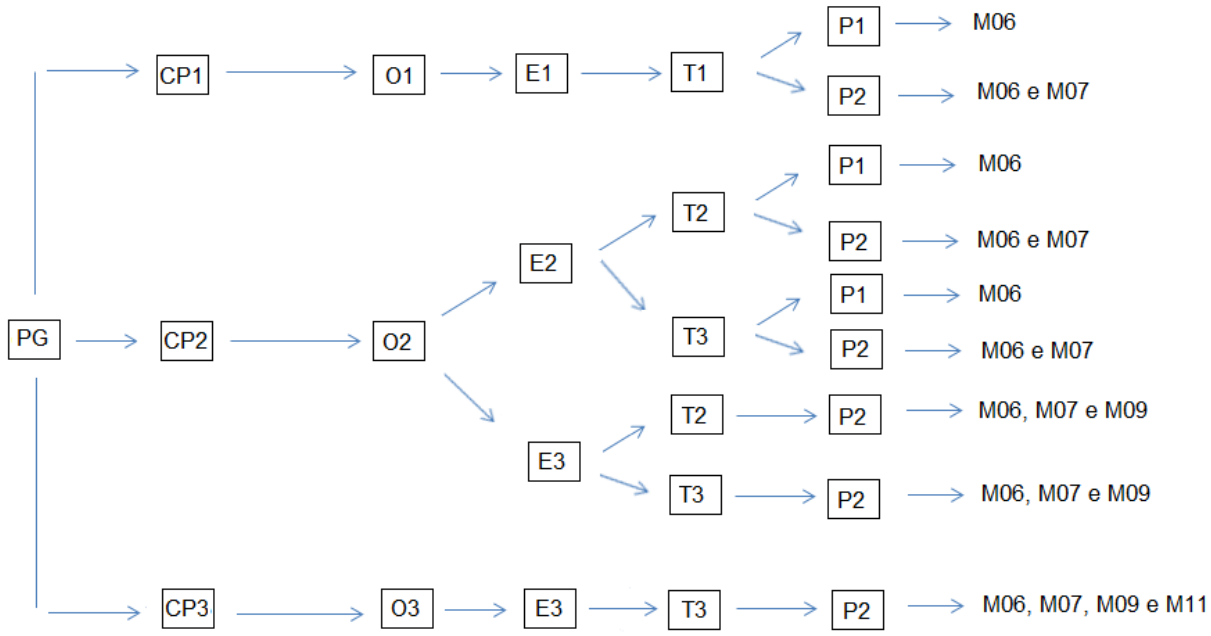

Figura 4.3: Árvore de decisão para os diferentes tipos de produções de *games*.

Com base na Figura 4.3, será apresentado três exemplos de *games* utilizando o modelo de referências para a escolha de metodologias. Cada *game*, desenvolvido por um tipo de produção diferente, em relação ao tamanho do projeto e, ao tipo de *game* a ser produzido (*game* casual, AAA, ou nenhum dos dois).

1. **Primeiro Exemplo:**

   1.1. Descrição: Deseja-se produzir um *game* com a proposta de ser casual e para celular, com um orçamento de 50 mil dólares, tempo médio de produção de 10 meses e com 8 pessoas na equipe de desenvolvimento.

   1.2. O caminho percorrido por este *game* na árvore de decisão da Figura 4.3 é: $PG > CP3 > O3 > E3 > T3 > P2$.

   1.3. Resultado: As metodologias indicadas para esse tipo de *game*, com uma produção de baixa complexidade, são as técnicas M06 ou M07 ou M09 ou M11.

2. **Segundo Exemplo:**

   2.1. Descrição: Será produzido um *game* com a proposta de ser *hardcore*[19] para um projeto AAA. O projeto deste *game* possui um orçamento de 2 milhões de dólares e um tempo médio de produção de 3 anos. A equipe de desenvolvimento para esse *game* possui 90 pessoas (programadores, animadores, artistas, músicos, *designers*, entre outros). Os produtores desse projeto optaram por trabalhar com um processo de *playtest* para melhorar a qualidade do *game*.

   2.2. O caminho percorrido por este *game* na árvore de decisão da Figura 4.3 é: $PG > CP1 > O1 > E1 > T1 > P1$.

   2.3. Resultado: A metodologia apropriada para esse tipo de *game*, com uma produção de alta complexidade, é a técnica M06.

3. **Terceiro Exemplo:**

   3.1. Descrição: Tomando desta vez, uma empresa independente, que deseja produzir um *game* de aventura, com uma intrigante história e várias fases. Contudo, a empresa não possui o orçamento de um projeto AAA. Seu orçamento total é de 800 mil dólares, com um cronograma geral, estimado em 1 ano e 2 meses. A empresa possui uma pequena equipe de 15 desenvolvedores. Para diminuir os custos de produção, o produtor decidiu retirar o processo de *playtest* para o desenvolvimento do *game*.

   3.2. O caminho percorrido por este *game* na árvore de decisão da Figura 4.3 é: $PG > CP2 > O2 > E3 > T2 > P2$.

   3.3. Resultado: As metodologias adequadas para esse tipo de *game*, com uma produção de complexidade nem baixa nem alta, são as técnicas M06 ou M07 ou M09.

É importante ressaltar que, as motivações pelos quais as demais técnicas apresentadas nesta monografia, não fazem parte da árvore de decisão da Figura 4.3, são as

[19]São *games* focados em enredos profundos, jogabilidade complexa e, geralmente difíceis. Seu público-alvo são jogadores habilidosos, experientes e que anseiam por muitas horas de *gameplay* (Chacon, 2015).

características incompatíveis com o desenvolvimento comercial de *games*, e grande possibilidade de serem gerados projetos *Death Math*. Essas características negativas são: Fluxo de desenvolvimento sequencial e linear, e/ou possui uma etapa de testes exclusiva e é executada no final da produção do *game*, e/ou a metodologia não é ágil, e/ou não avalia o *gameplay* do *game* durante todo o projeto (apenas no final da produção).

5 Conclusão

A indústria de *games* tornou-se uma das mais lucrativas com o passar dos anos. Contudo, sua evolução trouxe à indústria, desafios inexistentes em seus primórdios, a serem superados por qualquer empresa que produza *games*. Esses desafios, em geral, foram gerados devido ao aumento substancial da estrutura de um *game* (arte e *software*), demandando uma grande quantidade de profissionais e de diferentes áreas de atuação (não mais, apenas programadores como nas décadas de 70 e 80). Acarretando dessa forma, em grandes investimentos (chegando a milhões de dólares), e riscos cada vez mais intensos à produção de *games* para o entretenimento.

5.1 Considerações Finais

O objeto de estudo da presente monografia; o desenvolvimento profissional de *games*, de forma eficiente e adaptando-se às necessidades da indústria, gerando um *game* de qualidade; foi apresentado através do estudo, comparação e análise das metodologias de desenvolvimento existentes. Foi possível também, compreender que entre os problemas da indústria, o principal é a dificuldade no gerenciamento da produção de *games*, devido a sua peculiar característica de multidisciplinaridade e, a complexidade de um projeto desse tipo. Dessa forma, torna-se crucial a utilização de metodologias de desenvolvimento de *softwares*, devidamente ajustadas à projetos de *games* e aos problemas da indústria. Durante a presente monografia, foram encontradas doze metodologias, mais relevantes, em relação ao cenário profissional da produção de *games*. Dentre os doze modelos identificados, os mais utilizados pela indústria são aqueles que possuem como base o Modelo em Cascata ou processos de desenvolvimento ágil de *software* (em especial, baseados nas técnicas *Scrum* e XP).

É importante destacar que, as metodologias ágeis são importantes para a produção de *games*, em geral, devido ao tratamento da principal característica da indústria, ou seja, mudanças de requisitos a qualquer instante, quantas vezes forem necessárias, e com

a mínima perda de tempo e custo para o projeto. De fato, o desenvolvimento de *games*, é especificamente ágil. Outro importante fator a ser destacado, é o valor que a Engenharia de *Software* representa à indústria de *games*. Sendo inviável comercialmente, produzí-los sem o auxílio de processos e práticas já consolidadas da Engenharia de *Software* como: Utilização de modelos, engenharia de requisitos, validação de *software*, gerenciamentos (principalmente de riscos, custos, qualidade e tempo), entre outros.

Os resultados obtidos através da comparação entre as metodologias de desenvolvimento, intensificam a premissa de que o uso dessas técnicas são obrigatórias para uma produção comercial de *games*. Esta comparação está relacionada a critérios (tais como agilidade, ênfase em QA, flexibilidade quanto aos requisitos, entre outros) que representam boas práticas necessárias ao desenvolvimento de *games*, devido a adaptação, otimizada, aos desafios da indústria, de forma a sempre diminuí-los. Aliado a isso, após a comparação, pode-se dividir as metodologias em três grupos distintos.

O primeiro grupo, representado por 50% das técnicas pesquisadas, inclui metodologias já defasadas quanto a produção de *games*. Estas se mostram obsoletas à indústria, devido a quantidade considerável de critérios não atendidos, diante a tabela de comparação (aproximadamente, entre 67% a 92% de critérios não atendidos). Portanto, estas metodologias não estão adaptadas à indústria de *games*, devido a incapacidade de tratar, de forma eficiente, seus desafios. A utilização dessas metodologias, exceto por fins acadêmicos, devem ser evitadas. Esses modelos são: GWP, Modelo proposto por Sid Meier, Modelo proposto por Ed Logg, Modelo proposto por Rollings e Morris, RGP e GUP.

O segundo grupo, representado por aproximadamente 17% das técnicas pesquisadas, inclui metodologias intermediárias, ou seja, não são defasadas como o primeiro grupo porém, não são totalmente adaptadas à indústria. As metodologias SUM e AGP preenchem este grupo, e apresentam respectivamente 50% e 46% de critérios não atendidos, aproximadamente. São modelos, que apesar de mais evoluídos do que os do primeiro grupo, não são indicados para grandes projetos profissionais de *games*, devido a pouca quantidade de critérios atendidos e, como consequência, a possibilidade maior de cancelamentos de projetos, lançamentos de *games* com *bugs* ou com uma qualidade a baixo do

esperado.

O terceiro e último grupo, representado por aproximadamente 33% das técnicas pesquisadas, inclui as metodologias mais adaptadas à produção de *games*. Esses modelos atendem a maioria dos critérios de comparação, necessários a uma produção otimizada e eficiente de *games*. A metodologia mais adaptada, através da tabela de comparação, é o processo *Game Scrum* (com 71% de critérios atendidos, aproximadamente), seguido pelo *AgiGame* e GAMA (empatados com aproximadamente 67%), e por último entre as metodologias do grupo, a técnica XGD (com aproximadamente 63% de critérios atendidos). Assim sendo, as metodologias do terceiro grupo, em particular o *Game Scum*, são mais completas, ou seja, estão mais preparadas para a realização de projetos de *games* comerciais, enfrentando os problemas da indústria de forma eficaz e eficiente, aumentando cada vez mais as possibilidades de produzir *games* de boa qualidade, e sem desperdícios, principalmente em relação ao tempo e orçamento do projeto.

Os resultados apresentados com o auxílio do modelo de referência, no final do Capítulo 4, refletem as metodologias mais completas vistas na análise comparativa. Esses resultados, reforçam o exposto anteriormente, que as quatro técnicas mais apropriadas à produção de *games* para o entretenimento são os modelos *Game Scum*, XGD, *AgiGame* e GAMA. Dessa forma, através do resultado final da análise comparativa, aliado ao conjunto de metodologias vistas na Figura 4.3, observa-se empiricamente que:

A metodologia *Game Scum* aparece em todas as possibilidades da árvore de decisão, para os três tipos de produções de *games*. Para projetos de grande porte, sua utilização é recomendada devido a grande quantidade de boas práticas presentes em sua estrutura, vistas na tabela de comparação. E ainda pelo fato, de acordo com Keith (2010), de que o *Game Scum* já é utilizado profissionalmente. Apesar de não possuir um processo de *playtest*, pode ser adaptado à metodologia, pela presença de um forte gerenciamento de riscos em sua estrutura. E com isso, minimizando ou eliminando os problemas causados pelo acréscimo do processo de *playtest*, caso ocorram. Já para projetos de média e baixa complexidade, o *Game Scum* é indicado pois, como é uma metodologia ágil, ela funciona bem com um número reduzido de pessoas e com um cronograma limitado em relação às atividades do projeto.

A metodologia XGD aparece em quase todas as possibilidades da árvore de decisão, para os três tipos de produções de *games*. Assim como o *Game Scrum*, o XGD é recomendado para projetos de grande complexidade, devido a grande quantidade de boas práticas presentes em sua estrutura, vistas na tabela de comparação 4.16. Além de, segundo Demachy (2003) e Brauwers (2011), já ser usado na indústria de *games*. O XGD só não é recomendado a projetos de grande porte, quando há a necessidade do uso de *playtest*. Pois, a técnica XGD não possui em sua estrutura o processo de *playtest* e tão pouco, um gerenciamento de riscos para incluir, de forma segura, esse tipo de teste. A metodologia XGD, também é recomendada à projetos de médio e pequeno porte, devido a sua característica de agilidade. E como consequência, essa técnica está preparada para auxiliar o gerenciamento de um projeto com poucos profissionais e prazos curtos, com o intuito de gerar um *game* de boa qualidade.

A metodologia *AgiGame* é vista apenas em três situações da árvore de decisão. Para projetos de média complexidade (equipe reduzida e tempo médio de produção ou, equipe reduzida e tempo reduzido de produção) e projetos de baixa complexidade. Isso se deve ao fato de, apesar do *AgiGame* ser uma técnica promissora, contendo importantes boas práticas para o desenvolvimento de um *game* de boa qualidade, ainda é uma técnica nova e necessita ainda de mais estudos. Assim, não é recomendado o seu uso em projetos de alta complexidade ou, projetos de médio porte com uma equipe, também de médio porte. Segundo Posvolski et al. (2014), os autores do *AgiGame* recomendam seu uso para equipes pequenas. Portanto, o modelo *AgiGame* é recomendado apenas a projetos de média complexidade (com uma pequena equipe) e também, projetos de baixa complexidade.

A metodologia GAMA é incluída em apenas uma situação da árvore de decisão, para projetos de baixa complexidade. A técnica GAMA é inovadora, pois foi criada com o intuito de superar os particulares desafios da indústria e, é uma das quatro técnicas com a maior quantidade de critérios que representam boas práticas em desenvolver *games*. Contudo, assim como a metodologia *AgiGame*, o GAMA é um modelo novo e que também necessita de mais pesquisas. Ainda assim, o que torna o GAMA menos apropriado se comparado ao *AgiGame*, é a falta de um processo de gerenciamento de riscos e (em destaque),

a falta de uma estrutura de flexibilidade (ou seja, não é seguro no GAMA acrescentar ou alterar ou excluir *features* a qualquer momento do projeto). Dessa forma, não é recomendado o seu uso em projetos de alta ou média complexidade. Portanto, comercialmente, a metodologia GAMA é indicada somente a projetos de baixa complexidade.

É importante ressaltar que, não existem metodologias perfeitas, mesmo as mais adaptadas possuem desvantagens que podem prejudicar um projeto, gerando atrasos ou ainda, *games* com uma qualidade ruim. Por mais completa que seja uma metodologia, em relação a quantidade de critérios de boas práticas que ela possui, o grau de comprometimento e competência das pessoas envolvidas no projeto são cruciais para que o *game* seja bem sucedido no mercado. Entretanto, uma boa metodologia tem o potencial de canalizar o talento dos profissionais envolvidos no projeto, guiando o produtor a realizar um gerenciamento eficiente. E dessa forma, aumentando a possibilidade de ser desenvolvido um *game* de excelente qualidade e que agrade a maior quantidade possível de pessoas.

Portanto conclui-se que, o desenvolvimento de *games* é uma área da computação que, obrigatoriamente não deve ser separada da Engenharia de *Software*, aproveitando de seus processos e práticas já consolidadas. Um desses processos, fundamentais para a produção de bons *games* é a utilização de metodologias de desenvolvimento de *softwares*. Em especial, para a criação profissional de *games*, metodologias de caráter ágil. A preocupação em produzir *games* de forma correta, ou seja, sempre focado na qualidade, nunca deve ser deixada de lado. De modo a perpetuar por muitos anos, a cultura dos *games* entre as gerações futuras, através de boas práticas gerenciais, e sempre destacando a qualidade em primeiro lugar.

5.2 Projetos Futuros

Uma possível sugestão para projetos futuros, é a continuação, prática, da presente monografia. Ou seja, propõe-se destacar as quatro metodologias (levantadas neste trabalho) mais adaptadas à produção comercial de *games*. A partir de então, realiza-se o desenvolvimento de quatro *games* diferentes, cada um produzido por uma das quatro metodologias fixadas. Cada *game*, deverá ser desenvolvido pela mesma equipe. Por tratar-se de um projeto acadêmico, e a limitação em relação a disponibilidade de pessoas envolvidas

na pesquisa, recomenda-se a utilização de cinco tipos de profissionais, característicos da indústria de *games*. São eles: Um Produtor (o autor do projeto), um *Game Designer* (caso não haja uma pessoa específica, sugere-se o próprio autor do projeto), um artista, um programador e um *tester*. Durante os projetos, deve-se persistir fatores relacionados aos desafios da indústria, como atrasos de prazos, dificuldades de comunicação entre os membros da equipe, dificuldades de gerenciamento, flexibilidade dos requisitos, qualidade do *game*, entre outros. Ao final dos projetos, deve-se realizar comparações sobre os resultados específicos entre a produção de cada *game* com as diferentes metodologias.

Outra sugestão, consiste em adicionar novas metodologias ou ainda novas extensões de metodologias já existentes à pesquisa. Em seguida, refinar os atuais critérios de comparação. E por fim, realizar uma nova comparação entre as metodologias levantadas na pesquisa.

Referências Bibliográficas

Acerenza, N.; Coppes, A.; Mesa, G.; Fernandez, A.; Laurenzo, T. ; Vallespir, D. **Una metodologia para desarrollo de videojuegos**. In: Proceedings of the 10th Argentine Symposium on Software Engineering, Argentina, 2009.

Albino, R.; Souza, C. ; Prado, E. Benefícios alcançados por meio de um modelo de gestão Ágil de projetos em uma empresa de jogos eletrônicos. **Revista de Gestão e Projetos**, v.5, n.1, p. 15–27, 2014.

Ampatzoglou, A.; Chatzigeorgiou, A. Evaluation of object-oriented design patterns in game development. **Information and Software Technology**, v.49, n.5, p. 445–454, 2007.

Ampatzoglou, A.; Stamelos, I. Software engineering research for computer games - a systematic review. **Information and Software Technology**, v.52, n.9, p. 888–901, 2010.

Araujo, A. **Agile Game Process: Metodologia Ágil para Projetos de Advergames**. Pernambuco, 2006. Trabalho de graduação - Ciência da Computação, Universidade Federal de Pernambuco.

GEDIGames. **Relatório Final: Mapeamento da Indústria Brasileira e Global de Jogos Digitais**. Technical report, BNDS, São Paulo, Fevereiro 2014.

Barros, R. **Análise de Metodologias de Desenvolvimento de Software aplicadas ao Desenvolvimento de Jogos Eletrônicos**. Pernambuco, 2007. Trabalho de graduação - Ciência da Computação, Universidade Federal de Pernambuco.

Barros, T. **O que é DLC**, 2014. Disponível em: http://www.techtudo.com.br/dicas-e-tutoriais/noticia/2014/01/o-que-e-dlc-veja-a-historia-dos-conteudos-extras-para-jogos.html. [Acessado em 10 março 2016].

Bethke, E. **Game Development and Production**. 1º edição, Estados Unidos: Editora Wordware Publishing, 2003.

Brathwaite, B.; Schreiber, I. **Challenges for Game Designers**. 1º edição, Estados Unidos: Editora Course Technology, 2009.

Brauwers, R. **Estendendo e instanciando o game agile methods applied (gama)**. Porto Alegre, 2011. Trabalho de graduação - Ciência da Computação, Universidade Federal do Rio Grande do Sul.

Carmona, S. **O museu de game como experiência gamificada**. São Paulo, 2012. Dissertação de Mestrado - Pontifícia Universidade Católica de São Paulo.

Carvalho, E. **Os 5 Desprezíveis - Desenvolvimento de um Jogo Eletrônico Utilizando os Princípios de Engenharia de Software**. Brasília, 2013. Trabalho de graduação - Bacharelado em Engenharia de Software, Universidade de Brasília.

Carnasciali, R. **Metodologia ágil e a gestão da comunicação em projetos de games digitais**. In: Proceedings of the 13th Symposium Brazilian of Games and Digital Entertainment, Porto Alegre, 2014.

Chagas, M. **A Inserção do Designer de Games na Indústria Brasileira de Jogos Eletrônicos**. Rio de Janeiro, 2009. Tese de Doutorado - Pontifícia Universidade Católica.

Chandler, H. **Manual de Produção de Jogos**. 2º edição, São Paulo: Editora Bookman, 2012.

Chacon, A. **Jogos Casuais e Hardcore**, 2015. Disponível em: http://www.fabricadejogos.net/posts/artigo-jogos-casuais-e-hardcore/. [Acessado em 7 novembro 2016].

Cruz, A.; Garone, P. **A formação do conceito de um jogo: Estudo de processos metodológicos para a criação de um game**. In: Proceedings of the 12th Symposium Brazilian of Games and Digital Entertainment, São Paulo, 2013.

Davis, B. **What are game assets**, 2009. Disponível em: http://conceptdevelopmentbendavis.blogspot.com.br/2009/02/what-are-game-assets.html. [Acessado em 10 março 2016].

Demachy, T. **Extreme Game Development: Right on Time, Every Time**, 2003. Disponível em: http://www.gamasutra.com/resource_guide/20030714/demachy_pfv.htm. [Acessado em 23 Setembro 2015].

Freitas, E. **UFOQX: Desenvolvimento de um Jogo 2D para Plataforma Android**. Quixadá, 2014. Trabalho de graduação - Sistemas de Informação, Universidade Federal do Ceará.

Galeote, S. **Metodologia de desenvolvimento de software: Importância, conceitos e princípios**, 2010. Disponível em: http://www.galeote.com.br/blog/2010/02/metodologia-de-desenvolvimento-de-software-importncia-conceitos-e-princpios/. [Acessado em 02 Outubro 2015].

Gervazoni, T. **Importância de metodologia no desenvolvimento**, 2005. Disponível em: http://www.linhadecodigo.com.br/artigo/595/importancia-de-metodologia-no-desenvolvimento.aspx. [Acessado em 02 Outubro 2015].

Godoy, A.; Barbosa, E. **Game-scrum: An approach to agile game development**. In: Proceedings of the 9th Symposium Brazilian of Games and Digital Entertainment, Florianópolis, 2010.

Johnson, S. **Sid's Rules**, 2009. Disponível em: http://www.designer-notes.com/?p=119. [Acessado em 24 Setembro 2015].

Kabashi, A.; Saabi, H. **Game software processes - with focus on the rapid game process (rgp)**. Suécia, 2008. Dissertação de Mestrado - School of Engineering - Blekinge Institute of Technology.

Kasperavicius, L.; Bezerra, L.; Silva, L. ; Silveira, I. **Ensino de desenvolvimento de jogos digitais baseado em metodologias Ágeis: O projeto primeira habilitação**. In: Proceedings of the 28th Congress of the Brazilian Computer Society on Workshop about Education in Computing, Belém do Pará, 2008.

Keith, C. Scrum rising - agile development could save your studio. **Game Developer**, v.14, n.2, p. 21–26, 2007.

Keith, C. **Waterfall Game Development**, 2009. Disponível em: http://blog.agilegamedevelopment.com/2009/01/in-dawn-of-video-game-development.html. [Acessado em 21 Setembro 2015].

Keith, C. **Agile Game Development with Scrum**. 1º edição, Estados Unidos: Editora Pearson, 2010.

Keith, C. **Agile Game Development with Scrum**, 2010. Disponível em: http://www.informit.com/articles/article.aspx?p=1594851. [Acessado em 10 Outubro 2015].

Kitchenham, B. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. Technical report, EBSE Technical Report, Inglaterra, Julho 2007.

Lavor, R. **Metodologia Utilizada no Desenvolvimento de Games**. São Paulo, 2009. Trabalho de graduação - Tecnologia em Informática para a gestão de negócios, Faculdade de Tecnologia da Zona Leste.

Leite, A. **Metodologia de Desenvolvimento de Software**, 2006. Disponível em: http://www.devmedia.com.br/metodologia-de-desenvolvimento-de-software/1903. [Acessado em 02 Outubro 2015].

Lemes, D. **Games independentes. fundamentos metodológicos para criação, planejamento e desenvolvimento de jogos digitais**. São Paulo, 2009. Dissertação de Mestrado - Pontifícia Universidade Católica de São Paulo.

Luz, M. **Desenvolvimento de Jogos de Computador**. Itajubá, 2004. Trabalho de graduação - Ciência da Computação, Universidade Federal de Itajubá.

Machado, T. **Guidelines Para a Criação de Jogos - Boas Práticas Para Reduzir Conflitos Entre o Design e o Desenvolvimento**. Pernambuco, 2009. Trabalho de graduação - Ciência da Computação, Universidade Federal de Pernambuco.

McEntee, C. **Rational Design: The Core of Rayman Origins**, 2012. Disponível em: http://www.gamasutra.com/view/feature/167214/rational_design_the_core_of_.php. [Acessado em 21 Setembro 2015].

McGuire, R. **Game Design With Agile Methodologies**, 2006. Disponível em: http://www.gamasutra.com/view/feature/2742/paper_burns_game_design_with_.php. [Acessado em 8 Setembro 2015].

Medeiros, M.; Campos, F.; Benicio, I. ; Neves, A. **A importância da prototipação no design de games**. In: Proceedings of the 12th Symposium Brazilian of Games and Digital Entertainment, São Paulo, 2013.

Medeiros, H. **Introdução a Requisitos de Software**, 2013. Disponível em: http://www.devmedia.com.br. [Acessado em 5 abril 2016].

Mikoluk, K. **Agile Vs. Waterfall: Evaluating The Pros and Cons**, 2013. Disponível em: https://blog.udemy.com/agile-vs-waterfall/. [Acessado em 23 Setembro 2015].

Mirabello, J. **How Long Does It Take to Make an Indie Game ?**, 2014. Disponível em: http://www.gamasutra.com/blogs/JosephMirabello/20140407/214931/How-Long-Does-It-Take-to-Make-an-Indie-Game.php. [Acessado em 7 novembro 2016].

Nakano, D.; Nakamura, R. ; Sakuda, L. **Produção e operações em games: Visão geral e perspectivas**. In: Proceedings of the 11th Symposium Brazilian of Games and Digital Entertainment, Brasília, 2012.

OHagan, A.; Coleman, G. ; OConnor, R. **Software development processes for games: A systematic literature review**. In: Proceedings of the 21th European Conference on Systems, Software and Services Process Improvement, Alemanha, 2011.

Ollila, E.; Suomela, R. ; Holopalnen, J. Using prototypes in early pervasive game. **Computers in entertainmen**, v.6, n.2, p. 9, 2008.

Park, J. Y.; Park, J. H. A graph based representation of game scenarios - methodology for minimizing anomalies in computer game. **The Visual Computer**, v.26, n.6, p. 595–605, 2010.

Perani, L. **Game Studies Brasil: Um Panorama dos Estudos Brasileiros Sobre Jogos Eletrônicos**. Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2008.

Petrillo, F. **Práticas ágeis no processo de desenvolvimento de jogos eletrônicos**. Porto Alegre, 2008. Dissertação de Mestrado - Universidade Federal do Rio Grande do Sul.

Posvolski, A.; Torres, I.; Concilio, I. ; Pacheco, B. **Agigame: Proposta de uma metodologia híbrida para desenvolvimento de jogos**. In: Proceedings of the 13th Symposium Brazilian of Games and Digital Entertainment, Porto Alegre, 2014.

Pressman, R. **Engenharia de Software Uma Abordagen Profissional**. 7º edição, São Paulo: Editora Bookman, 2011.

Preisz, E. **Waterfall Game Development Done Right**, 2012. Disponível em: http://www.gamasutra.com/view/feature/181992/waterfall_game_development_done_.php. [Acessado em 21 Setembro 2015].

Reis, A.; Nassu, B. ; Jonack, M. **Um Estudo Sobre os Processos de Desenvolvimento de Jogos Eletrônicos (Games)**. Universidade Federal do Paraná, Paraná, 2002.

Rogers, S. **Level Up: Um Guia para o Design de Grandes Jogos**. 1º edição, São Paulo: Editora Blucher, 2013.

Rollings, A.; Morris, D. **Game Architecture and Design: A New Edition**. 1º edição, Estados Unidos: Editora New Riders, 2004.

Rouse, R. **Game Design: Theory and Pratice**. 2º edição, Estados Unidos: Editora Wordware Publishing, 2005.

Sales, R. **Proposta de método para gestão ágil da visão no desenvolvimento de jogos digitais**. In: Proceedings of the 12th Symposium Brazilian of Games and Digital Entertainment, São Paulo, 2013.

Santana, R. **I.A. Em Jogos – A Busca Competitiva entre o Homem e a Máquina**. Praia Grande, 2006. Trabalho de graduação - Informática com ênfase em Gestão de Negócios, Faculdade de Tecnologia de Praia Grande.

Santos, V.; Correia, J. **Adaptação da metodologia jad para o desenvolvimento de games**. In: Proceedings of the 9th Journey of Teaching, Research and Extension, Recife, 2009.

Sato, A. **Game design e prototipagem: Conceitos e aplicações ao longo do processo projetual**. In: Proceedings of the 9th Symposium Brazilian of Games and Digital Entertainment, Florianópolis, 2010.

Silva, P. **Modelação procedimental para desenvolvimento de jogos de computador**. Portugal, 2010. Dissertação de Mestrado - Faculdade de Engenharia da Universidade do Porto.

Sloper, T. **A Glossary of Game Biz Terms**, 2015. Disponível em: http://www.sloperama.com/advice/lesson28.htm. [Acessado em 10 março 2016].

Slyke, B. **How a Game Gets Made A Game's Journey from Concept to Store Shelves**, 2009. Disponível em: http://www.gamecareerguide.com/features/745/how_a_game_gets_made_a_games_.php?page=1. [Acessado em 24 Setembro 2015].

Sommerville, I. **Engenharia de Software**. 8º edição, São Paulo: Editora PEARSON, 2007.

Stateri, J. **O uso da metodologia iterativa na criação de videogames como sistemas emergentes**. In: Proceedings of the 12th Symposium Brazilian of Games and Digital Entertainment, São Paulo, 2013.

M. Taylor, M. Baskett, D. H.; Wade, S. Using soft systems methodology for computer game design. **Systems Research and Behavioral Science**, v.24, n.3, p. 359–368, 2007.

Valin, A. **Conheça Ralph Baer: O Inventor do Video Game**, 2013. Disponível em: http://www.tecmundo.com.br/video-game-e-jogos/37988-conheca-ralph-baer-o-inventor-do-video-game.htm. [Acessado em 03 Outubro 2015].

Velasquez, C. **Modelo de engenharia de software para o desenvolvimento de jogos e simulações interactivas**. Portugal, 2009. Dissertação de Mestrado - Universidade Fernando Pessoa.