AgiGame: Proposta de uma Metodologia Híbrida para Desenvolvimento de Jogos

Ariel S. Posvolski Igor V. Torres Ilana A. Souza-Concilio Beatriz A. Pacheco

Universidade Presbiteriana Mackenzie, Faculdade de Computação e Informática,
São Paulo - SP - Brasil

Resumo

O processo de desenvolvimento de jogos eletrônicos é uma tarefa complexa, que envolve custos elevados. A escolha de uma metodologia adequada e sua correta aplicação pode determinar o sucesso ou o fracasso do projeto. Na literatura atual, faltam metodologias e técnicas específicas ao desenvolvimento de jogos. No decorrer deste trabalho, por meio da compilação de dados, foi realizado um levantamento das metodologias ágeis que melhor se adaptam ao desenvolvimento de jogos e, a partir delas, propõe-se uma metodologia híbrida e específica para este fim. Inicialmente, foi realizado um estudo das metodologias já existentes e consolidadas no mercado: Scrum, FDD (Feature Drive Development) e Running Lean, visando levantar as características fortes de cada uma que possam ser aproveitadas no desenvolvimento de jogos. Ao longo deste artigo será mostrado porque esses recursos de cada uma das metodologias base são adequados ao desenvolvimento de jogos. Essa compilação de dados resulta na metodologia híbrida AgiGame, objetivo deste trabalho. Para finalizar a AgiGame tem seu funcionamento e fases cuidadosamente detalhadas e cada característica relacionada com a característica da metodologia base que foi usada para chegar no seu desenvolvimento final.

Palavras-chaves: Desenvolvimento de jogos eletrônicos, Metodologia híbrida, Metodologia ágil

Authors' contact:
ariel.posvolski@hotmail.com
igor.v.torres@gmail.com
iasouza@mackenzie.br
bia.pacheco@mackenzie.br

1. Introdução

Os jogos, sejam eles de tabuleiro, cartas ou eletrônicos são opções de lazer individual ou coletivo. Segundo Aguilera e Mendiz [1], os jogos eletrônicos desenvolvem a leitura, o pensamento lógico, a observação, a localização e representação espacial, a resolução de problemas e tomadas de decisão, por meio do planejamento de estratégias.

Segundo Kent [2], em 1958 William Higinbotham criou um jogo denominado Tênis para Dois, a partir de um osciloscópio, mas foi em 1961 que surgiu o primeiro jogo eletrônico propriamente dito. Um estudante do MIT, Steve Russell e alguns amigos, desenvolveram o jogo SpaceWar [3].

Os jogos eletrônicos podem conquistar usuários por meio dos gráficos complexos criando cenários realistas, pela trama ou história propriamente dita e/ou pelos desafios pelos recursos tecnológicos utilizados.

O desenvolvimento de um jogo eletrônico é um processo longo, complexo e composto de inúmeras fases até alcançar o produto final. Segundo Rodrigues [4], o desenvolvimento de jogos atualmente segue padrões bem distintos. Tal processo apresenta características bastante específicas como a longa duração do projeto, o trabalho com equipes multidisciplinares, a fase de pré-produção, o uso extensivo de recursos multimedia e a integração dos mesmos [5].

Estabelecer uma metodologia adequada para um projeto inibe a inconstância de diversos aspectos inerentes ao desenvolvimento, como os requisitos a se satisfazer. A grande variedade de tecnologias disponíveis fazem com que a adoção de processos rígidos e bem definidos seja dificultada [6].

Partindo do pressuposto que todo jogo é na realidade um software com várias particularidades, o presente artigo visa o estudo e proposta de uma metodologia híbrida para desenvolvimento de jogos a partir de características positivas de três metodologias ágeis de desenvolvimento de software: Scrum, Fast Driven Development e Running Lean. Tais metodologias foram selecionadas por serem ágeis e apresentarem características complementares, resultando em uma única metodologia, chamada AgiGame, com etapas e processos bem definidos e sem "gaps".

2. Processo de Desenvolvimento de Jogos

A área de conhecimento de requisitos de software se preocupa com a elicitação, análise, especificação e validação dos requisitos, bem como a gestão deles durante todo ciclo de vida do projeto. É reconhecido, entre os pesquisadores e profissionais da indústria, que

projetos de software são extremamente vulneráveis quando o as atividades relacionadas aos requerimentos são mal executadas [7].

Durante o planejamento de um projeto, a escolha da metodologia a ser utilizada é fundamental para assegurar a qualidade do produto final, pois para qualquer empresa de jogos, publicadora ou desenvolvedores independentes, a adoção de métodos, normas e padrões profissionais é necessária para que se garanta a qualidade do produto [8]. Dada a grande diversidade dos jogos existentes e as peculiaridades encontradas de acordo com seus propósitos, a adoção de uma metodologia de desenvolvimento rígida para este tipo de software torna-se inviável [9].

Sendo assim, cabe ao produtor estabelecer junto aos líderes de time, um plano de pré-produção do mesmo, determinando os pontos críticos, curso de ação e os fundamentos para atingir os objetivos desejados. Assim como o detalhamento artístico e o planejamento geral [10].

Os jogos não são iguais e nem os métodos usados para o desenvolvimento de cada um. Alguns dos métodos mais comuns são os seguintes [10]:

- "Code-Like-Hell, Fix-Like-Hell", este é provavelmente o método mais antigo e comum, algum planejamento é feito, porém raramente seguido ou atualizado. Os desenvolvedores programam o mais rápido possível para implementar o que eles acham que o design pede no momento, depois ele é testado e corrigido. Esse modelo é suscetível a falhas porque gera situações de grande estresse durante o desenvolvimento.

- "Increments to Completion" é um método de produção no qual o game é desenvolvido em incrementos relativamente compactos, sendo mais voltado para o desenvolvimento de um jogo de aventura ou FPS (First Person Shooter).

- No modelo Cascata toda a equipe se foca em uma determinada parte do projeto. Nessa abordagem, partes do jogo se juntam rapidamente, com pouco tempo para testes entre a criação e a implementação. Esse método requer que tudo corra bem desde o início da produção para que não ocorram graves problemas no final. É difícil realizar grandes mudanças no sistema, utilizando esta metodologia, não sendo recomendado para o desenvolvimento de jogos.

Também se faz necessário estabelecer um plano de produção que contenha o conjunto de documentos que abrange todo o desenvolvimento, desde a concepção até o produto final. Isso inclui o plano específico de cada equipe, artistas, designers e programadores. Uma vez concluído o projeto, as partes interessadas podem revisá-lo, analisando os riscos envolvidos, orçamento, lista de características, planejamento e ativos de arte para só então a produção ser iniciada.

Ainda segundo Irish [10], o plano de produção é composto dos seguintes documentos:

- Sumário executivo: simplesmente descreve porque o jogo é divertido.

- Documento de design criativo: descreve a visão criativa e artística do jogo.

- Documento de design técnico: descreve as características necessárias do jogo, como descrito no documento de design criativo.

- Planejamento e gerenciamento dos riscos: esse documento descreve quais são os riscos envolvidos e como minimizá-los.

- Planejamento do desenvolvimento: planejamento detalhado de todo projeto.

- Orçamento e necessidades financeiras: detalha o custo mensal, dispêndio de capital e afins.

O processo de desenvolvimento de um jogo passa por diversos estágios e cada produtor tem o seu próprio procedimento, mas a maioria segue o padrão apresentado por Irish [10]:

- Concepção: nesta fase é traçado o perfil do jogo, as ideias são geradas através de um brainstorming.

- Protótipo: o protótipo do jogo é desenvolvido, assim os usuários podem começar a testar o jogo, de acordo com o que foi planejado na documentação. Essa fase dura em média de 2 a 4 meses.

- Apresentação: nesta fase os desenvolvedores lançam o jogo para a apreciação da gerência, ou para o editor, mostrando porque o jogo é adequado ao mercado, se é capaz de ser produzido e como será desenvolvido.

- Sinal verde: após aprovação da fase anterior, deve-se reunir uma equipe para começar a trabalhar no jogo, entrevistando diversas produtoras de jogos para formar a equipe ideal para esse projeto. Essa fase também lida com os aspectos legais da produção.

- Pré-produção: nesta fase a equipe define a pipeline de produção, identificando as necessidades e ferramentas necessárias para produção.

- Produção: é neste momento que o jogo começa a ser construído, parte artística e técnica. Esta é a fase mais longa do projeto.

- Garantia de qualidade ou fase de testes: ocorre de três a quatro meses antes do início da produção em massa do jogo. O jogo é testado buscando bugs, deficiências ou incompatibilidades.

- Final: nesta fase o game é enviado para duplicação e preparação para venda. Mesmo depois que o jogo é lançado a equipe continua trabalhando em possíveis pacotes de correção, atualizações ou expansões.

2. Metodologias Ágeis

Em 2001, o manifesto ágil foi criado por membros proeminentes da comunidade ágil, eles se reuniram em Snowbird, Utah, EUA e adotaram o nome métodos ágeis, assim publicado no Manifesto Ágil, documento que reúne os princípios e práticas desta metodologia de desenvolvimento. Os métodos ágeis iniciais incluiam: Scrum (1986), Crystal Clear, Programação Extrema (XP), 1996, Adaptive Software Development e Feature Driven Development (FDD).

Segundo o manifesto ágil para o desenvolvimento de software, algumas vanatagens obtidas com um processo de desenvolvimento de software bem aplicado são:

Vantagens (cliente):

- Foco e maximização do ROI (Retorno do Investimento) e do valor de negócio;
- Entregas do produto mais rápidas, frequentes e regulares;
- Foco no que é prioritário traz mais valor para o usuário, o que se traduz em ganho de usabilidade;
- Transparência e visibilidade do status do projeto;
- Flexibilidade para mudanças de requisitos e prioridades além de maior agilidade na tomada de decisões;
- Melhoria da qualidade do produto final;
- Produtividade;
- Redução dos riscos e das indesejáveis surpresas.

Vantagens (gestor e equipes):

- Escopo e objetivos claros e priorizados;
- Equipes auto gerenciáveis, maior autonomia, disciplina e regularidade;
- Maior comprometimento;
- Melhoria na comunicação. A comunicação intensa com o cliente e a gestão de suas expectativas é parte do processo;
- Inspeção e adaptação constantes do processo em busca da melhoria contínua e a redução dos desperdícios;
- Antecipação dos problemas e maior agilidade na tomada de ações.

2.1. Scrum

O Scrum trabalha com pequenos times auto-organizáveis e multifuncionais, ou seja, não dependem de outras equipes ou que sejam dirigidos por alguém de fora da equipe, assim podem escolher a melhor forma de chegarem ao seu objetivo trazendo flexibilidade, criatividade e produtividade. A Figura 1 mostra o processo do Scrum.

Seu modo de desenvolvimento é feito na forma de ciclos chamados Sprints que são atividades propostas de acordo com as funcionalidades descritas no Product Backlog. As Sprints são geralmente executadas mensalmente, e o trabalho é divido em iterações e de forma incremental, o que garante que uma versão potencialmente funcional do produto do trabalho esteja sempre disponível.

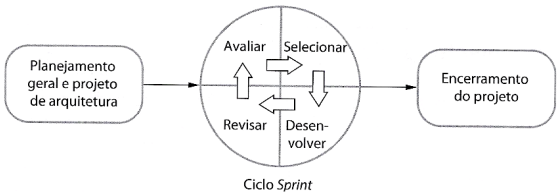

Figura 1: Processo do Scrum [11]

Segundo Schwaber e Sutherland [12] no inicio de cada Sprint temos o Sprint Planning Meeting que não é nada mais que uma reunião de planejamento no qual o Product Owner prioriza as funcionalidades do Product Backlog. O Product Owner é o dono do produto, responsável pelo trabalho da equipe de desenvolvimento e o gerenciamento do Product Backlog.

As tarefas alocadas em um Sprint são transferidas do Product Backlog para o Sprint Backlog, assim estabelecendo uma meta para a equipe de desenvolvimento produzir uma versão usável, por exemplo, uma funcionalidade, que potencialmente incrementa o produto final. Como a equipe de desenvolvimento é auto-organizável, gerencia seu próprio trabalho, a sinergia resultante aperfeiçoa a eficiência e eficácia da equipe. Em uma equipe Scrum não há o reconhecimento de títulos para os integrantes, independente do trabalho realizado, individualmente os integrantes podem ter habilidades e áreas especializadas, porém a responsabilidade pertence à equipe, com isso a equipe de desenvolvimento não contém sub-equipes dedicadas a domínios específicos como teste ou análise de negócios [12].

O Scrum Master é responsável por garantir que o Scrum seja entendido e aplicado de forma aderente a teoria, ensinando a liderar a equipe de desenvolvimento na criação de produtos com auto-gerenciamento e interdisciplinaridade, compreender e praticar a agilidade, atuar como facilitador removendo impedimentos para o progresso da equipe e facilitando os eventos Scrum conforme exigidos ou necessários.

Os eventos Scrum são rotineiros e minimizam a necessidade de reuniões desnecessárias, onde cada evento tem curtas durações com um determinado tempo máximo. Cada evento no Scrum é uma oportunidade de inspecionar e adaptar permitindo transparência e uma inspeção criteriosa do projeto.

O evento mais importante no Scrum é o Sprint, que são as atividades propostas de acordo com as funcionalidades descritas no Product Backlog e tem durações coerentes com o esforço do desenvolvimento, a cada conclusão de uma Sprint é iniciada outra logo em seguida. As Sprints são compostas por uma reunião de planejamento (Sprint Planning Meeting), reuniões diárias (Daily Scrum), trabalho de desenvolvimento, revisão da Sprint (Sprint Review Meeting) e a retrospectiva do Sprint (Sprint Retrospective). Durante o Sprint não podem ser feitas mudanças que afetem o objetivo do mesmo, as metas de qualidade não diminuem, o escopo pode ser clarificado e renegociado entre o Product Owner e a equipe de desenvolvimento dependendo do quanto foi aprendido. Cada Sprint pode ser considerada um projeto flexível que guiará a construção do trabalho e o resultado do produto, porém com o tempo máximo de um mês corrido, já que a complexidade e o risco podem aumentar caso seja um Sprint longo, caso contrário permitem previsibilidade que garante a inspeção e adaptação do progresso [12].

A reunião de planejamento do Sprint tem duração dependente da frequência com a qual ela é feita (uma vez por mês ou a cada duas semanas, por exemplo). O planejamento consiste em duas partes o que será entregue como resultado e o que é necessário para chegar nesse objetivo.

A reunião diária do Scrum tem um tempo aproximado de quinze minutos e tem o objetivo de sincronizar as atividades e planejar o que irá ser realizado no dia. Basicamente temos que responder as seguintes questões: O que foi realizado desde a última reunião? O que será feito até a próxima reunião? Quais os obstáculos que estão no caminho?

Assim pode-se avaliar o progresso do Sprint e inspecionar se a equipe de desenvolvimento esta no caminho certo. O Scrum Master tem que garantir que a reunião será feita dentro do prazo. Reuniões diárias melhoram a comunicação, eliminam a necessidade de outras reuniões, identificam e removem impedimentos para o desenvolvimento, destacam e promovem rápidas tomadas de decisão e melhoram o nível de conhecimento da Equipe de Desenvolvimento. Esta é uma reunião chave para inspeção e adaptação.

O objetivo da revisão do Sprint é verificar o que foi entregue ao final do mesmo, discutir o que foi bem, quais problemas ocorreram e como estes problemas foram resolvidos. A retrospectiva ocorre depois da revisão do Sprint e antes da reunião de planejamento do próximo Sprint e tem como base verificar as lições aprendidas, as formas de aumentar a qualidade do produto e identificar melhorias que serão implementadas na próxima Sprint [12].

2.2. Feature Driven Development (FDD)

Esta metodologia permite desenvolver um sistema de forma rápida e introduzir facilmente novas features (funcionalidades) a este. Uma nova feature demora entre duas horas a duas semanas para ser concluída. O fato de no FDD usarem-se processos simples, proporciona uma rápida exposição do projeto a novos elementos que venham a fazer parte do mesmo.

Métodos adaptáveis focam-se na adaptação a constantes mudanças. Uma equipe sabe o trabalho que lhe cabe na fase atual do projeto, mas desconhece o passo seguinte, que vai sendo decidido durante o decorrer de cada fase [13].

A Figura 2 apresenta, de forma simplificada, o ciclo de vida da Metodologia FDD. Composta por 2 fases e 5 processos, a FDD é muito simples e objetiva, as fases consistem em Concepção & Planejamento e Construção e os processos são definidos e integrados da seguinte forma [14]:

- DMA (Desenvolver um Modelo Abrangente)
  É uma atividade inicial que abrange todo o projeto, realizada por membros do domínio do negócio e por desenvolvedores, sob a orientação de um modelador de objetos experiente, no papel de arquiteto-líder. Realiza-se um estudo dirigido sobre o escopo do sistema e seu contexto. Então, são realizados estudos mais detalhados sobre o domínio do negócio para cada área a ser modelada. Após cada estudo dirigido sobre o domínio, pequenos grupos são formados por membros do domínio do negócio sendo estudado e por desenvolvedores, que comporão seus próprios modelos que satisfaçam o domínio em questão. Os pequenos grupos apresentam seus modelos para serem revisados por parceiros e para discussão. Um dos modelos propostos, ou uma combinação dos modelos, é selecionada por consenso, tornando-se, assim, o modelo para aquela área do domínio do negócio. Realiza-se, então, uma combinação do modelo da área do domínio dentro de um modelo abrangente, ajustando a forma do modelo se for necessário.

- CLF (Construir a Lista de Funcionalidades - Features)
  É uma atividade inicial que abrange todo o projeto, para identificar todas as funcionalidades que satisfaçam aos requisitos. Uma equipe, geralmente composta apenas por programadores-líderes do processo nº 1, é formada para decompor funcionalmente o domínio em áreas de negócio, atividades de negócio dentro delas e passos dentro de cada atividade de negócio, formando assim a lista categorizada de funcionalidades. A categorização de mais alto nível para a lista de

funcionalidades vem da divisão do domínio feita pelos especialistas do domínio no processo nº 1.

- PPF (Planejar por Funcionalidades - Features)
  É uma atividade inicial que abrange todo o projeto, para produzir o plano de desenvolvimento. O gerente de projeto, o gerente de desenvolvimento e os programadores-líderes planejam a ordem na qual as funcionalidades serão implementadas, baseado nas dependências entre elas, na carga de trabalho da equipe de desenvolvimento e também na complexidade das funcionalidades a serem implementadas. As principais atividades neste processo não são uma sequência estrita. Como muitas atividades de planejamento, elas são consideradas em conjunto, com refinamentos feitos a partir de uma ou mais atividades e então considerando os outros novamente. Um cenário típico é levar em conta a sequência de desenvolvimento, depois levar em conta a atribuição das atividades de negócio aos programadores-líderes e, ao fazê-lo, considerar quais das classes principais são atribuídas a quais desenvolvedores (o programador-líder também é um desenvolvedor). Quando esse equilíbrio for alcançado, e a sequência de desenvolvimento e a atribuição das atividades de negócio aos programadores-líderes estiver essencialmente completa, então a posse das classes estará concluída.

- DPF (Detalhar por Funcionalidades - Projetar por Features)
  É uma atividade executada para cada funcionalidade, para produzir o pacote de projeto para ela. Certo número de funcionalidades são agendadas para desenvolvimento ao atribuí-las a um programador-líder. Ele deve selecionar as funcionalidades para desenvolvimento a partir de sua "caixa de entrada" de funcionalidades atribuídas. Ele pode escolher funcionalidades que utilizem as mesmas classes. O programador-líder forma uma equipe de funcionalidades, identificando os proprietários das classes (desenvolvedores) que provavelmente serão envolvidos no desenvolvimento das funcionalidades selecionadas.

- CPF (Construir por Funcionalidade - Feature)
  É uma atividade executada para cada funcionalidade, para produzir uma função com valor para o cliente (funcionalidade). Começando com o pacote de projeto (design), os proprietários de classes programam os itens necessários para que suas classes suportem o projeto para esta funcionalidade. O código desenvolvido passa pelo teste de unidade e pela inspeção, a ordem é determinada pelo programador-líder. Após passar pela inspeção, o código é promovido à versão atual.

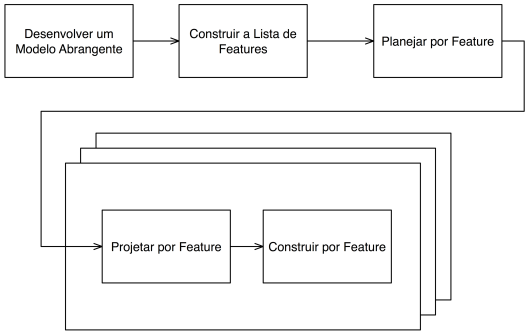

Figura 2: Ciclo de vida da FDD [15]

2.3. Running Lean

Outra metodologia ágil é a Running Lean, adaptada do Sistema Toyota de Produção, ela é um processo sistemático para iteração do plano A para outro plano que funcione, antes que os recursos acabem.

Essa metodologia fornece uma maneira melhor e mais rápida de examinar novas ideias de produtos e construir produtos de sucesso [16]:

- Os pontos principais são velocidade, aprendizado e concentração;
- Testa uma visão por meio do comportamento dos clientes;
- Deve-se envolver os clientes durante o ciclo de desenvolvimento do produto;
- Aborda, em paralelo, o produto e a validação de mercado, usando iterações curtas;
- É um processo disciplinado e rigoroso.

A essência desta metodologia pode ser dividida em 3 passos [16]:

1. Documente o seu plano A
   O primeiro passo é escrever o seu plano, ideia inicial e compartilhar com ao menos uma pessoa.
   É importante aceitar que sua visão inicial está largamente baseada em suposições não testadas. Running Lean ajudará a testar e refinar a ideia.

2. Identifique as partes mais arriscadas deste plano
   Há três estágios distintos pelos quais a ideia deve passar nessa fase, respectivamente:
   - Determinar se o problema vale a pena ser solucionado, antes de investir meses ou anos em uma solução.
   - Medir se o produto que você pretende criar é algo que de fato as pessoas comprariam.
   - foco neste momento muda para crescimento, como escalar o modelo de negócio.

3. Teste seu plano sistematicamente
   O plano deve ser testado por meio de uma série de experimentos. Um experimento é um ciclo ao redor do loop de aprendizado.

O ciclo Lean, apresentado na Figura 3, começa na fase de construção, com um conjunto de ideias ou hipóteses que são usados para criar um artefato (código, página de destino) com a finalidade de testar uma hipótese. Esse artefato é mostrado aos clientes para medir a resposta. Esta informação é usada para a fase de aprendizagem, que serve para validar ou refutar uma hipótese, que por sua vez impulsiona o próximo conjunto de ações [16].

Uma vez que as funcionalidades de cada uma das três metodologias escolhidas foram expostas, agora é o momento de explicar o motivo pelo qual a associação delas garantirá uma metodologia confiável e bem estruturada voltada especificamente para o desenvolvimento de jogos: O Scrum será o principal pilar, garantindo uma base sólida através do conhecimento aprendido por experiência, junto a ela, será incorporada a FDD, boa para o desenvolvimento de jogos devido a mesma ser orientada por funcionalidades e permitir a adição de novas funcionalidades durante o desenvolvimento do produto com facilidade.

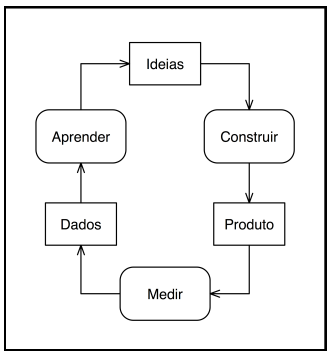

Figura 3: Ciclo Lean: construir, medir, aprender
Adaptado de [16]

A Running Lean é uma metodologia dinâmica, que permite fazer pequenas alterações no projeto, mesmo durante o desenvolvimento, sem comprometer o andamento. A combinação dessas três metodologias resultará em um projeto com baixas taxas de falhas, sem comprometer o orçamento e garantindo o escopo junto à produção.

3. AgiGame: uma Metodologia Híbrida

Muitos desenvolvedores ficam estagnados com metodologias (estudo dos métodos, as etapas a seguir num determinado processo) e processos tradicionais, porém o processo de desenvolvimento (conjunto de atividades, ações e tarefas realizadas na criação do produto) de um jogo vem se tornando mais complexo e mais custoso, fazendo-se necessárias o uso de novas metodologias [17].

Segundo Clinton Keith [18], escritor do livro "Agile Game Development With Scrum" e ex25-CTO (Chief Technical Officer) do estúdio de desenvolvimento de jogos High Moon Studios, a abordagem ágil consiste em iterar durante o desenvolvimento do jogo e reagir aos resultados. Dessa maneira o valor das componentes do jogo surgem mais cedo e podem ser avaliados, assim a equipe pode ajustar melhor o que deve ser feito em seguida.

O emprego do Scrum torna o projeto como um todo mais visível, para que decisões de senso comum possam ser tomadas, trata-se da remoção de impedimentos e também sobre certificar-se que está entregando valor desde o início do projeto. Não se pode esperar até o final do jogo para saber se ele é divertido e fara sucesso, depois que muitos recursos e tempo foram gastos é tarde demais para fazer grandes mudanças.

Keith [18] também cita que é interessante incorporar uma equipe de garantia de qualidade, que testa o jogo, assegurando que os bugs que vão surgindo sejam consertados diariamente. Atrasar o conserto de *bugs* atrasa a equipe de criação de conteúdo. Quando os artistas e designers são atrasados, a quantidade de conteúdo que eles podem criar diminui, o que resulta em um produto final de qualidade inferior.

Existem desafios também ao se adotar o Scrum, como a mudança para uma cultura interativa que promove a participação da equipe. Também é difícil, no início, para as equipes assumirem a responsabilidade envolvida com o nível de compromisso que o Scrum demanda, embora uma vez que as equipes se acostumam e enxergam os benefícios eles estão propensos a se adequar rapidamente a essa nova cultura [17].

Um exemplo de sucesso com o uso de metodologias ágeis (Scrum e XP) foi o jogo Darkwatch para as plataformas Playstation 2 e Xbox. Desenvolvido pela produtora High Moon Studios, da qual Clinton Keith era o CTO na época. As metodologias Scrum e XP (Extreme Programming) foram adotadas durante o desenvolvimento do jogo, foi necessário redefinir a estrutura do departamento de programação e graças ao Scrum foi possível manter o planejamento do jogo [17].

A metodologia Running Lean também se mostra bastante adequada ao desenvolvimento de jogos, devido ao seu ciclo de vida da fase de produção (hipótese, design, desenvolver, testar, analisar, iterar) e a sua cultura de eliminar desperdícios e entregas rápidas. Segundo Seth Sivak, CEO do estúdio Proletariat e ex-designer chefe do jogo Indiana Jones Adventure World (Zynga, desenvolvedora de jogos sociais), o processo de desenvolvimento básico de um jogo com Lean são as fases de conceito (pesquisa de mercado, análise dos concorrentes, preparação da equipe), design, produção, lançamento e operação [19].

Clinton Keith afirma que muitas equipes acabam abandonando o Scrum em uma fase tardia no ciclo de vida do projeto, porque quando a fase de produção se

inicia o valor do Scrum se degrada, fazendo com que eles voltem a métodos mais tradicionais de desenvolvimento. Porém Scrum trabalhando em conjunto com a Lean, baseado em experiências reais, mostrou uma melhora de 56% no custo de produção [20].

FDD se mostra adequada ao desenvolvimento de jogos devido as melhorias ágeis ao desenvolvimento de software que proporciona, tais como o desenvolvimento iterativo, tempo de desenvolvimento aprimorado, testes integrados, e resposta rápida às mudanças de requisitos. O projeto inicia-se fazendo uma lista de features (características), cada uma cuidadosamente planejada e projetada da maneira que será construída.

Essa abordagem oferece valor para o cliente final. As features podem ser dividas em partes menores e entregues aos membros da equipe, caso haja necessidade. Definir os requerimentos como features permite que as equipes de desenvolvimento possam priorizá-las em um backlog de trabalho e depois os desenvolvedores podem entregá-las na ordem que agregue maior valor ao negócio. Esse processo é iterado através de todo conjunto de features, até que o jogo esteja bom o bastante para cumprir os requerimentos e as expectativas do produto final [21].

4.1. Proposta da Metodologia

Como início da proposta de funcionamento da metodologia híbrida AgiGame o processamento será dividido em fases, como mostra a Figura 4. A fase 1 consiste em desenvolver um modelo abrangente, fase 2 consiste em detalhar o modelo da fase anterior em features e a fase 3 será composta pelo desenvolvimento por features, fechando o ciclo de vida da metedologia AgiGame.

Fase 1: Tem como entrada o escopo e os requisitos do projeto. O primeiro passo é listar as ideias e a partir delas construir um produto que seja o refinamento das ideias propostas, a partir disso o produto será mensurado e os dados validados. Após processar-se esta fase obtêm-se o Product Backlog Abrangente, contendo [22]:

- Conceito: engloba o público alvo, originalidade da ideia/inovação, objetivo, nome do jogo, apresentação resumida, estilo e principais regras.

- Escopo do jogo: engloba a história, enredo e tipos de emoção que o jogo irá propiciar ao usuário.

- Especificações técnicas: a plataforma de hardware é definida, linguagem para codificação, volumetria (caso haja suporte para modo online), sistema operacional, requisitos de software e recursos de carregamento.

Uma vez que os dados estejam satisfatórios a última etapa da sprint se inicia, na qual os líderes de equipe se reúnem e são feitas reuniões para planejar e aprender com o desenvolvimento que foi já realizado. Ao término da fase 1 o resultado é a lista abrangente de features, sendo agrupadas e enviadas ao cliente para validação e priorização.

Fase 2: Tem como entrada o *product backlog* abrangente, priorizado e validado pelo cliente, que foi gerado na fase 1. A fase 2 tem como objetivo gerar pequenas features bem detalhadas. ao término desta obtêm-se o Product Backlog Detalhado por Feature:

- Sonorização: engloba a definição das músicas (menus e fases), todo tipo de efeitos sonoros.

- Especificação do jogo: será definido o número de fases, enredo da fase, níveis de dificuldade, vidas, jogabilidade, número de jogares, personagens, sistema de pontuação/evolução do personagem, itens do cenário, conquistas secundárias e gravação (save e load).

- Design gráfico e arte: definição de vídeos e abertura, descrição do layout de menus e telas, descrição de cenário das fases.

Ao término desta fase acontece novamente o agrupamento das features detalhas, sendo enviadas para validação e priorização pelo cliente.

Fase 3: Esta fase engloba o desenvolvimento das features resultando nos primeiros entregáveis. Com o Product Backlog detalhado e priorizado por features, resultante da fase 2, cada sprint é planejada de acordo com o tempo necessário, 4 ou 8 horas a cada Sprint segundo Scrum. Uma vez planejada a Sprint inicia-se o processo de codificação, a medida que as features vão sendo construídas as mesmas passam pelo processo de teste e garantia de qualidade.

Após a equipe de desenvolvimento terminar a construção da Sprint a feature é enviada para validação do cliente, diferente da fase 1 e 2 agora a feature é validade individualmente. Caso o cliente aprove a feature entregue segue-se o fluxo e inicia-se o Sprint Review, onde todo processo é revisto, e então a próxima feature é planejada. Caso o cliente não aprove a feature a mesma deve ser replanejada e o ciclo da fase 3 inicia-se novamente.

A Figura 1 apresenta o ciclo de vida da metodologia AgiGame, processo modelado em BPMN (Business Process Modeling Notation – notação da metodologia de gerenciamento de processos de negócio).

Paralelismo: A ideia principal da metodologia AgiGame é trazer agilidade mitigando riscos, para atingir tal objetivo o conceito de paralelismo será utilizado. Durante a elaboração de cada feature dentro

de sua respectiva fase o processo pode ocorrer em paralelo. Contudo algumas features deverão ser construídas juntas, pois pode haver dependência entre elas, por exemplo no momento que um dispositivo de entrada é acionado o jogo deve emitir um som. Para tratar esse "gap" o líder de equipe, juntamente com outros líderes, devem alinhar nas reuniões diárias quais features devem ser paralelas ou não. Features paralelas são funcionalidades que não dependem de outras funcionalidades, já as features com dependência são funcionalidades que devem ser construídas juntas, tal processo ocorre em todas as fases da AgiGame.

Equipes: A equipe deve composta por especialista, desenvolvedor, líder e tester/analista de qualidade, sendo que sempre será possível realocar atividades quando necessário. O especialista será responsável pela parte específica na qual a equipe está trabalhando, por exemplo, sonorização ou design gráfico. O desenvolvedor é responsável pelo desenvolvimento do código, dependendo do tamanho da feature na qual equipe está trabalhando podem haver mais desenvolvedores. O líder tem a responsabilidade de representar sua respectiva equipe nas reuniões diárias, levando as dificuldades que possam surgir e alinhando as features paralelas com os outros líderes de equipe, assim como trabalhar nas features de acordo com a necessidade. O tester será responsável pela qualidade das features que serão entregues para validação do cliente.

Uma vez que o funcionamento da AgiGame foi detalhado e seu ciclo de vida apresentado é necessário relacionar a funcionalidade ou técnica com a metodologia que serviu de base para obtenção dela. A metodologia híbrida AgiGame, consiste em:

- Pequenos blocos de funcionalidades detalhadas, valorizadas pelo cliente, chamadas Features. Essa característica é fundamentada no FDD.
- Lista de features, cada uma cuidadosamente planejada e projetada. Essa característica é fundamentada no FDD.
- Durante o desenvolvimento a equipe de garantia de qualidade assegura que os bugs que possam surgir sejam corrigidos ao longo do desenvolvimento. Essa característica é fundamentada no Scrum.
- Realização de reuniões diárias. Essa característica é fundamentada no Scrum.
- Envolve-se o cliente durante o ciclo de desenvolvimento do produto. Essa característica é fundamentada no Lean.
- Determinar se o problema vale a pena ser solucionado, antes de investir meses ou anos em uma solução. Essa característica é fundamentada no Lean.
- Medir se o produto que você pretende criar é algo que de fato as pessoas comprariam. Essa característica é fundamentada no Lean.
- Monitoramento detalhado dentro do projeto, agrupamento das funcionalidades e envio para validação do cliente. Essas características são fundamentadas no Scrum e FDD.
- Fornece uma forma de saber, dentro dos primeiros 10% de um projeto, de acordo com as sprints de conceito e início da construção das features, se o plano e a estimativa são sólidos. Essas características são fundamentadas no Scrum e FDD.
- Priorização das funcionalidades em um backlog de trabalho. Essas características são fundamentadas no Scrum e FDD.
- Desenvolvimento iterativo, tempo de desenvolvimento aprimorado, testes integrados, e resposta rápida às mudanças de requisito. Essas características são fundamentadas no Scrum e FDD.
- Testa uma visão por meio do comportamento dos clientes. Essas características são fundamentadas no Scrum, FDD e Lean.

4. Conclusão

O desenvolvimento de jogos é um mercado lucrativo e com grandes oportunidades, porém construí-los exige conhecimentos específicos e o uso de metodologias de desenvolvimento contribui para o sucesso do projeto. Isso possibilita que as fases envolvidas possam ser corretamente planejadas e implementadas, a fim de se obter um produto de qualidade que tenha boa aceitação no mercado, sem comprometer o planejamento e o orçamento do mesmo.

Desta forma, este trabalho consiste no estudo e proposta de uma metodologia voltada especificamente para o desenvolvimento de jogos: a metodologia híbrida AgiGame. A proposta da AgiGame contempla aspectos positivos de três metodologias ágeis (Scrum, Fast Driven Development e Running Lean) relevantes para a construção de jogos.

A AgiGame apresenta processos adaptados em três fases, sendo as duas primeiras voltadas ao levantamento e detalhamento de características e a terceira na construção do produto final. Inicialmente, foi feito um estudo das três metodologias ágeis citadas acima e, após uma comparação entre elas e o levantamento de pontos positivos e negativos de cada uma foi possível elaborar o modelo híbrido proposto.

No entanto, esta metodologia não pôde ser validada por uma equipe desenvolvedora, mas, como continuidade, este trabalho já está sendo estudado e desenvolvido como trabalho de conclusão de curso de um aluno de Ciência da Computação da Universidade Presbiteriana Mackenzie.

Acknowledgements

Os autores gostariam de agradecer à Profa. Msc. Ana Claudia Rossi pelas considerações sobre o projeto.

Referências

[1] M. Aguilera, M. e A. Mendiz. Video games and education: Education in the Face of a Parallel School. Computers in Entertainment (CIE) - Theoretical and Practical Computer Applications in Entertainment archive. Volume 1, Artigo 1, 2003.

[2] S. Kent. The Ultimate History of Video Games. Edição 1, Three Rivers Press, 2001.

[3] R. Demaria e J. Wilson. High Score! McGraw-Hill/Osborne, 2004.

[4] H.F. Rodrigues. Um modelo de processos para o desenvolvimento de jogos educacionais. Programa de Pós-Graduação em Informática – UFPB, João Pessoa, 2009.

[5] C.M. Kanode e H.M. Haddad. Software Engineering Challenges in Game Development. Computer Science Department, Kennesaw State University, Kennesaw, GA, 2009.

[6] A.S.R. Junior, B.T. Nassu e M.A. Jonack. Um estudo sobre os Processos de desenvolvimento de jogos eletrônicos (Games). Departamento de Informática – UFPR, Curitiba, PR, 2002.

[7] P. Bourque e R.R. Fairley. Swebok V3.0: Guide to the software engineering body of knowledge. Piscataway, NJ, IEEE Computer Society, 2014.

[8] R.M. Lavor. Metodologia utilizada no desenvolvimento de games. Trabalho de Conclusão de Curso, Graduação em Informática para Gestão de Negócios, Faculdade de Tecnologia da Zona Leste, Centro Paula Souza, São Paulo, 2009.

[9] R.A. Santos, V.A. Góes e L.F. Almeida. Metodologia OriGame: um processo de desenvolvimento de jogos. Universidade de Taubaté, Departamento de Informática, Taubaté, 2012.

[10] D. Irish. The Game Producer's Handbook. Boston, MA, PTR, 2005.

[11] I. Sommerville. Engenharia de Software. 9a Edição, Ed. Pearson, 2008.

[12] K. Schwaber e J. Sutherland. The Scrum Guide. 2011, Disponível em: <https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf>. Acesso em: 19 out. 2013.

[13] Barbosa et al. Metodologia Ágil: Feature Driven Development. 2001, Disponível em: <http://paginas fe.up.pt/~aaguiar/es/artigos%20finais/es_final_22.pdf>. Acesso em: 17 nov. 2013.

[14] Heptagon. FDD: Estrutura. 2013, Disponível em: <http://www heptagon.com.br/fdd-estrutura>. Acesso em: 17 nov. 2013.

[15] E.S. Gonçalves. Ferramenta para Gerenciamento de Requisitos em Metodologias Ágeis. Monografia de Trabalho de Conclusão do Curso de Ciência da Computação da Universidade Luterana do Brasil, Campus Gravataí, 2008.

[16] A. Maurya. Running Lean – Iterate from Plan A to a Plan That Works. O'Reilly, 2 Ed, 2012.

[17] Bloomberg Businessweek. Game Development Enters the Scrum. 2005, Disponível em: <http://www.businessweek.com/stories/2005-12-21/game-development-enters-the-scrum>. Acesso em: 10 mar. 2014.

[18] C. Keith. Agile Game Development With Scrum. Addison-Wesley, 2010.

[19] S. Sivak. Lean Game Development. Lean Startup Meet Up Conference, Boston, MA, 2013, Disponível em: <http://www.slideshare net/sjsivak/lean-game-development>. Acesso em: 06 mar. 2014.

[20] C. Keith. Gamasutra. Beyond Scrum: Lean and Kanban for Game Developers. 2008, Disponível em: <http://www.gamasutra.com/view/feature/3847/beyond_scrum_lean_and_kanban_for_.php?print=1>. Acesso em: 06 mar. 2014.

[21] J. Meravý. Effective Leadership in Game Industry using Agile Methods. Masaryk University, Faculty of Informatics, República Tcheca, 2013.

[22] A.M.R. Fernandes, E.W.G. Clua, L. Alves e R.L.S. Dazzi. Jogos Eletrônicos: mapeando novas perspectivas. Florianópolis, Visual Books, 2009.

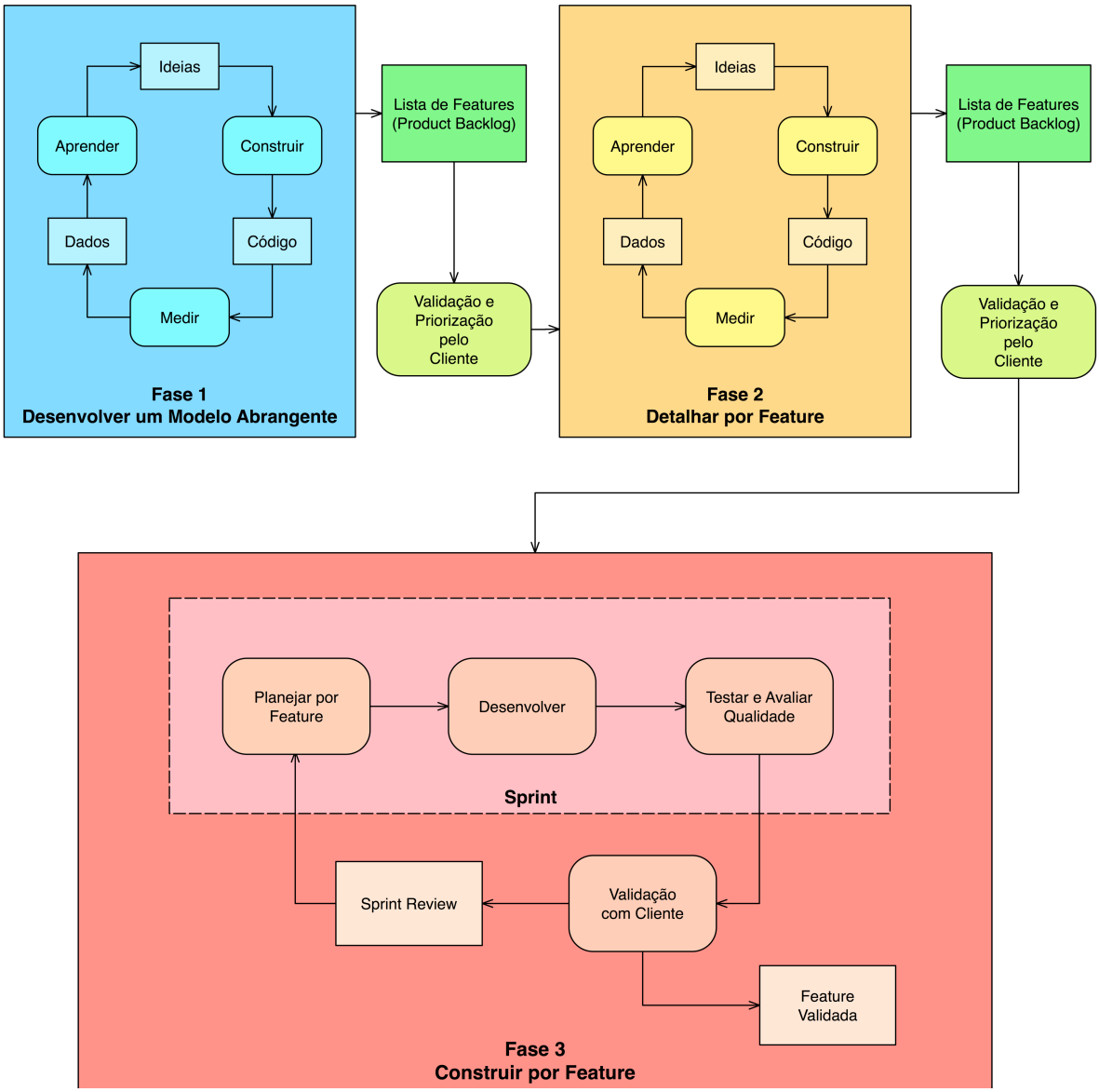

Figura 4: Ciclo de vida da metodologia híbrida AgiGame