**ESPECIFICAÇÃO DO PROJETO**



**BPM Game Engine**

**Versão 1.0**

Fabio Takeshi Ishikawa

Janeiro de 2020

Sumário

[**1. Introdução** 4](#_Toc37282039)

[**1.1 O que é BPM Game Engine** 4](#_Toc37282040)

[1.2 Motivação do Projeto 4](#_Toc37282041)

[1.3 Objetivo deste documento 4](#_Toc37282042)

[1.4 Público-alvo deste documento 4](#_Toc37282043)

[**2. Descrição Geral de BPM Game Engine** 5](#_Toc37282044)

[2.1 Funcionalidades essenciais 5](#_Toc37282045)

[2.2 Os Usuários 5](#_Toc37282046)

[2.3 Plataformas e Ferramentas 5](#_Toc37282047)

[2.4 Documentações 5](#_Toc37282048)

[2.5 Dependências 5](#_Toc37282049)

[**3. Processos de Software** 6](#_Toc37282050)

[**3.1 Planejamento** 6](#_Toc37282051)

[**3.2 Processos de software: Incremental Adaptado** 6](#_Toc37282052)

[**3.3 Descrição das Etapas do Projeto** 7](#_Toc37282053)

[**3.3.1 Análise da sprint** 7](#_Toc37282054)

[**3.3.2 Plano de Testes** 7](#_Toc37282055)

[**3.3.3 Desenvolvimento** 7](#_Toc37282056)

[**3.3.4 Teste de Unidade** 7](#_Toc37282057)

[**3.3.5 Teste de Integração** 7](#_Toc37282058)

[**3.3.6 Teste de Aceitação** 7](#_Toc37282059)

[**3.3.7 Entrega da versão** 7](#_Toc37282060)

[**3.4 Cronograma das sprints** 7](#_Toc37282061)

[**4. Engenharia de Requisitos** 8](#_Toc37282062)

[4.1 Como descrever requisitos de usuário e requisitos de sistema 8](#_Toc37282063)

[4.2 Como definir as prioridades e riscos dos requisitos 8](#_Toc37282064)

[4.3 Processo de elicitação e análise de requisitos 8](#_Toc37282065)

[**5. Modelagem Conceitual (é necessário?)** 9](#_Toc37282066)

[5.1 Cenários ou Diagramas UML? 9](#_Toc37282067)

[**6. Projeto de Software** 10](#_Toc37282068)

[6.1 Arquitetura Game Engine (Core) 10](#_Toc37282069)

[6.2 Visão Geral de todas as ferramentas 10](#_Toc37282070)

[6.2.1 Compiladores 10](#_Toc37282071)

[6.3 Parser BPMN (Camunda) 10](#_Toc37282072)

[6.4 Método PYP – Play Your Process 10](#_Toc37282073)

[**7. Teste de Software** 11](#_Toc37282074)

[7.1 Plano de Testes 11](#_Toc37282075)

[7.2 Testes de Unidade Automatizadas 11](#_Toc37282076)

[7.3 Testes de Integração Automatizada 11](#_Toc37282077)

[7.4 Testes de Aceitação 11](#_Toc37282078)

[7.5 Documentação de Testes 11](#_Toc37282079)

[**8. Gerência de Configuração de Software** 12](#_Toc37282080)

[8.1 Slack – Comunicação entre a equipe 12](#_Toc37282081)

[8.2 Trello – Gestão de tarefas (Kanban) 12](#_Toc37282082)

[8.3 Git e Github – Controle de Versão 12](#_Toc37282083)

[8.3.1 Versionamento Semântico 12](#_Toc37282084)

[8.3.2 Workflow: Pull Request Workflow 12](#_Toc37282085)

[8.3.3 Workflow: No Switch Yard 12](#_Toc37282086)

[8.3.4 Guia de contribuição open source 12](#_Toc37282087)

[8.4 DevOps com Github, TravisCI e Codacy 12](#_Toc37282088)

[**9. Licenças** 13](#_Toc37282089)

[**10. Cronograma** 13](#_Toc37282090)

[**10. Garantia de Qualidade** 14](#_Toc37282091)

# **1. Introdução**

Este documento tem como objetivo descrever em detalhes a especificação do projeto BPM Game Engine, um motor de jogo para o desenvolvimento de jogos baseados em modelos de processos de negócio.

São apresentados o processo de software, isto é, a descrição das etapas do projeto.

## **1.1 O que é BPM Game Engine**

BPM Game Engine é um motor de jogo que surgiu para apoiar a criação e desenvolvimento de jogos digitais baseados em modelos de processos de negócio. Este novo gênero de jogo surgiu a partir da pesquisa de doutorado “Play Your Process (PYP) – Um método de Design de Jogos Digitais Baseados em Modelos de Processos de Negócio”, onde é apresentado um método que permite transformar elementos de modelos de processos de negócio em elementos de design de jogos.

Jogos digitais baseados em modelos de processos de negócio são jogos com propósito que apresentam um processo de negócio de forma ludificada e permitem assim, aos jogadores, compreender e aprender seu funcionamento de forma divertida e engajante e desenvolver reflexões em relação à sua necessidade, sua prática, seus valores, seus desafios e limitações de execução. (CLASSE e ARAUJO).

Além do suporte completo para o desenvolvimento de jogos baseados em modelos de processos de negócio, este motor de jogo também pode ser utilizado para criar jogos de diferentes gêneros e propósitos.

## 1.2 Motivação do Projeto

## 1.3 Objetivo deste documento

## 1.4 Público-alvo deste documento

Este documento está destinado a todas as pessoas envolvidas no projeto e desenvolvimento de BPM Game Engine.

# **2. Descrição Geral de BPM Game Engine**

Nesta seção, estão descritas todas as funcionalidades que compõe BPM Game Engine.

## 2.1 Funcionalidades essenciais

Motor de jogo como middleware.

Fornecer funções para criar diferentes tipos de janelas de aplicação independente de plataforma.

Fornecer funções para construção de níveis do jogo

Fornecer funções para criação de objetos e entidades do jogo

Fornecer funções para configurar diferentes tipos de física e colisões

Fornecer funções para capturar eventos de teclado, mouse

Fornecer funções para tratamento de tempo

## 2.2 Os Usuários

## 2.3 Plataformas e Ferramentas

## 2.4 Documentações

## 2.5 Dependências

# **3. Processos de Software**

Com o intuito de entregar resultados de alta qualidade e dentro do prazo estabelecido, através de um planejamento cuidadoso, foi estabelecido um escopo bem definido do projeto e as etapas necessárias para o desenvolvimento do motor de jogo. As etapas foram organizadas utilizando a metodologia de processo incremental e algumas práticas da metodologia ágil. A seguir, segue os detalhes do planejamento do projeto e como foram definidos os processos de software.

## **3.1 Planejamento**

A ideia do planejamento é definir bem o escopo do projeto. A ideia é que todos os envolvidos no projeto tenham uma visão clara da versão final do produto e como será o processo para chegar a essa versão. Na fase de planejamento, conseguimos definir as principais funcionalidades do software, assim como os limites de projeto e os obstáculos a serem enfrentados durante o desenvolvimento.

Com a definição das principais funcionalidades, podemos então, dividi-las em funcionalidades menores a um nível em que possamos visualizar as dependências entre elas. Por exemplo, para criar um sistema de renderização é necessário que o sistema de janelas esteja pronto. A partir dessa divisão, conseguimos finalmente definir as etapas de desenvolvimento de acordo com a prioridade e as dependências das funcionalidades.

## **3.2 Processos de software: Incremental Adaptado**

Após a definição das etapas, precisamos definir como será o processo de desenvolvimento de cada etapa. Existem diversas metodologias fundamentadas na engenharia de software e adotadas no mercado atual, entretanto, a própria engenharia de software aconselha adaptar a metodologia existente para se adequar ao seu projeto. Portanto, a metodologia adotada neste projeto consiste em uma adaptação do processo incremental com algumas práticas da metodologia ágil.

Foi definido, então, que a duração de cada etapa (sprint) é de quinze dias para coincidir com as reuniões do projeto. A ideia é que sejam apresentados nestas reuniões os resultados obtidos e a versão desenvolvida em cada sprint.

## **3.3 Descrição das Etapas do Projeto**

A seguir segue os detalhes que devem ser seguidos para a execução de cada etapa.

### **3.3.1 Análise da sprint**

A análise de sprint tem como objetivo definir os requisitos e as tarefas a serem desenvolvidas durante a sprint. Esta fase inclui a análise do estado atual do projeto, definição da versão a ser entregue no final da sprint e a listagem das tarefas e requisitos a serem desenvolvidos. Nesta fase, é necessário atualizar o documento “Lista de Requisitos”.

### **3.3.2 Plano de Testes**

### **3.3.3 Desenvolvimento**

Após a análise e a definição da lista de tarefas, é iniciado o desenvolvimento das tarefas.

### **3.3.4 Teste de Unidade**

Para cada requisito funcional listado no documento “Lista de Requisitos” desenvolvido, é necessário também desenvolver o seu teste de unidade. A ideia é que para cada inclusão de funcionalidades no software, todos os testes de unidade sejam executados de forma automatizada. De fato, é necessário que todos os métodos das classes criadas no projeto passem pelo teste de unidade automatizada, garantindo alta cobertura de código e, consequentemente, a qualidade de software.

### **3.3.5 Teste de Integração**

### **3.3.6 Teste de Aceitação**

### **3.3.7 Entrega da versão**

Todos os documentos devem ser atualizados de acordo com o desenvolvimento e os resultados obtidos na etapa desenvolvida.

**Nota:** Os itens com fundo azul representam as etapas que utilização ferramentas automatizadas relacionada a DevOps como a Integração Contínua.

## **3.4 Cronograma das sprints**

A seguir, o cronograma das etapas definidas para o projeto:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Etapa (sprint)** | **Tarefas e Requisitos** | **Data de início** | **Data de conclusão** | **Estado** |
| **Planejamento** | Definição do Projeto | 02/01 | 12/02 | **Terminada** |
| Especificação da sprint #1 |
| Criação dos documentos iniciais |
| Criação do ambiente de desenvolvimento |
| **Sprint #1** | Atualização dos documentos do projeto | 13/02 | 27/02 | **Terminada** |
| Inclusão da classe GEWindow |
| Inclusão da classe GEApiWrapper e GEWinApiWrapper |
| Tratamento de mensagens do sistema operacional Microsoft Windows |
| Tratamento de eventos de mouse e teclado |
| Inclusão do sistema de renderização (OpenGL) |
| Inclusão da classe GEEventHandler |
| **Sprint #2** | Inclusão de TimeHandler e High Resolution Timer | 28/02 | 12/mar | **Terminada** |
| Cálculo da duração de quadros e FPS |
| Inclusão do teste de unidade |
| Inclusão do parser de arquivos .BMP |
| Inclusão do paser de arquivos .JPG |
| Atualização das classes desenvolvidas na sprint anterior |
| Sprint #3 |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Sprint #1

OBJETIVO: Versão exibindo diversos estilos de janelas de aplicação (Windows) e capturando eventos do mouse e teclado

Sprint #2

OBJETIVO: Versão exibindo a funcionalidade de gerenciador de quadros, cálculo do tempo de cada quadro e FPS

Reuniões

Reunião (13/2) ~Projeto Inicial

Reunião (27/2) (janela e estilos de janelas) \*entrega sprint 1

Reunião (12/3) (polygon e asset) \* entrega sprint 2

Reunião (26/3)\*cancelado

Reunião (06/4) (assets com maria e investigador) entrega sprint 3

Reunião (20/4)\*next

# **4. Engenharia de Requisitos**

## 4.1 Como descrever requisitos de usuário e requisitos de sistema

## 4.2 Como definir as prioridades e riscos dos requisitos

## 4.3 Processo de elicitação e análise de requisitos

* Descoberta e compreensão dos requisitos
* Classificação e organização dos requisitos
* Priorização e negociação dos requisitos
* Documentação dos requisitos

# **5. Modelagem Conceitual (é necessário?)**

## 5.1 Cenários ou Diagramas UML?

# **6. Projeto de Software**

Nesta seção abordamos todos os componentes que fazem parte do software, assim como todos os componentes utilizados para construí-lo. É descrita também a arquitetura de BPM Game Engine e o mapeamento utilizado no método Play Your Process.

## 6.1 Arquitetura Game Engine (Core)

## 6.2 Visão Geral de todas as ferramentas

### 6.2.1 Compiladores

Como a ideia do projeto é atender diversas plataformas, o projeto deve ser possível ser compilado por diversos compiladores. A seguir segue a lista de compiladores utilizados para compilar o projeto:

* GCC 9.2.0 (MinGW) – MinGW.org GCC Build-20200227-1

## 6.3 Parser BPMN (Camunda)

## 6.4 Método PYP – Play Your Process

# **7. Teste de Software**

## 7.1 Plano de Testes

## 7.2 Testes de Unidade Automatizadas

## 7.3 Testes de Integração Automatizada

## 7.4 Testes de Aceitação

## 7.5 Documentação de Testes

# **8. Gerência de Configuração de Software**

## 8.1 Slack – Comunicação entre a equipe

## 8.2 Trello – Gestão de tarefas (Kanban)

## 8.3 Git e Github – Controle de Versão

O sistema de controle utilizado neste projeto é o Git (<https://git-scm.com/>). A escolha de Git para o projeto é motivada pela sua popularidade e facilidade de integração com diversos projetos existentes. Além disso, ela é uma ferramenta livre e gratuita.

Para este projeto é recomendável utilizar qualquer versão a partir de **2.25.1**.



*Git Logo by Jason Long is licensed under the Creative Commons Attribution 3.0 Unported License.*

O projeto também com a plataforma online Github (<https://github.com/>) para a hospedagem do repositório do projeto. Esta plataforma também permite a integração com diversas ferramentas úteis e sistemas como o Travis-CI para automatização de builds e testes – práticas essências de integração contínua (DevOps).



## 8.3.1 Versionamento Semântico

## 8.3.2 Workflow: Pull Request Workflow

## 8.3.3 Workflow: No Switch Yard

## 8.3.4 Guia de contribuição open source

## 8.4 DevOps com Github, TravisCI e Codacy

# **9. Licenças**

# **10. Cronograma**

# **10. Garantia de Qualidade**