**ESPECIFICAÇÃO DO PROJETO**



**BPM Game Engine**

**Versão 1.0**

Fabio Takeshi Ishikawa

Janeiro de 2020

Sumário

[**1. Introdução** 4](#_Toc37844238)

[**1.1 O que é BPM Game Engine** 4](#_Toc37844239)

[**1.2 Motivação do Projeto** 4](#_Toc37844240)

[**1.3 Objetivo deste documento** 4](#_Toc37844241)

[**1.4 Público-alvo deste documento** 4](#_Toc37844242)

[**2. Descrição Geral de BPM Game Engine** 5](#_Toc37844243)

[**2.1 Funcionalidades essenciais** 5](#_Toc37844244)

[**2.2 Os Usuários** 5](#_Toc37844245)

[**2.3 Plataformas** 5](#_Toc37844246)

[**2.4 Documentações** 6](#_Toc37844247)

[**2.5 Limitações** 6](#_Toc37844248)

[**3. Processos de Software** 7](#_Toc37844249)

[**3.1 Planejamento** 7](#_Toc37844250)

[**3.2 Processos de software: Incremental Adaptado** 7](#_Toc37844251)

[**3.2.1 Descrição das Etapas do Projeto** 8](#_Toc37844252)

[**3.3 Cronograma das sprints** 9](#_Toc37844253)

[**4. Engenharia de Requisitos** 11](#_Toc37844254)

[4.1 Como descrever requisitos de usuário e requisitos de sistema 11](#_Toc37844255)

[4.2 Como definir as prioridades e riscos dos requisitos 11](#_Toc37844256)

[4.3 Processo de elicitação e análise de requisitos 11](#_Toc37844257)

[**5. Modelagem Conceitual (é necessário?)** 12](#_Toc37844258)

[5.1 Cenários ou Diagramas UML? 12](#_Toc37844259)

[**6. Projeto de Software** 13](#_Toc37844260)

[6.1 Arquitetura Game Engine (Core) 13](#_Toc37844261)

[6.2 Visão Geral de todas as ferramentas 13](#_Toc37844262)

[6.2.1 Compiladores 13](#_Toc37844263)

[6.3 Parser BPMN (Camunda) 13](#_Toc37844264)

[6.4 Método PYP – Play Your Process 13](#_Toc37844265)

[**7. Teste de Software** 14](#_Toc37844266)

[7.1 Plano de Testes 14](#_Toc37844267)

[7.2 Testes de Unidade Automatizadas 14](#_Toc37844268)

[7.3 Testes de Integração Automatizada 14](#_Toc37844269)

[7.4 Testes de Aceitação 14](#_Toc37844270)

[7.5 Documentação de Testes 14](#_Toc37844271)

[**8. Gerência de Configuração de Software** 15](#_Toc37844272)

[8.1 Slack – Comunicação entre a equipe 15](#_Toc37844273)

[8.2 Trello – Gestão de tarefas (Kanban) 15](#_Toc37844274)

[8.3 Git e Github – Controle de Versão 15](#_Toc37844275)

[8.3.1 Versionamento Semântico 15](#_Toc37844276)

[8.3.2 Workflow: Pull Request Workflow 15](#_Toc37844277)

[8.3.3 Workflow: No Switch Yard 15](#_Toc37844278)

[8.3.4 Guia de contribuição open source 15](#_Toc37844279)

[8.4 DevOps com Github, TravisCI e Codacy 15](#_Toc37844280)

[**10. Cronograma do Projeto** 16](#_Toc37844281)

[**11. Garantia de Qualidade** 17](#_Toc37844282)

# **1. Introdução**

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

Esta especificação está dividida em dez capítulos, que inclui a descrição geral de BPM Game Engine e as suas principais funcionalidades, a especificação dos processos de software e a engenharia de requisitos aplicadas no projeto, a modelagem conceitual (visão de alto nível) e a arquitetura de software em um nível mais detalhado, visão geral das ferramentas utilizadas para o desenvolvimento e a gerência de configuração do projeto e, para finalizar, a descrição do planejamento de testes e garantia de qualidade.

## **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 gênero de jogo, que apresenta um processo de negócio de forma ludificada, permite 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**

Este documento, além de servir como especificação de projeto, também pode ser utilizada como uma forma de comunicação entre os envolvidos no projeto e para a visualização do estado atual de desenvolvimento.

## **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**

BPM Game Engine é um motor de jogo multiplataforma com capacidade de renderização em tempo real de imagens bidimensionais e tridimensionais. Com o motor de jogo é possível criar um jogo 2D ou 3D completo do início até o final. Além disso, ele fornece diversas funcionalidades para criar um jogo do gênero baseado em modelos de processos de negócio como um parser de arquivos BPMN (Business Process Model Notation) e um mapeamento de elementos de negócio em elementos de jogos.

[Descrever aqui os diferentes tipos de motor de jogo e explicar em qual modelo BPM Game Engine se encaixa. Por exemplo, middleware, framework, etc]

Nesta seção, descrevemos as funcionalidades essenciais de BPM Game Engine, os usuários, as plataformas onde os jogos criados por este motor de jogo podem ser executados, listagem de todos os documentos do projeto e as limitações do motor de jogo.

## **2.1 Funcionalidades essenciais**

* Criações de janelas de aplicação em diferentes tipos de plataformas (Microsoft Windows, Linux, Android e Web).
* Entrada de dados a partir de diversos tipos de dispositivos como teclado, mouse e joystick.
* Renderização de imagens bidimensionais e tridimensionais em tempo real.
* Sistemas de colisões e simulações de física.
* Sistema de entidades e de mensagens.
* Utilização de scripts em linguagem LUA para a lógica do jogo e configuração de jogo.
* Gerenciamento de recursos externos que inclui o leitor de arquivos de imagens no formato JPEG, BMP e PNG e arquivos de modelos tridimensionais no formato OBJ.
* Leitor e analisador de arquivos BPMN e mapeamento do método Play Your Process.

## **2.2 Os Usuários**

Todas as pessoas que gostariam de criar jogos digitais de diferentes gêneros e em diferentes tipos de plataforma, sobretudo, jogos do gênero baseado em modelos de processos de negócio.

## **2.3 Plataformas**

A ideia do projeto nasceu com a ideia de possibilitar a criação de jogos em diferentes tipos de plataforma. O foco deste motor de jogo é criar jogos para as seguintes plataformas:

* Microsoft Windows 10
* Linux (distribuições Ubuntu e CentOS)
* Android™
* WEB (navegadores web como Google Chrome).

## **2.4 Documentações**

O projeto conta os seguintes documentos:

* Especificação do Projeto
* Guia de Desenvolvimento
* Guia de Usuário
* Lista de Requisitos
* Lista de Testes
* Planejamento de Testes
* Avaliação de Desempenho

## **2.5 Limitações**

Esta seção descreve o que não é possível realizar nesta versão de BPM Game Engine.

* Jogos online (comunicação via internet).

# **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**

O planejamento inicial é muito importante para definir bem o escopo do projeto. A ideia é que todos os envolvidos no projeto tenha uma visão clara da versão final do produto e dos processos de software para a execução do projeto. Na fase de planejamento deste projeto, foi definido as principais funcionalidades do software, assim como as limitações e os obstáculos a serem enfrentados durante o desenvolvimento.

Com a definição das principais funcionalidades, foi possível dividi-las em funcionalidades menores a um nível de visão suficiente para visualizar as dependências entre elas e definir a prioridade de cada uma. Por exemplo, para criar um sistema de renderização é necessário que o sistema de janelas esteja pronto. Portanto, a implementação do sistema de janelas tem a prioridade mais alta do que o sistema de renderização.

Com essa divisão de funcionalidades em requisitos mentores, foi possível definir as etapas iniciais de desenvolvimento (sprints) e a lista de requisitos com as prioridades e dependência para cada etapa.

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

Existem diversas metodologias fundamentadas na engenharia de software e consolidadas no mercado atual que podem ser aplicadas em diferentes tipos de projeto de software. Ao mesmo tempo, a própria engenharia de software (e a experiência no mundo real) aconselha a adaptação de metodologias existentes para que o desenvolvimento do projeto consiga fluir da melhor maneira possível. Afinal, com o crescimento da complexidade dos softwares modernos e a possibilidade de uso em diversas áreas diferentes, é praticamente impossível definir uma metodologia que funcione completamente para todos os tipos de projetos.

Portanto, para este projeto foram adotados métodos e práticas de duas metodologias existentes: incremental e ágil.

A duração de cada etapa (sprint) de desenvolvimento é de quinze dias. Este tempo foi definido para coincidir com as datas de reuniões do projeto. A ideia é que sejam apresentadas nestas reuniões os resultados obtidos e a versão gerada do produto de cada etapa.

### **3.2.1 Descrição das Etapas do Projeto**

Cada etapa de desenvolvimento é dividida em diversas fases: a análise da sprint atual, definição da lista de requisitos e a versão desta etapa, desenvolvimento dos requisitos, testes de unidade, testes de integração e a geração da versão de software da sprint e atualização de todos os documentos.

#### 3.2.1.1 Análise da sprint atual

Nesta fase, é realizado a análise do estado atual do projeto (em geral, depois da apresentação dos resultados da reunião) e a definição de novos requisitos a serem desenvolvidos. Além disso, é definido também a versão esperada que deverá ser gerada no final da etapa. Seguindo os métodos do processo incremental, a ideia é que ao final de cada sprint, o usuário possa receber uma versão funcional do produto. Assim, ao final de cada etapa, o produto é incrementado com novas funcionalidades de versão em versão, até alcançar a versão final. Portanto, nesta fase, é muito importante executar os seguintes itens:

* Análise do estado atual do projeto
* Lista de novos requisitos para a sprint atual
* Definição da versão para a sprint atual (objetivo)
* Atualização dos documentos “Lista de Requisitos”

#### 3.2.1.2 Desenvolvimento

Esta fase é a fase de desenvolvimento dos novos requisitos definidos anteriormente. Todos os detalhes de desenvolvimento como convenção de código, fluxo de trabalho com o sistema de controle de versão e outras práticas relacionadas ao desenvolvimento estão descritas no documento “Guia de Desenvolvimento”.

**Nota:** Conforme o desenvolvimento, é possível que os documentos “Guia de Desenvolvimento” e “Lista de Requisitos” sejam atualizados constantemente.

#### 3.2.1.3 Teste de Unidade e Testes de Integração

A ideia principal de incluir testes de unidade no projeto é obviamente garantir a qualidade do produto, porém, a forma como foi implementada neste projeto, permite também agilizar certos processos de software. Utilizando conceitos de DevOps, foi possível empregar a automatização de testes e a Integração Contínua no projeto. A forma como os testes de unidade é implementada no projeto permite que ferramentas externas executem os testes de unidade de forma automática e verifique também se a nova implementação não afetou outros pontos do software. A ferramenta utilizada neste caso é o Travis CI. Todos os detalhes sobre este assunto estão descritos no capítulo mais adiante deste documento.

#### 3.2.1.4 Entrega da versão

Após o término de desenvolvimento de todos os requisitos, todos os documentos devem ser atualizados de acordo com os resultados obtidos e a nova versão do software deverá ser gerada com o número de versão correto. Com isto, basta apresentar a apresentação para a entrega da versão.

## **3.3 Cronograma das sprints**

A seguir, o cronograma das etapas que já foram executadas e as etapas que ainda devem ser executadas:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Etapa (sprint)** | **Tarefas e Requisitos** | **Data de início** | **Data de conclusão** | **Estado** | **Data da Reunião** |
| **Planejamento** | Definição do Projeto | 02/01 | 12/02 | **Terminada** | 13/02 |
| Especificação da sprint #1 |
| Criação dos documentos iniciais |
| Criação do ambiente de desenvolvimento |
| **Sprint #1**  **Objetivo:**  O usuário deverá ser capaz de criar uma janela de aplicação de diversos estilos e capturar os eventos de telcado e mouse. | Atualização dos documentos do projeto | 13/02 | 27/02 | **Terminada** | 27/02 |
| Inclusão da classe GEWindow |
| Inclusão da classe GEApiWrapper e GEWinApiWrapper |
| Tratamento de mensagens do sistema operacional Microsoft Windows |
| Exibição da janela de aplicação na tela |
| Inclusão de diversos estilos de janela de aplicação. |
| Tratamento de eventos de mouse e teclado |
| Inclusão do sistema de renderização (OpenGL) |
| Inclusão da classe GEEventHandler |
| **Sprint #2**  **Objetivo:**  O usuário deve conseguir manipular o tempo de execução da aplicação através do uso de Timers e obter diagnósticos como número de quadros executados por segundo. | Inclusão da classe GameEngine | 28/02 | 12/mar | **Terminada** | 12/03 |
| Implementação do loop principal |
| Inclusão da classe GETimeHandler |
| Cálculo do tempo do quadro |
| Inclusão do teste de unidade |
| Inclusão de High-Resolution-Timer |
| Inclusão da avaliação |
| Inclusão da classe de GEDiag |
| Implementação da classe Timer |
| Cálculo de FPS |
| Melhoria de todas as classes |
| **Sprint #3** |  | 13/03 | 06/04 | Terminada | 06/04 |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| **Sprint #4** |  |  |  |  | 20/04 |

# **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

# **10. Cronograma do Projeto**

# **11. Garantia de Qualidade**