Diseño Detallado de Arquitectura de Software

Juan Carlos Gómez Hernández[1]

[1] Universidad Linda Vista, Ex-Finca Santa Cruz No. 1, 29750, México correo 1@ulv.edu.mx
[2] Springer Heidelberg, Tiergartenstr. 17, 69121 Heidelberg, Germany lncs@springer.com
[3] ABC Institute, Rupert-Karls-University Heidelberg, Heidelberg, Germany {abc,lncs}@uni-heidelberg.de

1. Propósito

Este documento tiene como objetivo proporcionar una descripción de la arquitectura del Simulador de Equilibrio, una herramienta educativa diseñada para ayudar a los estudiantes a comprender la primera condición de equilibrio en Física. A través de diversas vistas arquitectónicas, se busca capturar y comunicar las decisiones arquitectónicas clave que guiarán el desarrollo del sistema, asegurando que cumpla con los requisitos educativos establecidos.

1.1. Ámbito de aplicación.

El Simulador de Equilibrio es una herramienta independiente que permite a los estudiantes explorar conceptos fundamentales de equilibrio sin necesidad de conexión a Internet. Teniendo así una herramienta que pueden tener a la mano, y ayudarse con problemas de seguridad por parte de los estudiantes y proporcionar una experiencia de aprendizaje interactiva y accesible.

2. Definiciones y acrónimos

Para facilitar la comprensión del documento y asegurar que todos los lectores tengan una interpretación clara de los términos utilizados, se presenta a continuación un glosario de definiciones, acrónimos y abreviaturas relevantes:

- Simulador educativo: Herramienta digital diseñada para replicar virtualmente un fenómeno o proceso específico, permitiendo a los estudiantes explorar y comprender conceptos de forma interactiva y atractiva.
- Python: Lenguaje de programación de alto nivel, interpretado y versátil, conocido por su sintaxis clara y legible. Se utiliza en desarrollo web, análisis de datos, inteligencia artificial y otras áreas.
- Pygame: Biblioteca de Python utilizada para crear videojuegos y simulaciones, que proporciona funcionalidades para gráficos, sonido y manejo de eventos.

- Matplotlib: Biblioteca de Python que permite crear gráficos en 2D, facilitando la visualización de datos en un entorno cartesiano.
- Tkinter Canvas: Herramienta en Python que permite crear gráficos vectoriales 2D y superponer ventanas, útil para la construcción de interfaces gráficas interactivas.
- Diagrama de cuerpo libre: Representación gráfica que muestra todas las fuerzas que actúan sobre un cuerpo en equilibrio, facilitando el análisis de las condiciones necesarias para mantener el equilibrio.
- Primera condición de equilibrio: Principio físico que establece que la suma vectorial de todas las fuerzas actuando sobre un cuerpo es igual a cero ($F = 0$), lo que implica que no hay movimiento neto en ninguna dirección.
- Interfaz gráfica: Elemento visual del simulador que permite al usuario interactuar con el sistema, ajustando parámetros y visualizando resultados.
- Usabilidad: Medida de cuán fácil y eficiente es utilizar un sistema o herramienta, incluyendo aspectos como la satisfacción del usuario y la facilidad de aprendizaje.
- PSSUQ (Post-Study System Usability Questionnaire): Cuestionario utilizado para evaluar la usabilidad de sistemas después de su uso, centrado en la calidad del sistema y la interfaz.

3. Diseño arquitectónico

El diseño arquitectónico es MVC se aplica para separar la lógica del negocio, la presentación y el control de la aplicación. Esto permite una mayor modularidad y facilita el mantenimiento. Para este simulador se aplico el mismo patrón de diseño:

Modelo En este caso, el modelo incluye las funciones que realizan cálculos físicos, como "Solution", çalculateTensionsz çalculateBodyPosition". Estas funciones manejan la lógica del simulador y no están directamente relacionadas con la interfaz gráfica.

Vista. La vista está representada por las funciones que dibujan en la pantalla, como "drawScenez çrearGrafico". Estas funciones son responsables de mostrar los resultados al usuario.

Controlador. La función 2main.ªctúa como controlador. Maneja los eventos del usuario (clics del mouse, movimiento) y actualiza el modelo y la vista en consecuencia.

4. Descripción de la descomposición

La arquitectura del Simulador de Equilibrio se descompone en varios subsistemas, cada uno con funciones específicas que contribuyen al funcionamiento general del sistema. A continuación se presenta una descripción funcional de los subsistemas, acompañada de un diagrama de flujo de datos (DFD) de nivel superior y diagramas de descomposición estructural.

4.1. Subsistemas

Interfaz Gráfica

- Descripción: Este subsistema proporciona la interacción del usuario con el simulador. Permite a los usuarios ajustar parámetros, iniciar simulaciones y visualizar resultados.

Funciones:

- Métodos para iniciar la simulación al ajustar parámetros.
- Métodos para mostrar resultados y gráficos generados.
- Proporciona retroalimentación visual inmediata sobre las interacciones del usuario.

4.2. Módulo de Cálculo

- Descripción: Contiene la lógica para calcular las tensiones y posiciones en equilibrio basadas en los parámetros ingresados por el usuario.

Funciones:

- Métodos que reciben parámetros desde la interfaz gráfica (peso, ángulo).
- Métodos que devuelven resultados calculados (tensiones y posiciones).

Módulo Gráfico

- Descripción: Genera visualizaciones gráficas basadas en los cálculos realizados por el módulo de cálculo.

Funciones:

- Métodos que crean gráficos y devuelven la posición de las imágenes que se muestran en la interfaz gráfica.
- Actualiza las visualizaciones en tiempo real a medida que se modifican los parámetros.

Diagramas de Descomposición Estructural

4.3. Interfaz Gráfica

Clases:

- SimulationControl: Permite ajustar parámetros (peso, ángulo).
- Result: Muestra resultados y gráficos.

4.4. Módulo de Cálculo

Clases:

- PhysicsCalculator: Realiza cálculos relacionados con fuerzas y equilibrio.
- TensionCalculator: Específico para calcular el movimiento de las cuerdas.

5. Fundamento del diseño

La arquitectura seleccionada para el Simulador de Equilibrio se basa en el patrón Modelo-Vista-Controlador (MVC), que proporciona una clara separación entre la lógica del negocio, la presentación y el control del flujo de datos. Esta elección se fundamenta en varias razones:

- Modularidad: La separación entre los diferentes componentes permite un desarrollo más organizado y facilita el mantenimiento del código. Cada módulo puede ser desarrollado y probado independientemente.
- Facilidad de mantenimiento: Al seguir el patrón MVC, cualquier cambio en la lógica del negocio o en la interfaz gráfica puede realizarse sin afectar a otros componentes, lo que reduce el riesgo de introducir errores al modificar el sistema.
- Escalabilidad: La arquitectura permite agregar nuevas funcionalidades o mejorar las existentes sin reestructurar completamente el sistema, lo cual es crucial para futuras versiones del simulador.
- Usabilidad: La interfaz gráfica intuitiva mejora la experiencia del usuario, permitiendo una interacción fluida con el simulador, lo cual es esencial para un entorno educativo.

6. Descripción general de la interfaz de usuario.

La interfaz del Simulador de Equilibrio está diseñada para ser intuitiva y accesible, permitiendo a los usuarios completar todas las funciones esperadas sin complicaciones. Desde la perspectiva del usuario, las funcionalidades incluyen:

- Ajuste dinámico de parámetros: La interfaz permite a los estudiantes ajustar fácilmente parámetros como peso y ángulos mediante controles con el teclado. Esto facilita una interacción rápida y efectiva con el simulador.

- Visualización gráfica interactiva: A medida que los usuarios ajustan los parámetros, se generan gráficos en tiempo real que muestran cómo cambian las tensiones y posiciones. Esto proporciona retroalimentación inmediata sobre sus decisiones.

- Resultados instantáneos: Después de realizar ajustes, los usuarios recibirán resultados instantáneos sobre las tensiones calculadas y otras métricas relevantes relacionadas con el equilibrio.

- Ayuda contextual: Se incluye una sección de ayuda accesible desde cualquier parte del simulador, proporcionando información sobre cómo utilizar cada funcionalidad y explicaciones sobre conceptos físicos relevantes.

Esta estructura permite a los estudiantes experimentar con conceptos físicos complejos mientras reciben retroalimentación visual continua, mejorando su comprensión a través de la práctica activa.

7. Matriz de requisitos

La matriz a continuación proporciona una referencia cruzada que rastrea los componentes y estructuras de datos a los requisitos especificados en el documento SRS.

RF01	Interfaz Gráfica	Permite al usuario ingresar datos
RF02	Módulo de Cálculo	Realiza cálculos basados en entradas
RF03	Módulo Gráfico	Genera visualizaciones gráficas
RF04	Interfaz Gráfica	Proporciona retroalimentación visual
RF05	Módulo Gráfico	Actualiza gráficos según cambios

Esta matriz asegura que todos los requisitos funcionales estén claramente vinculados a sus respectivos componentes dentro del sistema, facilitando así su seguimiento durante el desarrollo e implementación del Simulador de Equilibrio.

8. Diagrama de clases.

8.1. Diagrama de clase interfaz.

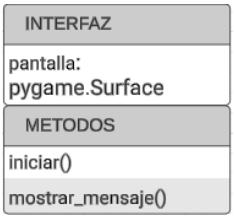

Figura 1. Diagrama de clase 1.

8.2. Diagrama de clase simulador.

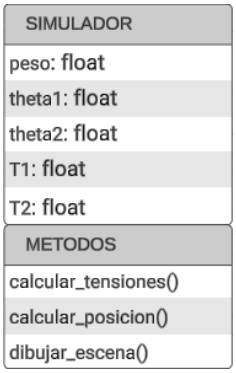

Figura 2. Diagrama de clase 2.

8.3. Diagrama de clase grafico.

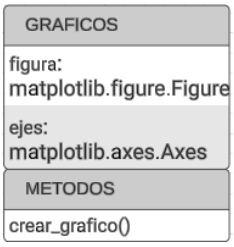

Figura 3. Diagrama de clase 3.

8.4. Detalle de Pseudocódigo

Calcular Tensiones

```
Calcular Tensiones
Funci n calcular_tensiones(peso, theta1, theta2):
Convertir theta1 y theta2 a radianes
Denominador = sin(theta1_rad + theta2_rad)
Si denominador es 0:
Retornar 0, 0
T1 = peso * sin(theta2_rad) / denominador
T2 = peso * sin(theta1_rad) / denominador
Retornar T1, T2
```

Dibujar Escena

```
Funci n dibujar_escena(pantalla, peso, theta1,
    theta2):
Limpiar pantalla con color blanco
Calcular tensiones usando calcular_tensiones(peso,
    theta1, theta2)
Calcular posici n del cuerpo usando
    calcular_posicion()
Dibujar im genes (cuerpos, poleas, cuerdas) en la
    pantalla
Mostrar tensiones y  ngulos  en la pantalla
```

Conversor

```
Funci n conversor(Kg):
retornar Kg * 9.81
```

Crear Gráfico

```
Funci n crear_grafico(peso, theta1, theta2):
Crear figura y ejes en matplotlib
Graficar cuerpo y vectores de fuerza (Peso, T1, T2)
Ajustar l mites y leyenda del gr fico
Retornar imagen para pygame
```

9. Estructura de datos

La estructura de datos utilizada en el Simulador de Equilibrio se organiza en varias funciones y variables que permiten manejar tanto la lógica del simulador como su representación gráfica. A continuación, se describen los principales elementos:

9.1. Simulador

Atributos	
peso: float	Representa el peso del objeto en estudio.
theta1: float	Primer ángulo en grados.
theta2: float	Segundo ángulo en grados.
T1: float	Tensión calculada en la cuerda 1.
T2: float	Tensión calculada en la cuerda 2
Métodos	
calcular_tensiones()	Calcula las tensiones basadas en los ángulos y el peso.
calcular_posicion()	Determina la posición del cuerpo basado en las tensiones.

9.2. Grafico

Atributos	
figura: matplotlib.figure.Figure	Almacena la figura del gráfico generado.
ejes: matplotlib.axes.Axes	Contiene los ejes para dibujar el gráfico.
Métodos	
crear_grafico()	Genera un gráfico que muestra las fuerzas actuantes sobre el cuerpo.

9.3. Interfaz

Atributos	
pantalla: pygame.Surface	Representa la superficie donde se dibujará el simulador.
Métodos	
iniciar()	Inicializa la interfaz gráfica y comienza el bucle principal del simulador.
mostrar_mensaje()	Muestra mensajes informativos al usuario.

10. Interfaces de los Componentes.

Las interfaces entre los componentes del sistema son cruciales para garantizar que cada parte funcione correctamente y se comunique eficazmente. A continuación, se describen las interfaces clave:

9.4. Otras implementaciones

Variables Globales	
WIDTH, HEIGHT	Definen las dimensiones de la ventana del simulador.
WHITE, BLACK, etc.	Colores utilizados para el fondo y otros elementos gráficos.
Variables para manejar el estado de arrastre del mouse, que permiten a los usuarios interactuar con los objetos en la pantalla.	
Parámetros Físicos	
Se definen variables como g (gravedad) y constantes relacionadas con el sistema físico que se simula, facilitando cálculos precisos durante la ejecución.	

Interfaz Gráfica (Interfaz):

- Esta interfaz actúa como punto de contacto entre el usuario y el simulador. Maneja eventos de entrada (como clics y arrastres) y actualiza la visualización basada en las interacciones del usuario.
- La interfaz gráfica está diseñada para ser intuitiva, permitiendo a los usuarios iniciar simulaciones, ajustar parámetros y ver resultados sin complicaciones.

Módulo de Cálculo (Simulador):

- Este módulo realiza todos los cálculos necesarios para determinar las tensiones y posiciones basadas en los parámetros ingresados por el usuario.
- Se comunica con la interfaz gráfica para proporcionar resultados que se mostrarán al usuario, como tensiones calculadas y posiciones de los cuerpos.

Módulo Gráfico (Grafico):

- Este módulo es responsable de generar gráficos utilizando Matplotlib. Convierte estos gráficos a un formato que puede ser utilizado por Pygame, permitiendo su visualización dentro del simulador.
- La comunicación entre este módulo y el módulo de cálculo es esencial, ya que necesita recibir datos sobre las tensiones y posiciones para crear representaciones gráficas precisas.

Interacción entre Componentes:

- Cuando un usuario ajusta un parámetro (por ejemplo, peso o ángulo), la interfaz gráfica llama al módulo de cálculo para actualizar las tensiones. Luego, esos resultados se envían al módulo gráfico para actualizar la visualización.
- Este flujo de información debe ser eficiente para asegurar que el simulador responda rápidamente a las acciones del usuario, manteniendo una experiencia fluida.

Referencias

1. Python Software Foundation.: Python. Python Software Foundation (2024).
2. Pygame Community.: Pygame Documentation (2022).
3. matplotlib development team.: Matplotlib Documentation (2022).
4. Tkinter Documentation.: Canvas framework within the Tkinter library (2022).
5. Zemasnki, S.: Física Universitaria con Física Moderna (2018).
6. Sauro, J., Lewis, J.R.: Quantifying the User Experience: Practical Statistics for User Research. Elsevier, Burlington (2012). https://doi.org/10.1016/B978-0-12- 384968-7.00008-4