Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 2do. Semestre 2024

PROYECTO 2

202105658 – Brayan Alexander Guzman Margos

Resumen

El proyecto tiene como objetivo desarrollar una solución de software para simular el funcionamiento de una máquina de ensamblaje creada por la empresa Digital Intelligence, S.A. La máquina cuenta con múltiples líneas de ensamblaje y brazos robóticos que ensamblan componentes de productos siguiendo una secuencia de instrucciones. Se debe implementar una solución en Python que utilice estructuras de programación secuenciales, cíclicas y condicionales, así como conceptos de programación orientada a objetos y tipos de datos abstractos (TDA) propios.El software debe cargar archivos XML que configuran las máquinas y sus productos, simular el ensamblaje, y generar reportes en formato HTML y gráficos utilizando Graphviz. La interfaz web en Flask permitirá gestionar acciones como cargar archivos, seleccionar productos para simular su construcción, y mostrar los tiempos óptimos de ensamblaje. Además, se requiere versionamiento del código en GitHub y documentación detallada del proyecto, incluyendo diagramas de clases y actividades, para su evaluación final.

.

Palabras clave

- *Ensamblaje*
- *Interfaz*
- *Simulación*

Abstract

The project aims to develop a software solution to simulate the operation of an assembly machine created by the company Digital Intelligence, S.A. The machine has multiple assembly lines and robotic arms that assemble product components following a sequence of instructions. A Python solution must be implemented using sequential, cyclic and conditional programming structures, as well as object-oriented programming concepts and proprietary abstract data types (ADTs). The software must load XML files that configure the machines and their products, simulate the assembly, and generate reports in HTML format and graphs using Graphviz. The web interface in Flask will allow managing actions such as uploading files, selecting products to simulate their construction, and displaying optimal assembly times. In addition, versioning of the code on GitHub and detailed documentation of the project, including class and activity diagrams, are required for final evaluation.

Keywords

- *Assemblj*
- Interface
- *Simulation*

Introducción

Este proyecto se centra en la creación de un software que simule el funcionamiento de esta máquina, utilizando herramientas de desarrollo en Python y estructuras de datos abstractas. A través de una interfaz web, los usuarios podrán cargar configuraciones de máquinas y productos, simular el proceso de ensamblaje y generar reportes detallados que muestren el tiempo óptimo requerido para la fabricación. El objetivo es proporcionar una solución integral que facilite la gestión y simulación de procesos de ensamblaje, contribuyendo a una mayor comprensión y control de la producción industrial. Además, se busca aplicar conceptos de programación orientada a objetos y manipulación de archivos XML para abordar el problema de manera efectiva.

Desarrollo del tema

El proyecto requerido es un programa en Python que implementa una solución basada en Programación Orientada a Objetos (POO) para gestionar matrices almacenadas en un archivo XML. Además, permite la manipulación de estos datos, incluyendo su procesamiento, exportación y visualización gráfica mediante Graphviz. A continuación, se presenta un desarrollo del tema con cuatro subtemas que explican los componentes y funcionalidades clave del programa.

a) Clase Nodo

Representa un nodo en una lista enlazada, que es una estructura de datos clave para almacenar los archivos y productos de manera dinámica.

```python
class Nodo:
    def __init__(self, data):
        self.data = data
        self.siguiente = None
```

b) Clase ListaEnlazada

Gestiona la inserción y obtención de nodos, facilitando la administración de datos en memoria dinámica.

```python
class ListaEnlazada:
    def __init__(self):
        self.head = None

    def insertar(self, data):
        nuevo_nodo = Nodo(data)
        if not self.head:
            self.head = nuevo_nodo
        else:
            actual = self.head
            while actual.siguiente:
                actual = actual.siguiente
            actual.siguiente = nuevo_nodo

    def obtener_lista(self):
        lista = []
        actual = self.head
        while actual:
            lista.append(actual.data)
            actual = actual.siguiente
        return lista
```

c) Clase XMLManager

Maneja la carga y almacenamiento de archivos XML, permitiendo al usuario importar configuraciones de máquinas y productos desde archivos.

```python
import xml.etree.ElementTree as ET

class XMLManager:
    @staticmethod
    def cargar_maquinas(archivo):
        tree = ET.parse(archivo)
        root = tree.getroot()
        maquinas = []
        for maquina in root.findall('Maquina'):
            nombre = maquina.find('NombreMaquina').text
            cantidad_lineas = int(maquina.find('CantidadLineasProduccion').text)
            cantidad_componentes = int(maquina.find('CantidadComponentes').text)
            tiempo_ensamble = int(maquina.find('TiempoEnsamblaje').text)
            productos = [p.find('nombre').text for p in maquina.find('ListadoProductos')]
            maquinas.append((nombre, cantidad_lineas, cantidad_componentes, tiempo_ensambl
        return maquinas
```

d) Clase Producto

Representa un producto a ensamblar, conteniendo su nombre y la secuencia de ensamblaje.

```python
class Producto:
    def __init__(self, nombre, secuencia):
        self.nombre = nombre
        self.secuencia = secuencia
```

e) Clase MachineManager

Administra la carga de máquinas desde archivos XML y su almacenamiento en memoria.

```python
class MachineManager:
    def __init__(self):
        self.maquinas = []

    def cargar_maquinas(self, archivo):
        maquinas = XMLManager.cargar_maquinas(archivo)
        for nombre, lineas, componentes, tiempo, productos in maquinas:
            maquina = Maquina(nombre, lineas, componentes, tiempo)
            for p in productos:
                producto = Producto(p, [])
                maquina.agregar_producto(producto)
            self.maquinas.append(maquina)
```

f) Clase AssemblySimulator

La clase AssemblySimulator realiza la simulación del ensamblaje y calcula el tiempo total y los pasos a seguir.

```python
class AssemblySimulator:
    def __init__(self, maquina):
        self.maquina = maquina

    def simular(self, producto):
        tiempo_total = 0
        pasos = []
        for i, componente in enumerate(producto.secuencia):
            tiempo_movimiento = (i + 1)  # Movimiento al componente
            tiempo_ensamble = self.maquina.tiempo_ensamble
            tiempo_total += tiempo_movimiento + tiempo_ensamble
            pasos.append(f"{tiempo_total} segundos: Ensamblar {componente}")
        return tiempo_total, pasos
```

Conclusiones

El desarrollo del software para simular el funcionamiento de la máquina ensambladora de Digital Intelligence, S.A. representa un avance significativo en la optimización de procesos industriales. A través de la implementación de una solución en Python, se ha logrado integrar diversos conceptos de programación, como estructuras de datos abstractas y la programación orientada a objetos, lo que permite una gestión eficiente de los recursos y una fácil manipulación de la información. La interfaz web diseñada facilita la interacción del usuario con el sistema, permitiendo cargar configuraciones y simular el ensamblaje de productos, así como generar reportes detallados y gráficos del proceso. Estos reportes no solo proporcionan una visión clara del tiempo requerido para cada producto, sino que también contribuyen a la toma de decisiones informadas en la gestión de la producción. En definitiva, este proyecto no solo cumple con los objetivos planteados, sino que también sienta las bases para futuras mejoras y adaptaciones en el campo de la automatización industrial.permite agrupar patrones de acceso similares, facilitando una distribución de datos que se ajusta mejor a las necesidades de la red, reduciendo costos de comunicación y maximizando la eficiencia del sistema en su conjunto.

Apendices

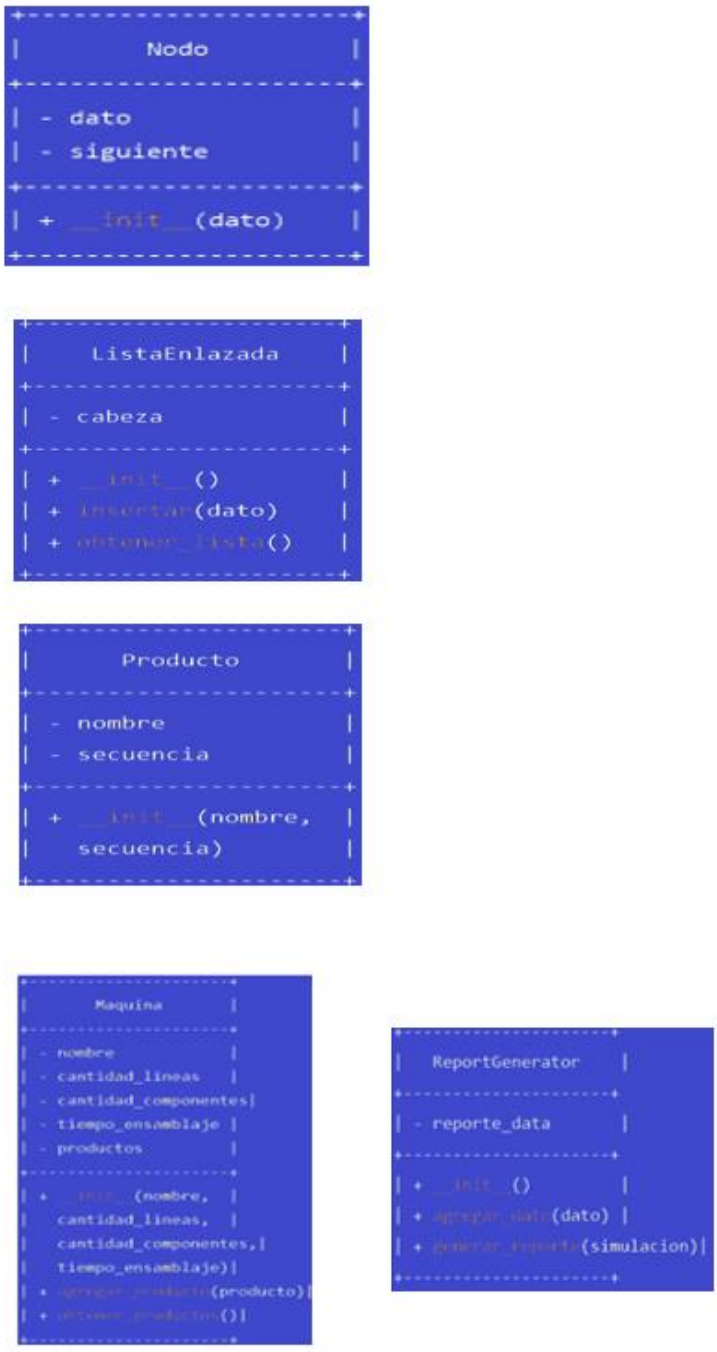

Diagrama de clases

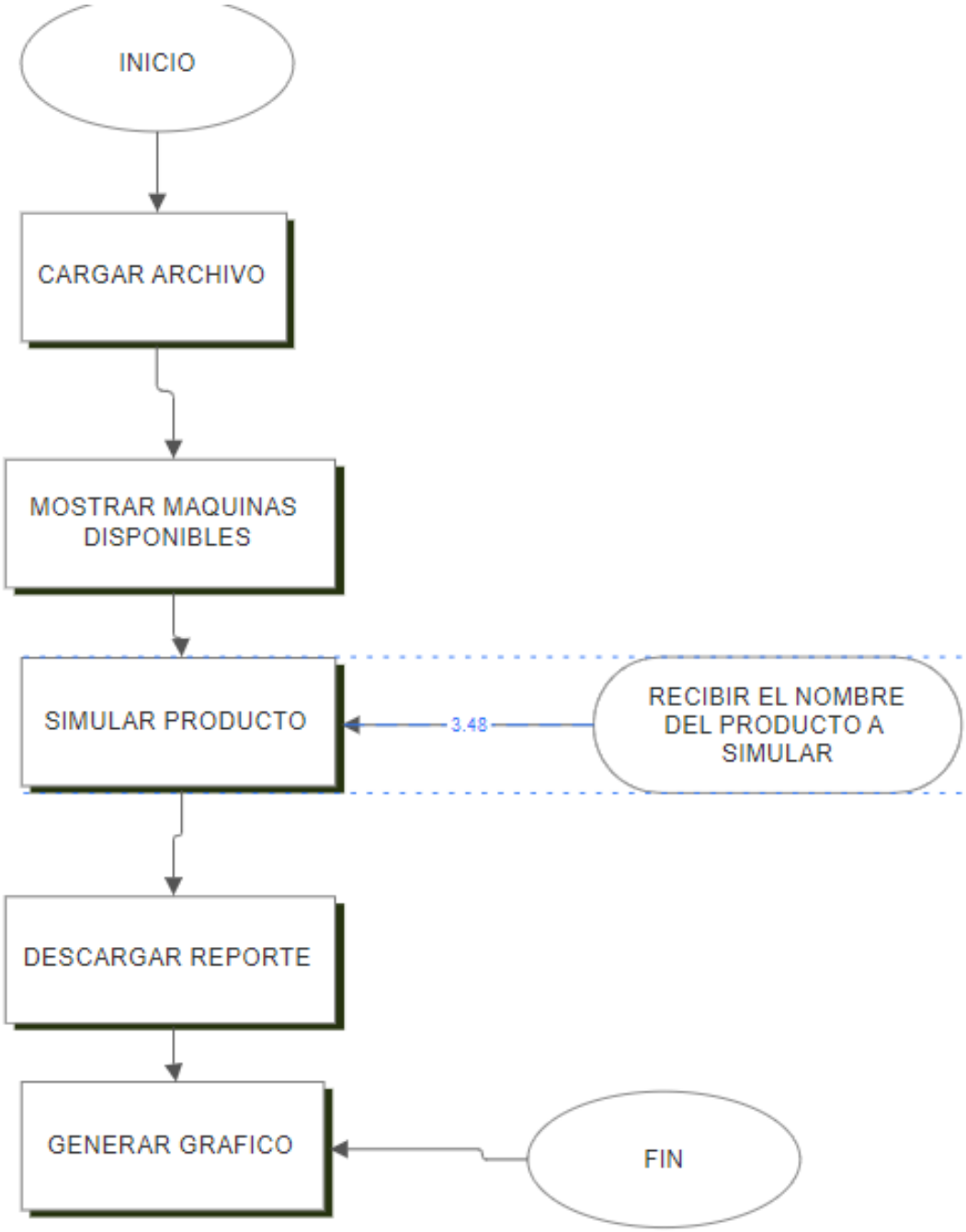

Diagrama de actividades