Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2024

SISTEMA DE OPTIMIZACION DE MOVIMIENTOS UTILIZANDO ARCHIVOS XML EN PYTHON

202300848 – Brayan Emanuel Garcia

Resumen

El sistema inicia leyendo archivos XML que contienen datos sobre máquinas y productos, utilizando la clase **LecturaXML** para convertirlos en estructuras manejables en Python, como listas enlazadas. Estas listas permiten una gestión eficiente de grandes volúmenes de datos. Posteriormente, se analizan las instrucciones almacenadas para su procesamiento, incluyendo el tiempo de ensamblaje y el número de líneas en cada máquina.

Una característica destacada del proyecto es la generación de grafos que representan visualmente las instrucciones procesadas, lo que facilita la interpretación y análisis. La simulación permite emparejar instrucciones con movimientos de ensamblaje, registrando cada acción en un archivo HTML.

Palabras clave

Máximo cinco palabras que servirán para identificar el estudio realizado.

Abstract

*The system begins by reading XML files containing data about machines and products, using the **LecturaXML** class to convert them into manageable structures in Python, such as linked lists. These lists allow for efficient management of large volumes of data. Subsequently, the stored instructions are analyzed for processing, including assembly time and the number of lines in each machine.*

A notable feature of the project is the generation of graphs that visually represent the processed instructions, facilitating interpretation and analysis. The simulation allows for pairing instructions with assembly movements, recording each action in an HTML file.

Keywords

Matrices, XML, Python, data processing, graphs.

Introducción

El análisis y procesamiento de grandes cantidades de datos es un tema de creciente relevancia en el contexto tecnológico actual. Los datos estructurados, que se almacenan en formatos como XML (Extensible Markup Language), permiten una organización clara y jerárquica de la información, facilitando su manipulación y análisis. Este ensayo aborda el desarrollo de un sistema basado en Python que tiene como objetivo principal leer, procesar y reducir matrices provenientes de archivos XML. La implementación de este sistema no solo permite mejorar la eficiencia en la manipulación de datos, sino que también ofrece una solución flexible y adaptable a diferentes escenarios. La importancia de este proyecto radica en su aplicación práctica en áreas como el análisis de datos, la optimización de recursos y la visualización de información a través de grafos, proporcionando una herramienta robusta para la investigación académica y la industria.

Marco teorico

Para facilitar la comprensión del ensayo, a continuación ofrezco algunas definiciones de términos que podrían no ser tan comunes para personas que no están familiarizadas con el ámbito de la tecnología:

XML (eXtensible Markup Language): Es un formato de texto utilizado para almacenar y transportar datos de forma estructurada. A menudo se usa para compartir información entre diferentes sistemas o programas porque es fácil de leer para las computadoras, pero también comprensible para los humanos. A diferencia de otros formatos de almacenamiento de datos, como bases de datos o archivos binarios, XML organiza los datos en una jerarquía mediante etiquetas.

Lista enlazada: Es una estructura de datos que se utiliza para almacenar una colección de elementos. A diferencia de un arreglo (o array), donde los elementos están almacenados en posiciones contiguas de memoria, en una lista enlazada cada elemento (o "nodo") contiene una referencia al siguiente, lo que permite añadir o eliminar elementos de forma dinámica y eficiente sin necesidad de reorganizar toda la estructura.

Grafo: Es una representación visual y matemática que describe relaciones entre elementos. Está formado por nodos (también llamados vértices) que representan objetos o entidades, y aristas (o conexiones) que muestran las relaciones o interacciones entre esos nodos. Los grafos se usan mucho en tecnología, por ejemplo, para representar redes sociales (donde los nodos son personas y las aristas son sus conexiones).

Simulación: En el contexto tecnológico, una simulación es un modelo computarizado que reproduce el comportamiento de un sistema real para estudiar su funcionamiento. En este proyecto, se simula el ensamblaje de productos en una línea de producción, permitiendo evaluar el rendimiento y optimizar el proceso sin tener que realizarlo físicamente.

Backend: Es la parte del software que no se ve ni interactúa directamente con el usuario, pero que realiza las operaciones fundamentales para que una aplicación funcione correctamente. El backend incluye bases de datos, servidores y la lógica que procesa las peticiones del usuario. En este caso, el backend del proyecto gestiona la carga de archivos, el procesamiento de instrucciones y la ejecución de simulaciones.

Interfaz gráfica (GUI, por sus siglas en inglés): Es el medio visual a través del cual los usuarios interactúan con un programa. Incluye botones, ventanas, menús y otros elementos gráficos que permiten al usuario utilizar el software de manera intuitiva, en lugar de tener que escribir comandos de texto.

Algoritmo de matcheo: En este caso, se refiere a un procedimiento que empareja o asocia instrucciones con los movimientos correspondientes de las líneas de ensamblaje. Es una técnica utilizada para coordinar diferentes acciones y asegurar que el ensamblaje de un producto ocurra de la manera correcta y en el orden adecuado.

Cola: En programación, una cola es una estructura de datos que sigue el principio de "primero en entrar, primero en salir" (FIFO por sus siglas en inglés). Imagina una fila de personas esperando su turno; la primera persona en la fila será la primera en ser atendida, y así sucesivamente. En este sistema, una cola se utiliza para gestionar las instrucciones que se van procesando durante la simulación.

Desarrollo del tema

Subtema 1: Lectura y Procesamiento de Archivos XML

El sistema inicia con la lectura de archivos XML que contienen maquinas y productos. Para este propósito, se emplea la clase LecturaXML, que extrae los datos almacenados en los archivos y los convierte en estructuras manejables en Python, como listas enlazadas. La elección de listas enlazadas para el almacenamiento de matrices permite una gestión eficiente de grandes volúmenes de datos, ya que facilita la inserción y eliminación de elementos sin necesidad de reestructurar la memoria. Este enfoque es particularmente útil para la manipulación dinámica de matrices, optimizando el rendimiento general del sistema.

Subtema 2: Análisis de instrucciones

Cuando se crea la estructura de datos se pasan a una lista de caracteres y se almacenan las instrucciones en producto.elaboracion. Esta se usa en procesarInstruccion para poder buscar las instrucciones en el programa, también almacenamos el tiempo que la maquina se tarda en ensamblar, cuantas lineas cuenta el archivo y también para cada maquina.

Subtema 3: Generación de Grafos

Uno de los aspectos más interesantes del proyecto es la generación de grafos, que representan visualmente las instrucciones procesadas. Utilizando la clase Grafo, el sistema crea nodos y aristas que facilitan la visualización de la estructura interna de las instrucciones. Esta representación gráfica es particularmente útil para el análisis y la toma de decisiones en contextos donde la visualización de datos facilita la interpretación y comprensión de instrucciones complejas..

Subtema 4: Simulación:

La simulación busca hacer matchear la instrucción en cola junto a los movimientos de las líneas de ensamblaje con los componentes y sus posiciones, al matchear se empieza el ensamblaje de esa pieza, luego de eso, cada movimiento se registra en un html que se crea a partir de cada instrucción, segundo, leído.

Subtema 5: Interfaz grafica:

La interfaz grafica hecha con html, css y un poco de Python para traer la información a tomar, empieza con pedirle al usuario que suba un archivo, luego de eso lo analiza, y nos permite la elección de productos y maquina, al seleccionarlas podemos presionar simulación, se llenara el espacio con un grafo que son de las instrucciones y con una tabla que describe de manera detallada el proceso optimizado, esto se guardara en un historial que podemos ver yéndonos a un el apartado de reportes.

Desde el apartado de reportes podemos hacer un informe de todas las maquinas y los productos, así como ver el historial de ensamblaje. Por ultimo el apartado de ayuda, que nos lleva a la información del app y su repositorio.

Conclusiones

El sistema de simulación de ensamblaje de productos combina múltiples tecnologías y enfoques teóricos para ofrecer una solución integral al problema de gestionar y optimizar líneas de producción. Desde la lectura de archivos XML, el análisis y almacenamiento de instrucciones, hasta la generación de grafos y la simulación, cada parte del sistema está diseñada para maximizar la eficiencia y facilitar el análisis del proceso de ensamblaje. La interfaz gráfica, además, permite a los usuarios interactuar con el sistema de manera intuitiva, accediendo a información clave y generando reportes detallados de manera sencilla. En conjunto, estos elementos permiten no solo la automatización del ensamblaje, sino también la evaluación continua y la mejora del rendimiento del sistema.

Apéndices

Apéndice A: Ejemplo de Archivo XML

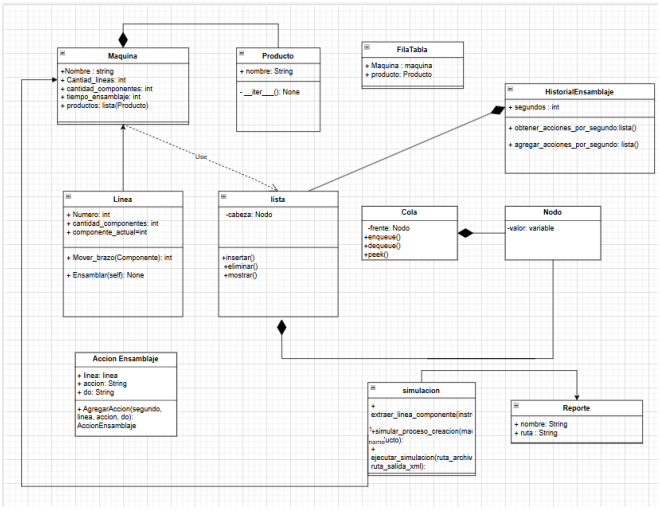
ntrada.xml

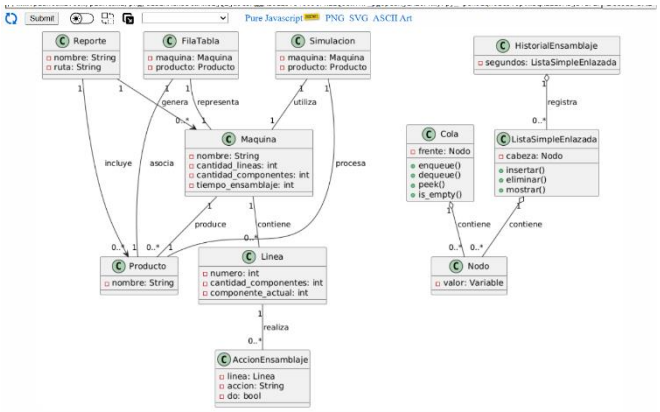
```
<?xml version="1.0"?>
<ListaMaquinas>
 <Maquina>
 <NombreMaquina> [Texto] </NombreMaquina>
 <CantidadLineasProduccion> n
</CantidadLineasProduccion>
 <CantidadComponentes> m </CantidadComponentes>
 <TiempoEnsamblaje> x </TiempoEnsamblaje>
 <ListadoProductos>
 <Producto>
 <nombre> [Texto] </nombre>
 <elaboracion>
 [Texto]1
 </elaboracion>
 </Producto>
 …
 </ListadoProductos>
 </Maquina>
 …
</ListaMaquinas>
```

Apéndice B: Codigo de clase ejecutar.py

```
def ejecutar_simulacion(ruta_archivo_xml,
ruta_salida_xml):
   maquinas = analizarArchivo(ruta_archivo_xml)
   actual_maquina = maquinas.cabeza

   while actual_maquina:
      maquina = actual_maquina.valor
      print(f"Simulando para la máquina:
{maquina.nombre}")

      actual_producto = maquina.productos.cabeza
      while actual_producto:
         producto = actual_producto.valor
         print(f"Producto: {producto.nombre}")
         simular_proceso_creacion(maquina, producto)

         actual_producto = actual_producto.siguiente

      actual_maquina = actual_maquina.siguiente

   generar_salida_xml(maquinas, ruta_salida_xml)
```

Apéndice C: Diagramas de clases





Apéndice D: Diagrama de actividades