PROYECTO #2

20201005 – Derek Esquivel Díaz

Resumen

El presente proyecto es un programa que simula el funcionamiento de la maquina creada por Digital Intelligence para predecir el tiempo utilizado para elaborar un producto y así optimizar el proceso.

Para realizar esto primero es necesario crear la máquina, esto se hace al leer un archivo XML que contiene las características de la máquina. En este archivo se enumeran las listas de producción y se menciona cuanto tiempo tarda en ensamblar y cuantos componentes tiene. También se debe enlistar los productos que puede elaborar la maquina seguido por los pasos para elaborarlos.

Una vez leída la maquina se ingresa un archivo de simulación, que incluirá los productos que se pueden elaborar en la simulación actual. El usuario podrá elegir cualquiera de estos productos para empezar la elaboración.

La elaboración se hace por medio de un algoritmo que separa las instrucciones por línea de elaboración y luego las ejecuta simultáneamente, tomando en cuenta el orden con el cual se debe de ensamblar, así dar con el proceso de elaboración optimo.

Palabras clave

Lista Enlazada; Python; Tipos de Datos Abstractos; Matriz Dispersa

Abstract

This project is a program that simulates the operation of the machine created by Digital Intelligence to predict the time used to produce a product and then be able to optimize the process.

To do this, it is necessary to create the machine, this is done by reading an XML file that contains the characteristics of the machine. This file contains the assembly lines of the machine and mentions how long it takes each one of them to assemble a product and how many components it contains. The products that the machine can make are also listed, followed by the steps to make each one of them.

Once the machine is read, a simulation file is entered, which will include the products that can be produced in the current simulation. The user will be able to choose any of these products to start the elaboration process.

The elaboration process is done by an algorithm that separates the instructions by assembly line and then executes them simultaneously, considering the order in which the components must be assembled, thus finding the optimal elaboration process.

Keywords

Linked List, Python, Abstract Data Types; Sparse Matrix

Introducción

El presente proyecto es un programa que simula el funcionamiento de la maquina creada por Digital Intelligence para predecir el tiempo utilizado para elaborar un producto y así optimizar el proceso.

Esto se hace al subir 2 archivos al programa, uno que contiene las características de la máquina, aquí se incluyen las listas de elaboración y los productos que puede construir.

El segundo archivo es un el archivo de la simulación actual, aquí se incluyen los productos que se podrán construir durante esta simulación.

Una vez ambos archivos han sido cargados al sistema se podrá simular la elaboración optima de uno de estos productos, esto se hace por medio de un algoritmo que separa las instrucciones por línea de elaboración y luego las ejecuta simultáneamente, tomando en cuenta el orden con el cual se debe de ensamblar, así dar con el proceso de elaboración optimo.

Desarrollo del tema

La primera parte del programa consiste en leer los archivos XML con las características de la maquina y de la simulación. Primero se deberá leer el de la maquina.

El archivo XML consiste en las siguientes partes:

- **Cantidad de líneas de producción**: Esta es la etiqueta raíz que indica el número de líneas que contendrá la maquina
- **Listado de líneas de producción**: Esta etiqueta contiene las líneas de producción de la maquina con sus características, estas características pueden ser:
    - **Numero:** Esta etiqueta servirá como el identificador de la línea de producción
    - **Cantidad de Componentes**: Contiene el numero de componentes que contiene la maquina
    - **Tiempo de Ensamblaje**: Este es el tiempo en segundos que tardara la línea en ensamblar un componente

- **Listado de Productos**: Esta etiqueta contendrá una lista con los productos que puede construir cada máquina, cada producto debe incluir:
    - **Nombre:** Nombre con el cual será identificado el proyecto (debe ser único)
    - **Elaboración:** Pasos que deberán seguir las líneas de elaboración para construir el producto

```
<?xml version="1.0"?>
<Maquina>
    <CantidadLineasProduccion>2</CantidadLineasProduccion>
    <ListadoLineasProduccion>
        <LineaProduccion>
            <Numero>1</Numero>
            <CantidadComponentes>5</CantidadComponentes>
            <TiempoEnsamblaje>1</TiempoEnsamblaje>
        </LineaProduccion>
        <LineaProduccion>
            <Numero>2</Numero>
            <CantidadComponentes>5</CantidadComponentes>
            <TiempoEnsamblaje>1</TiempoEnsamblaje>
        </LineaProduccion>
    </ListadoLineasProduccion>
    <ListadoProductos>
        <Producto>
            <nombre>SmartWatch</nombre>
            <elaboracion>L1C2 L2C1 L2C2 L1C4</elaboracion>
        </Producto>
        <Producto>
            <nombre>SmartPhone</nombre>
            <elaboracion>L1pC3p L2pC2p L1pC1p L2pC4p</elaboracion>
        </Producto>
    </ListadoProductos>
</Maquina>
```

Figura 1: Estructura del archivo de entrada de la maquina
Fuente: Elaboración propia (2021)

El segundo archivo que se subirá será el archivo de simulación, este archivo contendrá que productos se podrán elaborar durante la simulación actual.

- Nombre: Este será el nombre con el que se guardará la simulación
- Producto: Aquí se almacenará el nombre del producto que se podrá producir en la simulación

Una vez leídos los archivos XML se utiliza la API "ElementTree" para leer las etiquetas y valores del archivo para así poder generar con objeto tipo "Maquina", este archivo contiene listas que representan cada producto que se puede realizar, donde se especifica el nombre del producto y su elaboración.

```
140  def leerSimulacion():
141      global listadoProductos
142      listadoProductos = LinkedList()
143
144      filename = askopenfilename()
145      tree = ET.parse(filename)
146      root = tree.getroot()
147
148      global NombreSimulacion
149      NombreSimulacion = root.find('Nombre').text
150
151      for linea in root.findall("ListadoProductos"):
152          for producto in linea.findall("Producto"):
153
154              listadoProductos.insertar(producto.text)
155
```

Figura 2: Código para leer el archivo XML de la simulación.
Fuente: Elaboración propia (2021)

Una vez leídos ambos archivos se mostraran los datos de la maquina en la interfaz y los productos de la simulacion se mostraran en una lista donde el usuario podrá seleccionar cual elaborar.

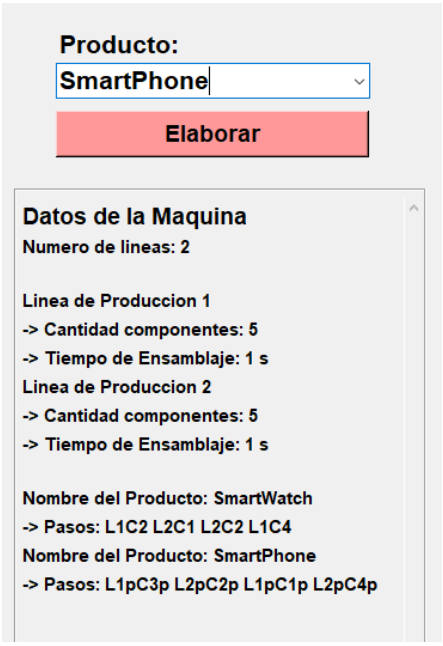

Figura 3: Datos de la maquina en la interfaz.
Fuente: Elaboración propia (2021)

Antes de poder elaborar el producto el programa debe procesar las instrucciones de elaboración, esto se realizara con un automata finito determinista para separar las instrucciones y asignárselas a cada línea de produccion. El diagrama del automata es el siguiente:

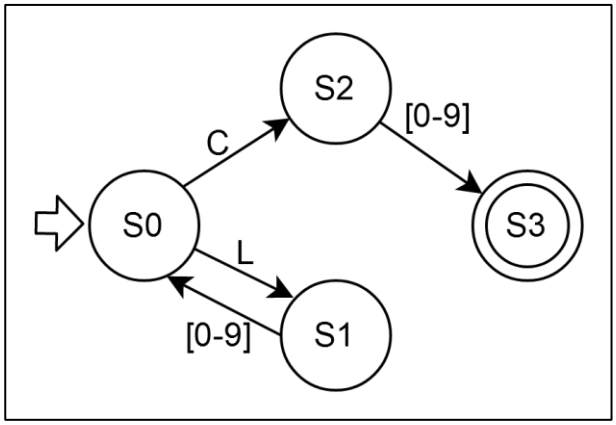

Figura 4: Diagrama del automata.
Fuente: Elaboración propia (2021)

El código de autómata se inicializara con el siguiente código:

```
7      def __init__(self, producto):
8          self.columna = 1
9          self.buffer = ''
10         self.estado = 0
11         self.i = 0
12         self.IntruccionesPorLinea = LinkedList()
13         self.producto = producto
14         self.lineaActual = None
```

Figura 5: Constructor del autómata.
Fuente: Elaboración propia (2021)

Luego el automata leerá las instrucciones carácter por carácter, separando las instrucciones por línea de produccion, así cada línea tendrá solamente una lista con las instrucciones que llevara a cabo para ese producto.

Una vez se han separado las instrucciones se realiza una lista que llevara el control el orden de ensamblaje, pues, aunque cada línea se ejecuta individualmente todas deberán seguir el orden de ensamblaje. Esta lista se genera con el siguiente código:

```
    def getOrden(self):
        lista = LinkedList()
        lista_re = re.findall(r"L[0-9]", self.elaboracion)

        i = 0
        while i < len(lista_re):
            lista.insertar(lista_re[i].replace("L","0"))
            i += 1

        return lista
```

Figura 6: generación del orden de ensamblaje de los componentes.
Fuente: Elaboración propia (2021)

Ya con el orden de ensamblaje establecido se procede a realizar el algoritmo de elaboración, este se ejecuta realizando dos ciclos simultáneos, primero tendrá uno que recorrerá cada línea de elaboración, haciendo la instrucción que le corresponde a cada línea. Esto se repetirá hasta que ninguna de las listas tenga mas instrucciones pendientes.

Para ensamblar un componente la línea se debe de mover a la posición del componente, el brazo se moverá con el siguiente código:

```
elif lineaActual.valor.parar is False:

    objetivo = lineaActual.valor.peek()
    objetivo_int = int(objetivo.valor.replace('C',""))


    if lineaActual.valor.componente_actual < objetivo_int:
        lineaActual.valor.componente_actual += 1
        print("Linea: ", str(lineaActual.valor.lineaProduccion) ," - Mover Brazo -> Componente ", str(lineaActual.valor.componente_actual))
        lsPasoActual.insertar(("Mover Brazo -> Componente " + str(lineaActual.valor.componente_actual)), str(lineaActual.valor.lineaProduccion))

        if lineaActual.valor.componente_actual == objetivo_int:
            lineaActual.valor.parar = True
            lineaActual.valor.ensamblando = int(lineaActual.valor.costo)

    elif lineaActual.valor.componente_actual > objetivo_int:
        lineaActual.valor.componente_actual -= 1
        print("Linea: ", str(lineaActual.valor.lineaProduccion) ," - Mover Brazo -> Componente ", str(lineaActual.valor.componente_actual))
        lsPasoActual.insertar(("Mover Brazo -> Componente " + str(lineaActual.valor.componente_actual)), str(lineaActual.valor.lineaProduccion))

        if lineaActual.valor.componente_actual == objetivo_int:
            lineaActual.valor.parar = True
            lineaActual.valor.ensamblando = int(lineaActual.valor.costo)
```

Figura 7: Parte del código para mover el brazo hacia adelante y atrás.
Fuente: Elaboración propia (2021)

Una vez el brazo se encuentra sobre el proyecto procederá a ensamblar, mientras una línea esta ensamblando todas las demás deberán parar hasta que esta termine. Algunas líneas pueden tardar más de un segundo en ensamblar, por lo que esto también se tendrá en cuenta. El código para ensamblar es:

```
elif lineaActual.valor.parar and lineaActual.valor.lineaProduccion == int(ensambleActual.valor):
    lineaActual.valor.ensamblando -= 1
    if int(lineaActual.valor.ensamblando) == 0:
        lineaActual.valor.dequeue()
        print("Linea: ", str(lineaActual.valor.lineaProduccion) ," - Ensamblando -> Componente ", str(lineaActual.valor.componente_actual))
        lsPasoActual.insertar(("Ensamblando -> Componente " + str(lineaActual.valor.componente_actual)), str(lineaActual.valor.lineaProduccion))
        lineaActual.valor.parar = False
        done = True

    else:
        print("Linea: ", str(lineaActual.valor.lineaProduccion) ," - Ensamblando -> Componente ", str(lineaActual.valor.componente_actual))
        lsPasoActual.insertar(("Ensamblando -> Componente " + str(lineaActual.valor.componente_actual)), str(lineaActual.valor.lineaProduccion))
```

Figura 8: Código del proceso de elaboración.
Fuente: Elaboración propia (2021)

Una vez se terminó un segundo se deberá realizar una comprobación para ver si aun hay instrucciones pendientes, si ya no hay instrucciones se detendrá el algoritmo y se enviará la lista a la interfaz.

```
    stop = True
    comprobacion = self.listaElaboracion.head
    while comprobacion is not None:
        if comprobacion.valor.empty() is False:
            stop = False
        comprobacion = comprobacion.siguiente
    if stop:
        terminado = True

    listaPasos.insertar(lsPasoActual)
```

Figura 9: comprobación si ya se terminó la elaboración.
Fuente: Elaboración propia (2021)

La lista de pasos realizados se envia a la interfaz para mostrar los pasos por línea que se realizaron para construir el producto.

Estos pasos se mostrarán en una tabla, donde cada columna representara una línea de producción y cada fila representa un segundo de la elaboración.

```
lbl = tk.Label(frame_buttons, text=("Producto: " + nombre), font = "Arial 16 bold",  anchor="nw", justify="left")
lbl.grid(row=0,column=0, columnspan=largo)

lbl = tk.Label(frame_buttons, text=("Segundo"), font = "Arial 12 bold",  anchor="w")
lbl.grid(row=1,column=0)

noLinea = 1
aux = pasos.head.valor.head
aux = aux.siguiente
while aux is not None:
    lbl = tk.Label(frame_buttons, text=("Linea " + str(aux.linea)), font = "Arial 12 bold",  anchor="w")
    lbl.grid(row=1,column=noLinea)
    noLinea += 1
    aux = aux.siguiente

noFila = 2
noColumna = 0

fila = pasos.head
while fila is not None:
    columna = fila.valor.head
    while columna is not None:

        lbl = tk.Label(frame_buttons, text=(columna.valor), font = "Arial 12",  anchor="w")
        lbl.grid(row=noFila,column=noColumna, pady=10, padx=8)
        noColumna += 1
        columna = columna.siguiente

    noFila += 1
    noColumna = 0
    fila = fila.siguiente
```

Figura 10: Creación de la tabla de pasos.
Fuente: Elaboración propia (2021)

La tabla de pasos ya completada se ver de la siguiente manera:

Producto: SmartWatch

Segundo	Linea 1	Linea 2
1	Mover Brazo -> Componente 1	Mover Brazo -> Componente 1
2	Mover Brazo -> Componente 2	Mover Brazo -> Componente 2
3	Ensamblando -> Componente 2	Detenida
4	Ensamblando -> Componente 2	Detenida
5	Mover Brazo -> Componente 3	Mover Brazo -> Componente 3
6	Mover Brazo -> Componente 4	Mover Brazo -> Componente 4
7	Ensamblando -> Componente 4	Detenida
8	Ensamblando -> Componente 4	Detenida
9	Mover Brazo -> Componente 3	Mover Brazo -> Componente 5
10	Detenida	Mover Brazo -> Componente 6
11	Detenida	Ensamblando -> Componente 6
12	Detenida	Ensamblando -> Componente 6

Figura 11: Tabla de la elaboración del producto.
Fuente: Elaboración propia (2021)

Una vez se ha procesado un archivo se podrán generar tres tipos de reportes, reportes HTML y XML que tendrán la lista de pasos realizados por cada línea y un reporte realizado en GRAPHVIZ que mostrara un diagrama con las instrucciones que se utilizaron para elaborar dicho producto.

Para elaborar el reporte HTML se concatenaron los valores usados para la tabla de la interfaz en una cadena de texto con etiquetas HTML para poder mostarlas en una tabla, apoyándose del toolkit Bootstrap para darle diseño a esta tabla.

```
fila = self.pasos.head
while fila is not None:
    html += '<tr style="height: 18px;">'
    columna = fila.valor.head
    while columna is not None:
        html += '<td>' + columna.valor +'</span></td>'
        columna = columna.siguiente
    html += '</tr>'
    fila = fila.siguiente
```

Figura 12: Fragmento de código donde se crean las celdas del reporte HTML. Fuente: Elaboración propia (2021)

Para generar el reporte XML se puede utilizar una libraría, en este caso la librería que se utilizara se llama “LXML”, con esta librería podremos crear un objeto XML al cual le podremos agregar valores que serán colocados acorde a su raíz por la librería.

```
def generarXml(self):
    root = etree.Element('SalidaSimulacion')
    tree = etree.ElementTree(root)

    nombreSimulacion = etree.Element('Nombre')
    nombreSimulacion.text = str(self.NombreSimulacion)
    root.append(nombreSimulacion)

    listadoProductos = etree.Element('ListadoProductos')
    root.append(listadoProductos)

    producto = etree.Element('Producto')
    listadoProductos.append(producto)

    nombreProducto = etree.Element('Nombre')
    nombreProducto.text = str(self.nombre)
    producto.append(nombreProducto)

    tiempo = self.pasos.largo + 1
    tiempoTotal = etree.Element('TiempoTotal')
    tiempoTotal.text = str(tiempo)
    producto.append(tiempoTotal)

    ElaboracionOptima = etree.Element('ElaboracionOptima')
    producto.append(ElaboracionOptima)
```

Figura 13: Fragmento de código donde se elabora el reporte XML.
Fuente: Elaboración propia (2021)

Por ultimo esta el grafico de GRAPHVIZ donde se muestran las instrucciones realizadas por la maquina. Para esto se concatenan los valores en una cadena con la estructura de GRAPHVIZ, este cadena se guardara en un archivo .dot, el cual puede ser convertido a PNG posteriormente.

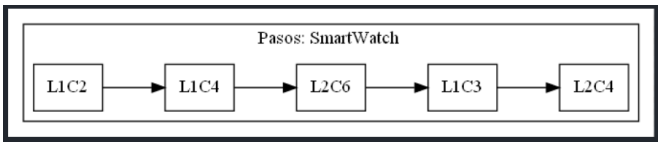

Figura 14: Reporte de cola de secuencia.
Fuente: Elaboración propia (2021)

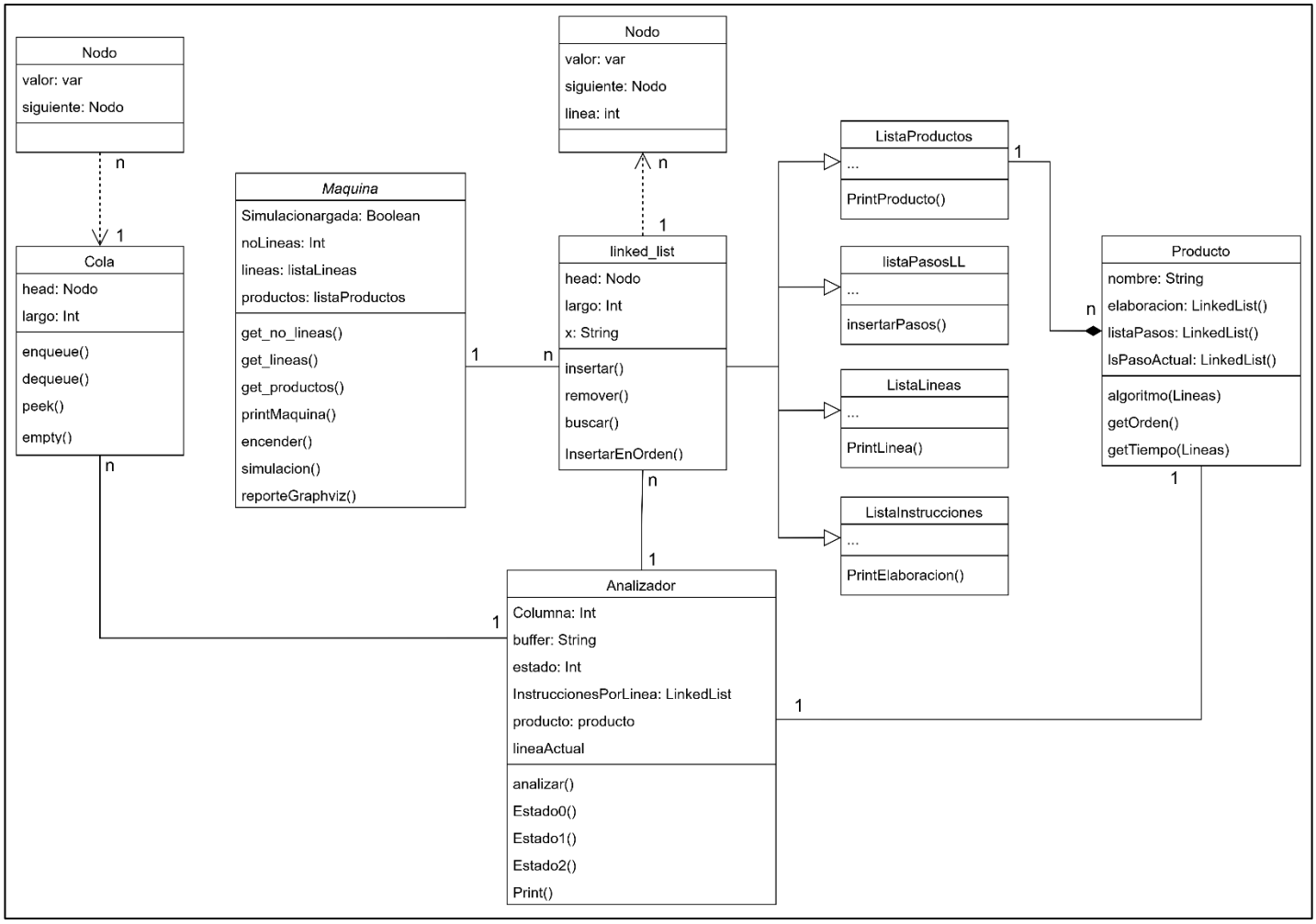

Figura 15: Diagrama de Clases
Fuente: Elaboración propia (2021)

Conclusión

El uso de colas permitió emular las líneas de producción con facilidad pues permitió almacenar las instrucciones en las que se necesitaba que se realizaran.

Al utilizar listas enlazadas se pudieron almacenar los pasos realizados en cada segundo de tal manera de que se puede acceder a ellos en el orden en el que se ingresaron, pudiendo almacenar varios valores por nodo, disminuyendo la cantidad de listas que se hubieran utilizado de otra manera.

Citas

- Miller, B y Ranum, D (2006) «Problem Solving with Algorithms and Data Structures Using Python», Decorah, Iowa, Estados Unidos de America: Luther College
- Bustos, B *(2020)* «Tipos de datos abstractos», Santiago, Chile: Universidad de Chile
- Weiss, M. A. (2007) «Data Structures and Algorithm Analysis in Java» Pearson Education.