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 2021.

PROYECTO #2 IPC2

202004810 – Henry Ronely Mendoza Aguilar

Resumen

Empleando la programación orientada a objetos se desarrolló un software que permite la optimización del tiempo de ensamblaje de una línea de producción encargada de manufacturar productos electrónicos.

Para el desarrollo de la aplicación se utilizaron Estructuras de datos abstractos para el manejo de la memoria dentro del programa los cuales fueron una lista doblemente enlazada y el concepto de lista de listas. Asimismo, para la optimización de la producción se emplearon métodos de recorrido de las listas y verificaciones. A su vez, respecto a la generación de reportes se graficó utilizando la aplicación Graphviz. Dentro de Python se utilizó la librería ElementTree para la lectura y escritura de archivos de extensión xml

Palabras clave
Optimización, Python, POO, software, programación

Abstract

Using object-oriented programming, software was developed that allows the optimization of the assembly time of a production line in charge of manufacturing electronic products.

For the development of the application, abstract data structures were used for memory management within the program, which were a doubly linked list and the concept of a list of lists. Likewise, for the optimization of production, methods of touring the lists and verifications were used. In turn, regarding the generation of reports, it was graphed using the Graphviz application. Within Python, the ElementTree library was used to read and write files with an xml extension

Keywords
Optimization, Python, OOP, software, programming

Introducción

La optimización de procesos es importante para cualquier área de la industria, en un mundo globalizado en el cual la producción de productos a gran escala ha sido muy importante para la creación de múltiples dispositivos para la vida cotidiana de las personas tanto para trabajo, comunicación o entretenimiento.

Es por ello que la empresa Digital Intelligence S. A. solicitó la elaboración de un software que le permita el ensamblaje en el menor tiempo posible de los productos que son encargados de producir.

Desarrollo

Para el desarrollo de la aplicación solicitada por Digital Intelligence inicié con la elaboración del diagrama de clases que implementaría para el desarrollo del software. (*ver figura 1*).

Dentro de las clases creadas para el almacenamiento de los datos de entrada a través de un archivo con extensión xml. Se implementó una clase llamada máquina la cual funcionaria como una lista de listas enlazadas a esta se le añadieron distintos a tributos que corresponden a las líneas de producción, el tiempo de ensamblaje de cada línea y la cantidad de componentes por la línea de producción. (*ver figura 2*) para almacenar cada línea de producción se implemento una lista enlazada para así con esta en cada nodo asignar una nueva lista y con ello tener la lista de listas para el manejo de los brazos en la línea de ensamblaje (*ver figura 3).*

Teniendo claro la forma con la que se almacenaría los datos extraídos del archivo de entrada con extensión xml. A solicitud para el desarrollo de la aplicación se debían implementar Estructuras de Datos Abstractos o TDA'S para el manejo de la memoria del software, en este caso de forma dinámica y tener un mejor rendimiento durante la ejecución de este.

La lista enlazada dentro del programa únicamente posee los atributos del dato, el cual almacena los objetos a través de una clase nodo en el apuntador first y los apuntadores next y preview. Dentro de los métodos se obtiene el método empty el cual únicamente retorna si la lista está vacía o no. Y el método append el cual sirve para añadir los objetos de terreno a la lista enlazada. (*ver figura 4*).

Dentro del archivo xml venían más datos a demás de la mencionada clase máquina, dentro del archivo de entrada se contienen los datos de los productos que la máquina es capaz de ensamblar, para ello se creo una nueva clase llamada productos (*ver figura 5*) en este objeto se designaron los atributos de nombre del producto y adicional una lista con los componentes que necesita para ser ensamblados.

Dentro de esta clase se añadió un método para visualizar de forma gráfica el proceso de ensamblado (*ver figura 6*) para ello se utilizó la herramienta Graphviz la cual a través de la escritura de un archivo en formato dot, se realizó un grafo con una secuencia del proceso de ensamblaje (*ver figura 7*).

El sistema admite un archivo secundario el cual lo nombran como simulación y puede haber tantos como el usuario lo desee, este archivo únicamente contiene un identificador de que numero de simulación es y un listado de los productos a realizar de forma masiva. Para el almacenamiento de estos datos se creo una clase llamada simulación la cual posee como atributos los datos mencionados anteriormente. (*ver figura 8*)

Proceso de ensamblado:

El proceso de ensamblado tenia varias restricciones sin embargo dichas restricciones ayudaban a la optimización del proceso. Una de ellas era que los brazos de las líneas de producción se deben mover de forma simultánea. Esto permitía que el tiempo de ensamblaje fuera el menor posible ya que los multiprocesos ayudan a la disminución y optimización de tiempos de ejecución en este caso de tiempos de ensamblaje.

Esto supuso un problema ya que dentro de Python que es el lenguaje principal en el cual se desarrolló la aplicación se trabaja de forma secuencial y la implementación de hilos o threads para la creación de los multiprocesos lleva una optimización más amplia la cual seria sobretrabajar el proyecto ya que existen diferentes formas de simular el proceso.

La solución a la que se llegó y por ultimo se programó dentro del software fue crear una serie de variables auxiliares ligadas a la clase máquina, esto significa que por cada línea de producción existen variables las cuales sirven para comparaciones, verificaciones y guardar los movimientos que realizan los brazos. Estas variables fueron Tproduccion el cual almacena el tiempo que lleva de recorrido esa línea específicamente, flotante esta variable se utilizó como verificador para ignorar movimientos repetidos dentro del proceso del brazo, y por último moves, el cual fue una sublista que contiene los movimientos que realiza el brazo de la línea de producción. (*ver figura 9).*

La lista de movimientos contenía nuevamente objetos ya que empleándolos es más sencillo la transferencia de datos entre los métodos. En este caso se creó una clase llamada Acciones la cual contiene como atributos el tiempo de ese movimiento, la descripción del movimiento que puede ser: Moverse a, Esperar o Ensamblar, y por ultimo el componente al cual se movió o ensambló. Si se realiza la acción de esperar no existe un componente asignado al objeto. (*ver figura 10*).

Teniendo en cuenta los datos y variables auxiliares a emplear para facilitar la simulación para el ensamblaje de cada producto, se crea el método de fabricación. Este consiste en una anidación de tres ciclos while. El primero recorre los componentes para la creación del producto seleccionado, el segundo recorre las líneas de producción que contiene la máquina y el tercero recorre los componentes que posee la línea de producción. Dentro de este último ciclo while se añadió un if el cual es el encargado de asignar el movimiento a efectuar a la lista auxiliar de movimientos, en este caso si ingresa directamente al if significa que ha sido encontrado por lo que se ensambla, si existen números almacenados en la variable de Tproducción se realizan diferentes comparaciones para añadir la función de esperar correspondiente según sea necesario. Si no entra a este if principal dentro del else se añade la función de moverse a, dentro de este existen dos verificaciones extra las cuales se encarga de que el movimiento se realice dentro de la misma línea de producción y así añadirlo a la lista de movimientos, y otra que se encarga de verificar si el movimiento realizado ya fue efectuado por lo que no lo añade para no tener movimientos repetidos. (*ver figura 11*)

Creación de Reportes:
Para esta aplicación se solicitaron diferentes archivos de salida que servirían para el análisis de los datos de una forma más sencilla. Los cuales son reportes en formato html, reporte en formato xml y grafos de la barra de proceso.

El grafo fue explicado anteriormente dentro de la clase de productoa, por lo que partimos del reporte en html, este formato sirve para el despliegue de información en el navegador, (Lenguaje de Marcas de Hipertexto, del inglés HyperText Markup Language) es el componente más básico de la Web. Define el significado y la estructura del contenido web. Además de HTML, generalmente se utilizan otras tecnologías para describir la apariencia/presentación de una página web (CSS) o la funcionalidad/comportamiento (JavaScript).

"Hipertexto" hace referencia a los enlaces que conectan páginas web entre sí, ya sea dentro de un único sitio web o entre sitios web. Los enlaces son un aspecto fundamental de la Web. Al subir contenido a Internet y vincularlo a las páginas creadas por otras personas, te conviertes en un participante activo en la «World Wide Web» (Red Informática Mundial). para este método se utilizaron los datos de las listas de

movimientos para plantearlos en una tabla la cual contiene el tiempo por movimiento, el movimiento de cada brazo por columna por lo que se generan tantas columnas como líneas de producción existan. (*ver imagen 12)* El archivo de html no fue el único empleado para el despliegue de la información en el navegador, también se utilizó un archivo de estilos de formato CSS el cual es (siglas en inglés de Cascading Style Sheets), en español «Hojas de estilo en cascada», es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado.2 Es muy usado para establecer el diseño visual de los documentos web, e interfaces de usuario escritas en HTML o XHTML; el lenguaje puede ser aplicado a cualquier documento XML, incluyendo XHTML, SVG, XUL, RSS, etcétera. Junto con HTML y JavaScript, CSS es una tecnología usada por muchos sitios web para crear páginas visualmente atractivas, interfaces de usuario para aplicaciones web y GUIs para muchas aplicaciones móviles (como Firefox OS).

CSS está diseñado principalmente para marcar la separación del contenido del documento y la forma de presentación de este, características tales como las capas o layouts, los colores y las fuentes.4 Esta separación busca mejorar la accesibilidad del documento, proveer más flexibilidad y control en la especificación de características presentaciones, permitir que varios documentos HTML compartan un mismo estilo usando una sola hoja de estilos separada en un archivo .css, y reducir la complejidad y la repetición de código en la estructura del documento.

Otro de los reportes generados fue el archivo de salida en formato xml, valiéndonos de la librería de elemetTree se facilió la escritura de un archivo en este formato, la información que contiene es la misma que el archivo de salida html, sin embargo, el formato que se emplea cambia la escritura y la forma de lectura de este (*ver figura 13*)

Cada uno de los archivos tienen dos variantes las cuales son de forma individual donde se presenta la información del producto seleccionado, la segunda variante es la empleada por las simulaciones la cual genera el archivo html con todos los productos dentro de la simulación escogida y de la misma forma con el archivo xml.

Interfaz gráfica:
Por último, para el desarrollo de la aplicación se realizó una interfaz gráfica para el manejo de las funciones de una forma adecuada y agradable al usuario ya que de esta forma seria más fácil el manejo de la carga de archivos y la visualización de la información.

La interfaz consiste en una ventana (*ver imagen 14*) que en la parte superior cuenta con cuatro botones, el primero nombrado configurar máquina se encarga de crear un explorador de archivos para cargar el archivo con extensión xml que se encarga de configurar la maquina con las líneas de producción necesarias, el segundo es el botón simulaciones que se encarga de cargar los datos de las simulaciones, el tercer botón es reportes xml el cual en un principio esta inactivo pero en esencia se encarga de generar el reporte xml individual de cada producto elegido, el botón de soporte muestra la información del programador de la aplicación la cual sirve en dado caso el programa dejase de funcionar o tuviera algún error.

Seguidamente se muestran dos menús desplegables, en el primero se cargan los productos individuales que pueden ser creados por la máquina, al seleccionar un producto se habilitaran tres botones, el primero de ellos será el de reporte xml, los siguientes son el de guardar grafo que guarda en formato png la barra de producción el producto generado por Graphviz, el tercer botón es el exportar html el cual se encarga de generar el archivo html del producto seleccionado.

Al seleccionar el producto del lado derecho de la ventana se llenara una tabla para la visualización de los movimientos realizados por la máquina para el

ensamblaje del producto, luego del lado izquierdo se muestra toda la información relacionada para la producción, indica el nombre del producto, el tiempo total de producción y la barra de secuencia.

El segundo menú desplegable contiene las simulaciones que han sido cargadas, al seleccionar una automáticamente se generaran los archivos xml y html mencionados anteriormente.

Anexos:

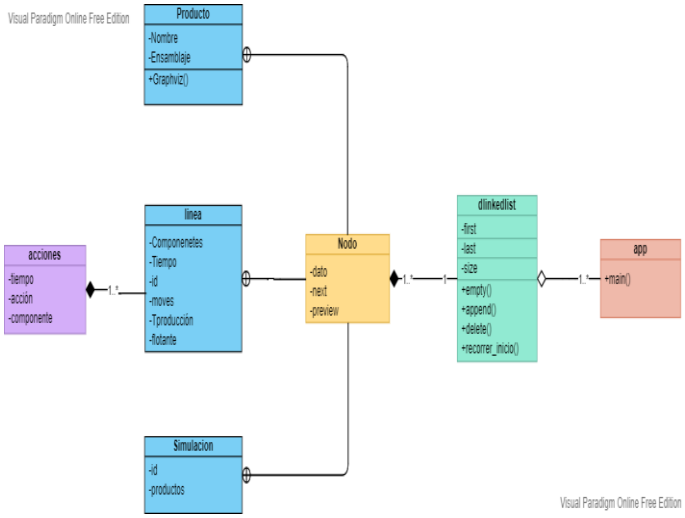

Figura 1. Diagrama de Clases
Fuente: Elaboración propia, 2021

```
class linea:
    def __init__(self,componentes,tiempo,id,acciones,Tp,flot):
        self.componentes = componentes  #lista = [C1, C2, C3....]
        self.tiempo = tiempo
        self.id = id #L1, L2, L3...
        self.moves = acciones
        self.Tproduccion=Tp
        self.flotante=flot
```

Figura 2. Clase línea
Fuente: Elaboración propia, 2021

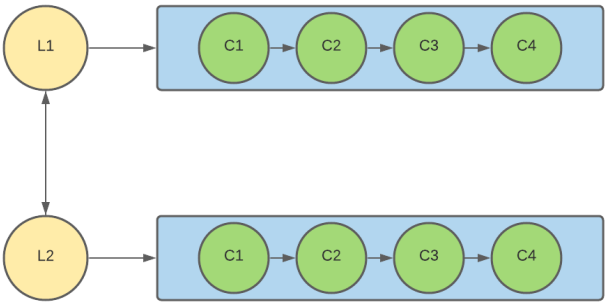

Figura 3. Representación de línea de producción
Fuente: Elaboración propia, 2021

```
lista.py > ...
1   from nodo import nodo
2   class dlinkedlist:
3       def __init__(self):
4           self.first=None
5           self.last=None
6           self.size=0
7
8   >   def empty(self): ···
10
11  >   def append(self,dato): ···
19
20  >   def delete(self): ···
30
31  >   def recorrer_inicio(self): ···
36
37
```

Figura 4. Lista enlazada
Fuente: Elaboración propia, 2021

```
class producto:
    def __init__(self,nombre,ensamblaje):
        self.nombre = nombre
        self.ensamblaje = ensamblaje #lista
```

Figura 5. Clase producto
Fuente: Elaboración propia, 2021

```
def Graphviz(self):
    file = open(self.getNombre()+'.dot','w')
    contenido='''
            digraph G {

                    bgcolor = "#9DDEFC"
                    node[shape="box" fillcolor="#E6D4BE" style =filled]
                    fontsize="20"
            '''
    contenido+='label="'+self.getNombre()+'"\n'

    aux=self.getEnsamblaje().first
    while aux:
        if aux.next!=None:
            contenido+='rank=same{'+aux.dato+'->'+aux.next.dato+'}\n'
        aux=aux.next

    contenido+='}'
    file.write(contenido)
    file.close()
    system('cmd /c "dot.exe -Tpng ' + self.getNombre()+'.dot'+' -o '+self.getNombre()+'.png' + '"')
```

Figura 6. Creación de grafo empelando Graphviz.
Fuente: Elaboración propia, 2021

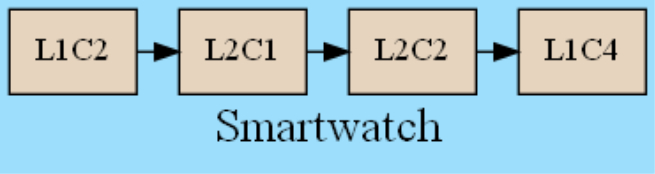

Figura 7. Ejemplo grafo de secuencia
Fuente: Elaboración propia, 2021

```
class simulacion:
    def __init__(self,id,productos):
        self.id = id
        self.productos = productos

    def getId(self):
        return self.id

    def getProductos(self):
        return self.productos
```

Figura 8. Clase simulación
Fuente: Elaboración propia, 2021

```
self.moves = acciones
self.Tproduccion=Tp
self.flotante=flot
```

Figura 9. Variables temporales
Fuente: Elaboración propia, 2021

```
class acciones:
    def __init__(self,tiempo,accion,componente):
        self.tiempo=tiempo
        self.accion=accion
        self.componente=componente
```

Figura 10. Clase acciones
Fuente: Elaboración propia

```
    def fabricar(self,nombre,maquina,productos):
        ensamble=''

        aux = productos.first
        while aux:
            if aux.dato.getNombre()==nombre:
                ensamble=aux.dato.getEnsamblaje()
                aux.dato.Graphviz()
            aux=aux.next

        print('Ensamblar: '+nombre)
        print('Con componentes: ')
        ensamble.recorrer_inicio()

        L = maquina.first
        com=ensamble.first

        esp=0
        pos=1
        while com:
            while L:
                C=L.dato.getComponentes().first
                temp=L.dato.getMoves()
```

Figura 11. Función fabricar
Fuente: Elaboración propia, 2021

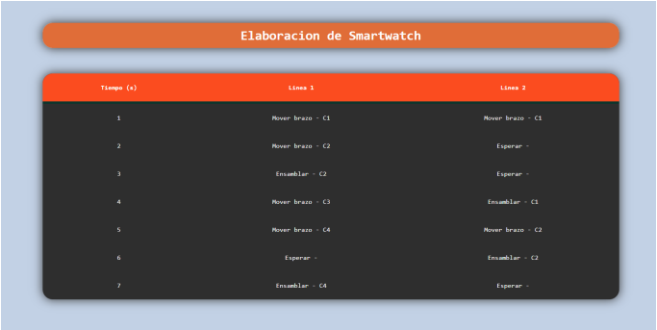

Figura 12. Reporte HTML
Fuente: Elaboración propia, 2021

```
<SalidaSimulacion>
  <ListadoProductos>
    <Producto>
      <Nombre>Smartwatch</Nombre>
      <TiempoTotal>7</TiempoTotal>
      <ElaboracionOptima>
        <Tiempo NoSegundos="1">
          <LineaEnsamblaje Nolinea="1">Mover brazo - C1</LineaEnsamblaje>
          <LineaEnsamblaje Nolinea="2">Mover brazo - C1</LineaEnsamblaje>
        </Tiempo>
        <Tiempo NoSegundos="2">
          <LineaEnsamblaje Nolinea="1">Mover brazo - C2</LineaEnsamblaje>
          <LineaEnsamblaje Nolinea="2">Esperar - </LineaEnsamblaje>
        </Tiempo>
        <Tiempo NoSegundos="3">
          <LineaEnsamblaje Nolinea="1">Ensamblar - C2</LineaEnsamblaje>
          <LineaEnsamblaje Nolinea="2">Esperar - </LineaEnsamblaje>
        </Tiempo>
        <Tiempo NoSegundos="4">
          <LineaEnsamblaje Nolinea="1">Mover brazo - C3</LineaEnsamblaje>
          <LineaEnsamblaje Nolinea="2">Ensamblar - C1</LineaEnsamblaje>
        </Tiempo>
        <Tiempo NoSegundos="5">
          <LineaEnsamblaje Nolinea="1">Mover brazo - C4</LineaEnsamblaje>
          <LineaEnsamblaje Nolinea="2">Mover brazo - C2</LineaEnsamblaje>
        </Tiempo>
        <Tiempo NoSegundos="6">
          <LineaEnsamblaje Nolinea="1">Esperar - </LineaEnsamblaje>
          <LineaEnsamblaje Nolinea="2">Ensamblar - C2</LineaEnsamblaje>
        </Tiempo>
        <Tiempo NoSegundos="7">
          <LineaEnsamblaje Nolinea="1">Ensamblar - C4</LineaEnsamblaje>
          <LineaEnsamblaje Nolinea="2">Esperar - </LineaEnsamblaje>
        </Tiempo>
      </ElaboracionOptima>
    </Producto>
  </ListadoProductos>
</SalidaSimulacion>
```

Figura 13. Reporte XML
Fuente: Elaboración propia, 2021

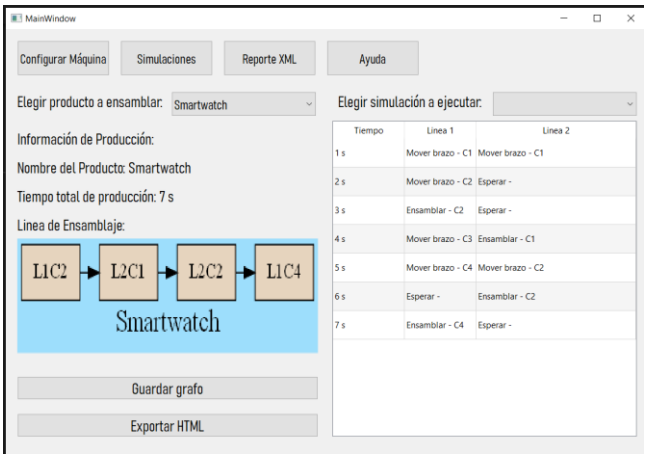

Figura 14. Interfaz gráfica
Fuente: Elaboración propia. 2021