DESDE 2013
https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI
ISSN: 2007-6363

Publicación Semestral Pädi Vol. 10 No. Especial 2 (2022) 100-108

# Creación de Redes de Petri a partir de la descripción de procesos de negocios utilizando Python
# Creation of Petri Nets from the business processes description using Python

S. Medina-García [a]. J. Medina-Marín [a] O. Montaño-Arango [a] M. González-Hernández [a]

[a] *Área Académica de Ingeniería y Arquitectura, Universidad Autónoma del Estado de Hidalgo, 42184, Pachuca, Hidalgo, México.*

**Resumen**

Una red de Petri (PN) se utiliza con fines de análisis de procesos de negocios (BP ) para posteriormente perfeccionarla y sirva como base para la implantación del proceso real. Es claro que para lograrlo es necesario establecer las actividades, requerimientos de entrada y las salidas de los BP, los cuales están definidos en su descripción de este. Las PN tienen como ventaja a diferencia de otras herramientas de modelado que permiten la representación de eventos combinando la visualización grafica con una notación matemática subyacente. Una característica requerida por las redes de Petri es la función para exportar PN a otras herramientas y para importarlas de otras herramientas. El autor ha creado un algoritmo que convierte el texto de la descripción de un proceso de negocio en una red de Petri (importar), este ha sido programado en Python. En el presente artículo se describe el funcionamiento del mencionado algoritmo.

*Palabras Claves: Procesos de Negocio, Redes de Petri, Python, algoritmo*

**Abstract**

A Petri net (PN) is used for business process analysis (BP) purposes to later refine it and serve as the basis for the implementation of the real process. It is clear that to achieve this it is necessary to establish the activities, input requirements and outputs of the BP, which are defined in its description. Unlike other modeling tools, PNs have the advantage that they allow the representation of events by combining graphical visualization with an underlying mathematical notation. A feature required by Petri nets is the ability to export PNs to other tools and to import them from other tools. The author has created an algorithm that converts the text of a business process description into a Petri net (import), it has been programmed in Python. This article describes the operation of the aforementioned algorithm.

*Keywords: Business Processes. Petri nets, Python, algorithm*

## Introducción

El modelado de los procesos de negocio (Business Process Modeling) permite representar los aspectos relevantes de un proceso con ciertos objetivos, como puede ser, analizar, automatizar o simular el comportamiento de este. Obviamente la complejidad puede extenderse ya que un proceso puede estar compuesto de varios subprocesos (Van Hee, 2013)

En la mayoría de los casos, los procesos de negocios representan sistemas complejos ya que intervienen varias entidades para su desarrollo. Si hablamos del proceso de facturación de un automóvil, probablemente intervengan 6 o 7 actividades (registro de datos del cliente y vehículo, verificación de buró de crédito, realización de pago inicial, convenio para pago de parcialidades restantes, facturar, timbrado de factura, envió de factura), pero en el proceso de fabricación de dicho automóvil estaríamos hablando de muchas más (diseño del automóvil, creación de prototipos, estudios de mercado, adquisición y almacenamiento de materia prima, fabricación de partes para carrocería, fabricación de partes para motor, fabricación de sistema eléctrico, fabricación de interiores, fabricación de vidrios, creación de los sistemas de seguridad, ensamblado de todos los componentes).

*Autor para la correspondencia: me429457@uaeh.edu.mx
**Correo electrónico**: me429457@uaeh.edu.mx (Samuel Medina García), jmedina@uaeh.edu.mx (Joselito Medina Marín), omontano@uaeh.edu.mx (Oscar Montaño Arango), mghdez@uaeh.edu.mx (Manuel González Hernández)

**Fecha de recepción: 07/02/2022 Fecha de aceptación: 27/04/2022 Fecha de publicación: 24/06/2022**
**https://doi.org/10.29057/icbi.v10iEspecial2.8673**

Existe una gran cantidad de técnicas de modelado de procesos de negocio dirigidas a representar los diferentes elementos que los componen con el propósito de comprenderlos y analizarlos desde un punto de vista operacional (Vergidis, 2007) y solo algunas técnicas de modelado pueden enfocarse hacia el análisis cuantitativo, por ejemplo, las redes de Petri coloreadas, programación lineal, utilidad y teoría de juegos, etc., y de esa manera estar en condiciones de proponer una mejora u optimización

Una condición necesaria para que los diseñadores de procesos de negocios puedan establecer los contextos reales y parámetros necesarios para el diseño de sistemas es que el análisis esté completo tanto en entidades como sus respectivas propiedades, y una vez que dichas condiciones están dadas es posible organizar todas las operaciones de una manera lógica y ordenada para posteriormente modelar el mencionado proceso (Zamora, 2016)

En tal contexto, se deben replicar los resultados que el proceso de negocio esté arrojando, en otras palabras, se debe establecer el algoritmo de modelación de procesos de negocio.

Las redes de Petri son un ejemplo de una técnica de modelado de procesos de negocios que combina representación visual usando notación estándar con una representación matemática subyacente (Thal, 2018). Las redes de Petri cuentan con un lenguaje gráfico apropiado para modelar sistemas concurrentes, consta de dos tipos de nodos, llamados lugares y transiciones. Las redes de Petri han sido modificadas y extendidas por varios investigadores para permitir capacidades de modelado más potentes. Algunas de sus variaciones incluyen redes de Petri cronometradas, redes de Petri estocásticas, redes de Petri de colores y redes de Petri jerárquicas (Thal, 2018)

En la sección inicial del presente artículo el lector encontrará el marco conceptual del modelado de procesos de negocios, estableciendo los beneficios de la modelación con redes de Petri. En la segunda sección se establece el estado del arte de y se describen las formas básicas de las Redes de Petri. En la tercera sección se muestra un caso de estudio de un organismo operador donde se traducen los flujos de trabajo a redes de Petri, para finalizar con la exposición de resultados y conclusiones. En la segunda, se describen los flujos de trabajo de un organismo operador de agua utilizando estructuras elementales de redes de Petri. La tercera parte, presenta las conclusiones y por qué la modelación de procesos de negocios con redes de Petri ayudará a las organizaciones a ser más competitivas

Los resultados de esta investigación se muestran en un caso de estudio relacionado con un proceso de negocio de un organismo operador de agua (toma de lectura del aparato medidor de consumo y entrega de la boleta de cobro).

## 1. Marco Conceptual

### *1.1. Proceso de negocio*

Proceso de negocio; es una secuencia de tareas estructuradas o semi-estrucutradas relacionadas entre sí que se ejecutan en serie o paralelo para llevar a cabo todas las operaciones de la empresa, ya sea las que están relacionadas para agregar valor al producto o servicio (actividades primarias) o bien las actividades de apoyo. En este conjunto de tareas suelen intervenir diferentes áreas y departamentos e involucrar perfiles de profesionales distintos que hacen un trabajo colaborativo. Una de las características de los procesos de negocios es que tienen un inicio, pero no necesariamente un fin, esta característica distingue a un proceso de un proyecto (Hamilton, 2013)

### *1.2. Modelo de proceso de negocio*

El modelo de proceso describe cualquier representación abstracta de un proceso. Un modelo de proceso puede ser tan simple como un diagrama de cajas y flechas o tan complejo como el software de computadora que permite la simulación del proceso (Lee, 2015)

### *1.3. Mapeo de Procesos*

El mapeo del Proceso desarrolla una representación del proceso del "como es", con el objetivo de exponer los puntos débiles que deben ser abordados. Una vez que una empresa desarrolla un mapa de procesos, se puede realizar un análisis de carencias, que es; una evaluación de las disparidades entre la situación actual de la organización y sus objetivos a largo plazo (Lee, 2015)

### *1.4. Reingeniería de Procesos de Negocios (Business Process Reingineering BPR)*

La reingeniería es un enfoque que analiza y modifica los procesos básicos de trabajo en el negocio. Se refiere a repensar de manera fundamental los procesos de negocios y rediseñarlos radicalmente, con el fin de obtener mejores logros en el desempeño.

Para la gente de negocios, permite aplicar los conocimientos de sus empresas, con el propósito de hacerlos más efectivos: más rápidos mayor cantidad, mayor calidad, menores costos, mayores ganancias.

BPR se refiere a las actividades que desarrollan las empresas para optimizar y adaptar sus procesos. El concepto se refiere a proveer un mecanismo para integrar procesos, personas e información. Se requiere una infraestructura capaz de separar flujos, reglas de negocios y servicios. Los factores clave del concepto son: la orientación hacia los procesos, el cambio radical y la gran magnitud de los resultados esperados, además de implica una integración en tiempo real entre los procesos de la empresa y los procesos de sus proveedores, socios de negocios y clientes (Vergidis, 2007)

### *1.5. Rediseño de Procesos*

El diseño de procesos es una etapa de la gestión de procesos que se produce después del análisis del proceso

actual, es decir, cómo está ahora, teniendo como objetivo crear nuevos procesos o mejorar los existentes. El diseño del proceso está orientado hacia el futuro y es imposible llevar a cabo esta actividad de manera correcta sin haber hecho el análisis previo (Gross, 2021)

*1.6. Ejecución del proceso*

La manera común de ejecutar un proceso automáticamente es utilizando una aplicación que realice todas las funciones del proceso.

La ejecución completa del proceso se logra combinando y conectando distintas aplicaciones más una intervención humana significativa. Debido a estas razones de complejidad, cambiar un proceso es costoso como también es difícil tener una visión general del estado del proceso (Yao, 2008).

## 2. Estado del Arte

Las Redes de Petri es una herramienta que permite modelar el comportamiento de procesos. En este contexto es importante mencionar que un modelo es una representación de la realidad y obviamente una representación no implica su construcción real con los inconvenientes tanto técnicos como financieros. En los procesos de negocio, la intervención de dos o más áreas de la organización de manera paralela es frecuente, en el ejemplo de la reinscripción de alumnos en una universidad, la consulta de adeudos a tesorería, biblioteca, almacén de deportes puede hacerse de manera paralela a pesar de estar hablando de tres áreas diferentes. Las Redes de Petri fueron diseñadas específicamente para modelar este tipo de sistemas (Vejrazkova, 2013).

Durante el proceso del ciclo de vida del desarrollo del sistema, el equipo de desarrollo toma decisiones difíciles con respecto a dónde asignar los recursos del proyecto. A medida que los sistemas se vuelven más complejos y los presupuestos más restringidos, entender cómo tomar estas decisiones se vuelve más problemático. Los gerentes de proyecto y los ingenieros de sistemas pueden usar las Redes de Petri para representar con precisión su sistema y calcular de manera más precisa la madurez de la tecnología a lo largo del ciclo de vida del desarrollo del sistema. Este modelo puede ayudar en el proceso de decisión de asignación de recursos del sistema y permite una mejor comprensión de cómo funcionan los componentes clave del sistema. (Tian, 2018)

Es importante recordar que las redes de Petri son una representación gráfica de un sistema está formada por lugares, transiciones, arcos dirigidos y tokens las condiciones son: los arcos conectan un lugar a una transición, así como una transición a un lugar. No puede haber arcos entre lugares ni entre transiciones.

Las estructuras básicas de las redes de Petri (Fig 1) permiten modelar los procesos de negocios y ensamblando dichas estructuras entre sí podrán modelarse procesos más complejos. Según sea el flujo de actividades en dicho proceso y dependiendo de las condiciones establecidas, se llevan a cabo actividades en forma secuencial o paralela (Sivaraman, 2002).

Existen herramientas de software que utilizan interfaces gráficas para crear redes de Petri, una de ellas es WoPed (Workflow Petri Net Designer). Desarrollada en Universidad Estatal Cooperativa de Karlsruhe en Alemania, es una aplicación de código abierto cuyo objetivo principal es proporcionar un software fácil de usar para modelar, simular y analizar procesos utilizando estructuras de redes de Petri. Cuenta con diferentes funcionalidades, entre las que se cuenta un conversor de proceso a texto y de texto a proceso (https://sourceforge.net/projects/woped/files/woped/)

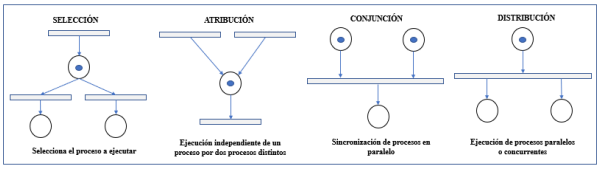



*Fig 1. Estructuras básicas de las redes de Petri.*

Según agua.org.mx los organismos operadores de agua, se encargan de operar, conservar y administrar los sistemas de agua potable, alcantarillado y saneamiento, con el objeto de dotar estos servicios a los habitantes de un municipio o de una entidad federativa.

La descripción de uno de los procesos de negocio de un organismo operador es la visita del "lecturista" al domicilio del usuario a efecto de registrar la lectura marcada por aparato medidor de consumo y entregar la boleta de pago del servicio (recibo de pago). La descripción de tal proceso en lenguaje natural pudiera quedar de la siguiente manera:

*"El lecturista llega al domicilio del usurario registra la lectura del medidor y al mismo tiempo entrega el recibo de pago"*

Se puede observar que la estructura de red de Petri que se aplica en este caso es la de Distribución; dos actividades (registra y entrega) realizadas en forma paralela.

**Algoritmo**

Premisa: Se establece que todas las organizaciones cuentan con su propio argot. En un organismo operador es común escuchar los términos: lecturista, medidor, recibo de pago, usuario, etc. (queda fuera del alcance del presente artículo la explicación de los términos citados).

El algoritmo creado para esta instancia cuenta con una base de datos que a su vez está conformada por tres diccionarios o tablas (plazas, transiciones, conjunciones), una tabla con la descripción del proceso de negocios y una quinta tabla con el proceso de negocios desglosado por elemento de la red de Petri.

Se busca la coincidencia de todos los elementos de los tres diccionarios en el texto descriptivo del proceso de negocios, cuando se encuentra tal coincidencia se registran las plazas, las transiciones y conjunciones encontradas junto con la posición dentro del texto (Fig. 2). El registro obtenido se

ordena de acuerdo con la posición de los elementos permitiendo dibujar la red de Petri

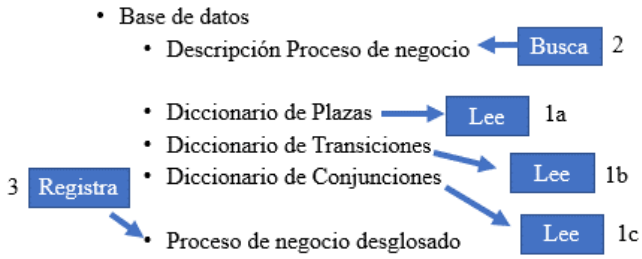



*Fig 2. Ciclo del algoritmo*

**Consideraciones iniciales. Entradas**

La organización debe proporcionar:

a) La descripción de las plazas y transiciones que integran sus diferentes procesos de negocios siendo estos muy particulares para cada organización
b) La redacción del proceso de negocio en lenguaje natural

Se dispone una estructura de base de datos de la siguiente manera:

| Campo | Tipo | Descripción |
|---|---|---|
| id_plaza | Numérico | Consecutivo de registro |
| des_plaza | Texto | Descripción de la plaza |

*Tabla 1. Catálogo de plazas (cat_plazas)*

| Campo | Tipo | Descripción |
|---|---|---|
| id_transicion | Numérico | Consecutivo de registro |
| des_transicion | Texto | Descripción de la transición |

*Tabla 2. Catálogo de conjunciones (cat_transiciones)*

| Campo | Tipo | Descripción |
|---|---|---|
| id_pn | Numérico | Consecutivo de registro |
| des_pn | Texto largo | Descripción del proceso de negocio en lenguaje natural |

*Tabla 3. Catálogo de procesos de negocio (cat_pn)*

Cualquier proceso de negocio de cualquier organización es redactado utilizando conjunciones para dar comprensión a la lectura, dichas conjunciones son: “y”, “o”, “al mismo tiempo”, “en forma paralela”, etc. Se dispone como elemento de entrada adicional la siguiente tabla:

| Campo | Tipo | Descripción |
|---|---|---|
| id_conjuncion | Numérico | Consecutivo de registro |
| des_conjuncion | Texto | Descripción de la conjunción. |

*Tabla 3. Catálogo de conjunciones (cat_conj)*

**Consideraciones iniciales. Salidas**

La salida del proceso consiste en el registro de plazas, transiciones y conjunciones en el texto proporcionada del proceso de negocios.
Se dispone de la siguiente tabla como parte de la base de datos.

| Campo | Tipo | Descripción |
|---|---|---|
| id_renglon | Numérico | Consecutivo de registro |
| tipo_simbolo | Texto | Descripción del tipo de símbolo: círculo, rectángulo o línea |
| Desc_ptc | Texto | Descripción de: la plaza, transición o conjunción |
| Posición_en _texto | Numérico | Posición donde se encuentra localizada, la plaza, la transición o la conjunción en la redacción del Proceso de Negocio |

*Tabla 3. Catálogo de conjunciones (dat_PetNet)*

**Funcionamiento**

1. Registrar el proceso de negocio en DAT_PN

El algoritmo se basa en identificar las plazas, las transiciones y las conjunciones que se encuentran redactadas en el proceso de negocio. De tal manera que tendremos como información de entrada:

a. Catálogo de plazas – CAT_PLAZAS (id_plaza, descrip_plaza)
b. Catálogo de transiciones – CAT_TRANS (id_trans, descrip_transicion))
c. Catálogo de conjunciones – CAT_CONJ (id_conj, descrip_conj)
d. Registro del proceso de negocio – DAT_PN (id_proceso, texto_proceso)
e. Registro de plazas, transacciones, conjunciones y posiciones que se encuentran en el texto de Procesos de negocios – DAT_PetNet (tipo_elemento, desc_elemento, posicion)

Los dos primeros catálogos deberán ser poblados con información del dueño del proceso.

2. Redactar el proceso de negocio en lenguaje natural utilizando las plazas y transiciones registradas en CAT_PLAZ y CAT_TRANS
3. Leer CAT_PLAZAS y buscar que registros encuentran en DAT_PN. Registrar el nombre de la plaza junto con la ubicación en el texto en DAT_PetNet
4. Leer CAT_TRANS y buscar que registros se encuentran en DAT_PN. Registrar el nombre de la

plaza junto con la ubicación en el texto en DAT_PetNet

5. Leer CAT_CONJ y buscar que registros se encuentran en DAT_PN. Registrar el nombre de la plaza junto con la ubicación en el texto en DAT_PetNet
6. DAT_PetNet, contiene por separado todas las plazas, todas las transiciones, todas las conjunciones y la posición contenidas en la redacción del Proceso de negocios.
7. Seleccionando los registros de DAT_PetNet en orden ascendente por el campo posición se tiene en orden la "aparición" de los elementos que juegan en la creación de la red de Petri, a esta entidad se le asignará el nombre de DAT_PetNet_Asc
8. El tipo de elemento "conjunción" da la pauta para selección que tipo de "estructura básica de red de Petri" se va a dibujar, encontraremos "Selección, Conjunción" cuando se lea "y, o", por otra parte "Distribución, Atribución", cuando se lea, "en forma paralela, al mismo tiempo".
9. Considerando el campo posición se dibujará en un plano cartesiano círculos, rectángulos y líneas, relacionadas entre sí para representar de manera gráfica la Red de Petri.

## 3. Herramientas de software utilizadas

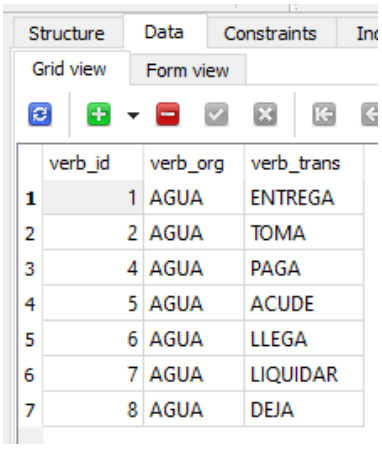
- Python. Lenguaje de programación interpretado cuya principal filosofía es que sea legible por cualquier persona con conocimientos básicos de programación.
- Thonny. Entorno de desarrollo integrado (IDE) para Python
- SQLite. Sistema de gestión de bases de datos relacional es un proyecto de dominio público
- Pandas. Biblioteca de software para manipulación y análisis de datos para el lenguaje de programación Python.
- Pygame. Conjunto de módulos del lenguaje Python que permiten la creación de videojuegos en dos dimensiones.

## 4. Aplicación en el Caso de estudio

Se muestra el contenido de las tablas de la base de datos.

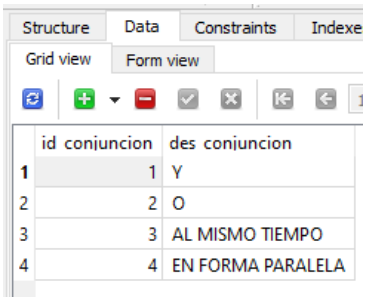

| | id_plaza | org_plaza | des_plaza |
|---|---|---|---|
| 1 | 1 | AGUA | CAJA |
| 2 | 2 | AGUA | POZO |
| 3 | 3 | AGUA | ALMACEN |
| 4 | 4 | AGUA | TESORERIA |
| 5 | 5 | AGUA | ATENCION A USUARIOS |
| 6 | 6 | AGUA | CONTRALORIA |
| 7 | 7 | AGUA | DIRECCION GENERAL |
| 8 | 8 | AGUA | DOMICILIO DEL USUARIO |
| 9 | 9 | AGUA | LECTURA DEL MEDIDOR |
| 10 | 10 | AGUA | RECIBO DE PAGO |
| 11 | 11 | AGUA | CAJERO AUTOMATICO |

*Fig 3. Diccionario de plazas de Organismo Operador*




| | verb_id | verb_org | verb_trans |
|---|---|---|---|
| 1 | 1 | AGUA | ENTREGA |
| 2 | 2 | AGUA | TOMA |
| 3 | 4 | AGUA | PAGA |
| 4 | 5 | AGUA | ACUDE |
| 5 | 6 | AGUA | LLEGA |
| 6 | 7 | AGUA | LIQUIDAR |
| 7 | 8 | AGUA | DEJA |

*Fig 4. Diccionario de transiciones de Organismo Operador*




| | id conjuncion | des conjuncion |
|---|---|---|
| 1 | 1 | Y |
| 2 | 2 | O |
| 3 | 3 | AL MISMO TIEMPO |
| 4 | 4 | EN FORMA PARALELA |

*Fig 5. Diccionario de conjunciones de Organismo Operador*

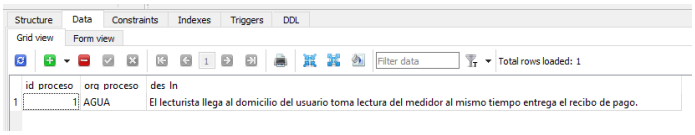
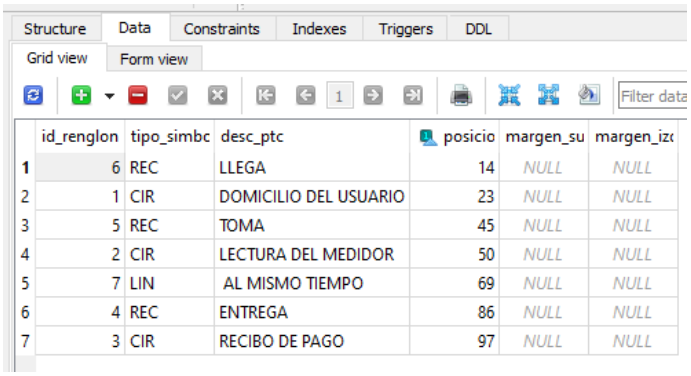



*Fig 6. Proceso de negocio de Organismo Operador*

Al ejecutar el programa de prueba se desglosa la descripción del proceso de negocios en los diferentes elementos de la red de Petri y al mismo tiempo crea el modelo gráfico. (Fig. 7 & Fig. 8)




| | id_renglon | tipo_simbc | desc_ptc | posicio | margen_su | margen_izc |
|---|---|---|---|---|---|---|
| 1 | 6 | REC | LLEGA | 14 | NULL | NULL |
| 2 | 1 | CIR | DOMICILIO DEL USUARIO | 23 | NULL | NULL |
| 3 | 5 | REC | TOMA | 45 | NULL | NULL |
| 4 | 2 | CIR | LECTURA DEL MEDIDOR | 50 | NULL | NULL |
| 5 | 7 | LIN | AL MISMO TIEMPO | 69 | NULL | NULL |
| 6 | 4 | REC | ENTREGA | 86 | NULL | NULL |
| 7 | 3 | CIR | RECIBO DE PAGO | 97 | NULL | NULL |

*Fig 7. Proceso de negocio desglosado por elemento de red de Petri de Organismo Operador*

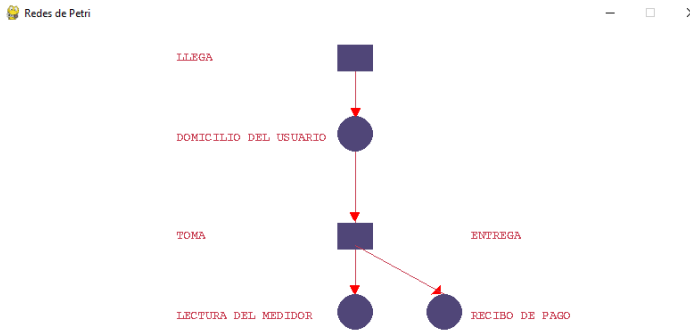



*Fig 8. Red de Petri de proceso de negocio de Organismo Operador*

El código fuente programa en Python junto con el script de la base de datos se encuentran en los apéndices A, B y C

## 5. Conclusiones.

En la presente investigación se ha descrito el desarrollo y funcionamiento de un algoritmo que traduce la descripción de un proceso de negocio en prosa a una red de Petri; las principales características de dicho algoritmo son: convierte un texto escrito en lenguaje natural a una estructura de base de datos, está pensado para que funcione en idioma español, sin embargo, podrían realizarse algunas adaptaciones para cualquier otro idioma. En el estado de arte se menciona un software open source, denominado WoPed (Workflow Petri net desgined) que cuenta con funciones de conversión de proceso (grafica) a texto y de texto a proceso, lo cual es una aportación interesante, sin embargo no queda registrado como elemento del capital intelectual (CI) de la organización, la ventaja del algoritmo del presente paper es que posibilita contar con una base de datos que contenga todos los procesos de negocio de la organización y al momento que se realice el registro de alguno se incrementa el CI.

Como se observó en la aplicación es posible crear el gráfico del proceso a partir de una descripción del proceso. Entonces, será posible modificar el gráfico del proceso modificando la redacción y esto a su vez dará la pauta para analizar y en su caso mejorar dicho conjunto de actividades. Adicionalmente es posible establecer análisis cuantitativos y estadísticos de los procesos de negocios (métricas) dando valores a cada una de las actividades (investigaciones futuras) que lo conforman y por tal medir desempeños para establecer medidas de mejora en caso de ser necesario.

En base al caso de estudio que se está plasmando, es posible observar el alcance que tienen las redes de Petri dado su rigurosa metodología para formar estructuras (plaza – transición – plaza - transición).

## 6. Referencias

Gross, S., Stelzl, K., Grisold, T., Mendling, J., Röglinger, M., & vom Brocke, J. (2021). The Business Process Design Space for exploring process redesign alternatives. Business Process Management Journal.

Hamilton Scott. Maximizing your ERP system: a practical guide for managers. 2013 McGraw Hill. Cap. 3. Justificaction of ERP investments.

Lee, W. T., Ma, S. P., Lee, S. J., Law, P. J., & Chang, T. S. (2015, October). Extending the Multiple Domain Matrix by Using Business Process Model and Notation for Business Process Analysis. In 2015 IEEE 12th International Conference on e-Business Engineering (pp. 311-318). IEEE.

Sivaraman, E., & Kamath, M. (2002). On the use of Petri nets for business process modeling. In IIE Annual Conference. Proceedings (p. 1). Institute of Industrial and Systems Engineers (IISE)

Thal Brent, Olson Bill, Blessner Paul. A robust system maturity model for complex systems utilizing system readiness level and Petri nets. Diciembre 2018. Springer Link

Tian, Y, Du, Y, Li, M, Han, D, Hu, Q. Reduced alignment based on Petri nets. Concurrency Computat Pract Exper. 2018; 30:e4411. https://doi.org/10.1002/cpe.4411

Van der Aalst, W. M. (2015). Business process management as the “Killer App” for Petri nets. Software & Systems Modeling, 14(2), 685-691.

Van Hee, K. M., Sidorova, N., & van der Werf, J. M. (2013). Business process modeling using petri nets. In Transactions on Petri Nets and Other Models of Concurrency VII (pp. 116-161). Springer, Berlin, Heidelberg

Vejrazkova Z., Meshkat A. (2013) Translating DEMO Models into Petri Net. In: Barjis J., Gupta A., Meshkat A. (eds) Enterprise and Organizational Modeling and Simulation. EOMAS 2013. Lecture Notes in Business Information Processing, vol 153. Springer, Berlin, Heidelberg

Vergidis, K., Tiwari, A., & Majeed, B. (2007). Business process analysis and optimization: Beyond reengineering. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38(1), 69-82.

Yao, Q., Zhang, J., & Wang, H. (2008, August). Business process-oriented software architecture for supporting business process change. In 2008 International Symposium on Electronic Commerce and Security (pp. 690-694). IEEE.

Zamora Elvira, Value Chain Analysis: A Brief Review (2016). Asian Journal of Innovation an Policy. 5.002:116-128

## Apéndice A. Código fuente en Python, programa 1.

Este programa analiza el proceso de negocio y lo desglosa en componentes de redes de Petri.

```
# Creación de Redes de Petri a partir de la descripción de procesos de
negocios utilizando Python
# Primera parte

import sqlite3
import pandas as pd

con = sqlite3.connect('BaseDatos.db')
cur = con.cursor()
plaza=""
empresa=input("Indique el nombre de la empresa: ")
#Proceso en lenguaje natural
df1 = pd.read_sql_query("SELECT org_proceso, des_ln FROM
dat_proceso_desln where org_proceso='%s'" % empresa, con)
proceso=df1.iloc[0,1]
proceso_may=proceso.upper()

#Plazas
df = pd.read_sql_query("SELECT org_plaza, des_plaza FROM cat_plazas
where org_plaza='%s'" % empresa, con)
reg_plazas=df.org_plaza.count()
x=0
frec_plaza=0
lista = []
pos_inicial = -1
long_cadena = len(proceso_may)
pos_car=0
renglon=1
pos_elem=0
for x in range (0,reg_plazas):
   plaza=df.iloc[x,1]
   posicion = proceso_may.find(plaza)

   if posicion != -1:
      print (' ')
   try:
      for pos_car in range (0,long_cadena-1):
         pos_inicial = proceso_may.index(plaza, pos_inicial+1)
         lista.append(pos_inicial)
         pos_car=pos_car+2
   except:
      if len(lista) != 0:
         for n in lista:
            sam=n
            entities = (renglon, "CIR", plaza,sam)
            cur.execute('INSERT INTO dat_proceso (id_renglon,
tipo_simbolo, desc_ptc, posicion_en_texto) VALUES(?,?,?,?)', entities)
            con.commit()
            renglon=renglon+1

      lista.clear()

   pos_inicial = -1
   frec_plaza = frec_plaza + proceso_may.count(plaza)
   x=x+1
print()
print("No. de plazas en el texto: ",frec_plaza)
print()

#Transciones
df1 = pd.read_sql_query("SELECT verb_org, verb_trans FROM VERBOS
where verb_org='%s'" % empresa, con)
reg_verbos=df1.verb_org.count()
x=0
frec_verbo=0
lista = []
pos_inicial = -1
pos_car=0
pos_elem=0

for x in range (0,reg_verbos):
   verbo=df1.iloc[x,1]
   posicion = proceso_may.find(verbo)

   if posicion != -1:
      print (' ')
   try:
      for pos_car in range (0,long_cadena-1):
         pos_inicial = proceso_may.index(verbo, pos_inicial+1)
         lista.append(pos_inicial)
         pos_car=pos_car+1
   except:
      print(" ")
      if len(lista) != 0:


         for n in lista:
            sam=n
            entities = (renglon, "REC", verbo,sam)
            cur.execute('INSERT INTO dat_proceso (id_renglon,
tipo_simbolo, desc_ptc, posicion_en_texto) VALUES(?,?,?,?)', entities)
            con.commit()
            renglon=renglon+1

      lista.clear()

   pos_inicial = -1
   frec_plaza = frec_plaza + proceso_may.count(plaza)
   x=x+1
print()

reg_verbos=df1.verb_org.count()
y=0
frec_verbo=0
for y in range (0,reg_verbos,1):
   verbo=df1.iloc[y,1]
   frec_verbo = frec_verbo + proceso_may.count(verbo)

print("No. de verbos en el texto: ", frec_verbo)

#Conjunciones
df2 = pd.read_sql_query("SELECT des_conjuncion FROM
cat_conjunciones", con)
#reg_conjunciones=df2.org_conjuncion.count()
reg_conjunciones=df2.des_conjuncion.count()
x=0
frec_conjuncion=0
lista = []
pos_inicial = -1
pos_car=0
pos_elem=0

for x in range (0,reg_conjunciones):
   #conjuncion=df2.iloc[x,1]
   conjuncion=df2.iloc[x,0]
   posicion = proceso_may.find(conjuncion)

   if posicion != -1:
      print (' ')
   try:
      for pos_car in range (0,long_cadena-1):
         pos_inicial = proceso_may.index(conjuncion, pos_inicial+1)
         lista.append(pos_inicial)
         pos_car=pos_car+1
   except:
      print(" ")
      if len(lista) != 0:


         for n in lista:
            sam=n
            entities = (renglon, "LIN", conjuncion,sam)
            cur.execute('INSERT INTO dat_proceso (id_renglon,
tipo_simbolo, desc_ptc, posicion_en_texto) VALUES(?,?,?,?)', entities)
            con.commit()
            renglon=renglon+1


      lista.clear()
```

```
    pos_inicial = -1
    frec_conjuncion = frec_conjuncion + proceso_may.count(conjuncion)
    x=x+1
print()


reg_conjuncion=df2.des_conjuncion.count()
y=0
frec_conjuncion=0
for y in range (0,reg_conjunciones,1):
    conjuncion=df2.iloc[y,0]
    frec_conjuncion = frec_conjuncion + proceso_may.count(conjuncion)

print("No. de conjunciones en el texto----: ", frec_conjuncion)
con.close()
```

## Apéndice B. Código fuente en Python, programa 2.

Este programa analiza lee la tabla con los elementos de la red de Petri y muestra un gráfico

```
# Creación de Redes de Petri a partir de la descripción de procesos de
negocios utilizando Python
# Segunda parte

import sqlite3
import pandas as pd
import pygame,sys
from pygame.locals import *

#---------------------------------
cadena=""
numero_filas=7
numero_columnas=6
matriz = []
colors = ["green", "blue", "orange", "purple", "white"]
for i in range(numero_filas):
    matriz.append([])
    for j in range(numero_columnas):
        matriz[i].append(None)
#---------------------------------
con = sqlite3.connect('BaseDatos.db')
cur = con.cursor()
df3 = pd.read_sql_query("SELECT * FROM dat_proceso order by
posicion_en_texto", con)
#"----------------------------
registros_en_proceso=df3.tipo_simbolo.count()
print("registros",registros_en_proceso)

x=0

for x in range (0,registros_en_proceso):
    simbolo=df3.iloc[x,1]
    if simbolo=="REC":
        cadena=cadena+"1"
    if simbolo=="CIR":
        cadena=cadena+"0"
    if simbolo=="LIN":
        cadena=cadena+"2"
    x=x+1
print(cadena)

pygame.init()
ventana=pygame.display.set_mode((800,600))
pygame.display.set_caption("Redes de Petri")
ColorDos = pygame.Color(200,60,80)
ColorTres = pygame.Color(80,70,120)
Blanco = pygame.Color(255,255,255)
ventana.fill(Blanco)
#------------------------

x=0
pos_X1_ventana=380 # Rectángulo
pos_X2_ventana=400 # Circulo
pos_X3_ventana=200 # Texto
pos_Y_ventana=20
miFuente=pygame.font.SysFont("courier10pitch",14)
estructura="SELECCION"
if estructura=="SELECCION":

    for x in range (0,registros_en_proceso):
        simbolo=df3.iloc[x,1]
        etiqueta=df3.iloc[x,2]
        if simbolo=="REC":
            if etiqueta != "ENTREGA":

pygame.draw.rect(ventana,ColorTres,(pos_X1_ventana,pos_Y_ventana,40,3
0))

pygame.draw.line(ventana,ColorDos,(pos_X1_ventana+20,pos_Y_ventana+
30),(pos_X1_ventana+20,pos_Y_ventana+100))
                pygame.draw.polygon(ventana, color=(255,0,0),
points=[(pos_X1_ventana+15,91), (pos_X1_ventana+20,100),
(pos_X1_ventana+25,91)])
                pygame.draw.polygon(ventana, color=(255,0,0),
points=[(394,208), (399,217), (404,208)])
                pygame.draw.polygon(ventana, color=(255,0,0),
points=[(394,290), (399,299), (404,290)])
                pygame.draw.polygon(ventana, color=(255,0,0),
points=[(485,300), (495,299), (495,290)])
                # 0 + 5 + 5
                # 0  + 9  0
            miTexto=miFuente.render(etiqueta,0,(200,60,80))
            ventana.blit(miTexto,(pos_X3_ventana,pos_Y_ventana+5))


        if simbolo=="CIR":

pygame.draw.circle(ventana,ColorTres,(pos_X2_ventana,pos_Y_ventana),20
)

            if x!=registros_en_proceso-1:
                print("registttt",registros_en_proceso)

pygame.draw.line(ventana,ColorDos,(pos_X2_ventana,pos_Y_ventana+20),(
pos_X2_ventana,pos_Y_ventana+100))
            long=len(etiqueta)

            print("etiqqqq",long)

            miTexto=miFuente.render(etiqueta,0,(200,60,80))
            ventana.blit(miTexto,(pos_X3_ventana,pos_Y_ventana-5))
        if simbolo=="LIN":
            print("A")

pygame.draw.line(ventana,Blanco,(pos_X2_ventana,pos_Y_ventana+20-
100),(pos_X2_ventana,pos_Y_ventana+100-100))

pygame.draw.line(ventana,ColorDos,(pos_X1_ventana+20,pos_Y_ventana+
15-190),(pos_X1_ventana+100+20,pos_Y_ventana+100-220))
            pos_Y_ventana=pos_Y_ventana-300
            pos_X1_ventana=pos_X1_ventana+100
            pos_X2_ventana=pos_X2_ventana+100
            pos_X3_ventana=pos_X3_ventana+330
        pos_Y_ventana=pos_Y_ventana+100
        print(pos_Y_ventana)
    x=x+1
#------------------------
while True:
    #ventana.fill(Blanco)
    for evento in pygame.event.get():
        if evento.type == QUIT:
            pygame.quit()
            sys.exit()
    #ventana.blit(miTexto,(300,25))

    pygame.display.update()
```

## Apéndice C. Script de la base de datos

Antes de ejecutar el script es necesario crear una base de datos denomina "BaseDatos"

```
PRAGMA foreign_keys = off;
BEGIN TRANSACTION;

-- Table: cat_conjunciones
CREATE TABLE cat_conjunciones (id_conjuncion INTEGER (7, 0),
des_conjuncion CHAR (10));
INSERT INTO cat_conjunciones (id_conjuncion, des_conjuncion) VALUES
(1, ' Y ');
INSERT INTO cat_conjunciones (id_conjuncion, des_conjuncion) VALUES
(2, ' O ');
INSERT INTO cat_conjunciones (id_conjuncion, des_conjuncion) VALUES
(3, ' AL MISMO TIEMPO ');
INSERT INTO cat_conjunciones (id_conjuncion, des_conjuncion) VALUES
(4, ' EN FORMA PARALELA ');

-- Table: cat_plazas
CREATE TABLE cat_plazas (id_plaza INTEGER PRIMARY KEY,
org_plaza CHAR (10), des_plaza CHAR (20));
INSERT INTO cat_plazas (id_plaza, org_plaza, des_plaza) VALUES (1,
'AGUA', 'CAJA');
INSERT INTO cat_plazas (id_plaza, org_plaza, des_plaza) VALUES (2,
'AGUA', 'POZO');
INSERT INTO cat_plazas (id_plaza, org_plaza, des_plaza) VALUES (3,
'AGUA', 'ALMACEN');
INSERT INTO cat_plazas (id_plaza, org_plaza, des_plaza) VALUES (4,
'AGUA', 'TESORERIA');
INSERT INTO cat_plazas (id_plaza, org_plaza, des_plaza) VALUES (5,
'AGUA', 'ATENCION A USUARIOS');
INSERT INTO cat_plazas (id_plaza, org_plaza, des_plaza) VALUES (6,
'AGUA', 'CONTRALORIA');
INSERT INTO cat_plazas (id_plaza, org_plaza, des_plaza) VALUES (7,
'AGUA', 'DIRECCION GENERAL');
INSERT INTO cat_plazas (id_plaza, org_plaza, des_plaza) VALUES (8,
'AGUA', 'DOMICILIO DEL USUARIO');
INSERT INTO cat_plazas (id_plaza, org_plaza, des_plaza) VALUES (9,
'AGUA', 'LECTURA DEL MEDIDOR');
INSERT INTO cat_plazas (id_plaza, org_plaza, des_plaza) VALUES (10,
'AGUA', 'RECIBO DE PAGO');
INSERT INTO cat_plazas (id_plaza, org_plaza, des_plaza) VALUES (11,
'AGUA', 'CAJERO AUTOMATICO');

-- Table: dat_proceso
CREATE TABLE dat_proceso (id_renglon INTEGER PRIMARY KEY,
tipo_simbolo CHAR (3), desc_ptc CHAR, posicion_en_texto INTEGER (7,
0), margen_superior INTEGER, margen_izquierdo INTEGER);

-- Table: dat_proceso_desln
CREATE TABLE dat_proceso_desln (id_proceso INTEGER PRIMARY
KEY, org_proceso VARCHAR, des_ln VARCHAR);
INSERT INTO dat_proceso_desln (id_proceso, org_proceso, des_ln)
VALUES (1, 'AGUA', 'El lecturista llega al domicilio del usuario toma
lectura del medidor al mismo tiempo entrega el recibo de pago.');

-- Table: verbos
CREATE TABLE verbos (verb_id INTEGER PRIMARY KEY, verb_org
CHAR (10), verb_trans);
INSERT INTO verbos (verb_id, verb_org, verb_trans) VALUES (1,
'AGUA', 'ENTREGA');
INSERT INTO verbos (verb_id, verb_org, verb_trans) VALUES (2,
'AGUA', 'TOMA');
INSERT INTO verbos (verb_id, verb_org, verb_trans) VALUES (4,
'AGUA', 'PAGA');
INSERT INTO verbos (verb_id, verb_org, verb_trans) VALUES (5,
'AGUA', 'ACUDE');
INSERT INTO verbos (verb_id, verb_org, verb_trans) VALUES (6,
'AGUA', 'LLEGA');
INSERT INTO verbos (verb_id, verb_org, verb_trans) VALUES (7,
'AGUA', 'LIQUIDAR');
INSERT INTO verbos (verb_id, verb_org, verb_trans) VALUES (8,
'AGUA', 'DEJA');

COMMIT TRANSACTION;
PRAGMA foreign_keys = on;
```