International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 15(2023) pp. 87-96
© MIR Labs, www.mirlabs.net/ijcisim/index.html

Submitted: 28 Nov., 2022; Accepted: 3 April, 2023; Publish: 9 June, 2023

Microservices identification through a multi-model approach

Malak saidi[1], **Anis Tissaoui**[2], **and Sami Faiz**[6]

[1]National School for Computer Science, Manouba, Tunisia
malaksaidi16@gmail.com

[2]VPNC Lab, University of Jendouba, Tunisia
anis.tissaoui@fsjegj.rnu.tn

[3]Higher Institute of Multimedia Arts, Manouba, Tunisia
sami.faiz@isamm.uma.tn

Abstract: **The life cycle of the company is characterized today by increasingly frequent phases of change induced by a continuous search for competitiveness. As a result, and in order to preserve their competitive advantages, companies must be able to react quickly to market movements, changes in the needs of their customers and business transformations of the various interlocutors. In this context, business processes playing a major role in the definition and management of information systems, must be flexible to meet the requirements of users and government regulations and all organizations must engage in the continuous maintenance of their business models.**
However, these models are monolithic which makes the task of maintenance difficult and very complicated
Partitioning the system into microservices is usually done intuitively, based on the experience of the system designers. But, if the functionalities of a system are strongly interconnected, it is difficult to decompose the system into appropriate microservices.
To meet so this challenge, this paper proposes a multi-model based on a set of business processes. This approach combines two different independent dimensions: structural dependency and data dependency. We will be based on three clustering algorithm in order to automatically identify potential microservices. ***Keywords***: monolithic architecture; microservices; business processes;structural dependency;data dependency

I. Introduction

With globalization and its strong pressure on today's companies are experiencing multiple changes [8][9] both in the forms of organization and in their way of designing and producing. Indeed, the life cycles of increasingly short products, the increased requirement for flexibility and frequent changes in technique and technology have forced companies to be more seriously concerned about their modeling in order to cope with all these complex factors.

Ensuring control of the evolution of each organization so becomes a crucial issue and requires rapid adjustment of the information system to increase its agility and the ability of teams to react easily.

In fact, despite the willingness of these organizations to remain proactive, the monolithic [10] nature of their information systems puts them in front of challenges related to the performance of its services, the cost of the technical infrastructure, development and maintenance. A monolithic application is typically an application system in which all of the relevant modules are packaged together as a single deployable unit of execution. These systems start small but tend to grow over time to meet business needs. At some point, as new features are added, a monolithic application can begin to suffer from the following problems:

- The individual parts of the system cannot be scaled independently, because they are tightly coupled.
- It is hard to maintain, because of tight coupling and hidden dependencies.

However, the micro-services architecture [4][8][22][23] was invented to solve the problems cited such as tight coupling and scalability because over time IT projects tend to grow and little by little we extend the existing functionalities, with many additions and few deletions, we ends up with a thousand complicated and difficult to manage functional tasks. The term micro-service[14][15] describes a set of small, independent, fine-grained, loosely coupled and highly cohesive services. Each micro-service performs its business process autonomously and communicates synchronously via an API or asynchronously via a lightweight messaging bus [7].

So far, the micro-services discovery exercise is done intuitively based on the experience of system designers and domain experts, mainly due to missing formal approaches and lack of automated tools support.

In this context, research work has been proposed recently

[13] [18] [19]. Although business process models are a rich reservoir of many details like who does what, when, where, and why, BPs seem almost neglected in the exercise of identifying micro-services. To our knowledge, Amiri [1], Daoud et al. [3] and Saidi et al. [14] [15] are the only ones to have adopted BPs in this discovery exercise.
A BP is defined as an orchestration of activities [6] that includes an interaction between several actors in order to achieve a well-defined organizational goal.
In this paper, we will propose a multi-model approach which aims to deal with the case of several variants of business processes in order to analyze the structural and data dependencies at first and to calculate the final dependency matrix based on our proposed formulas in a second step and finally generate the micro-services.
The rest of this paper is organized as follows. Section 2 presents the related work. Section 3 presents a case study, gives an overview of our approach to automatically identify micro-services from a set of BPs, and formalizes the control and data dependency models. Section 4 presents the experimentation of our proposed approach. Finally, we conclude with some future work.

II. Related work

A. Background

1) Business process

A business process model is an orchestration of activities that takes an input (of any form) adds value to it using resources and provides an output (product or service) that meets business goals.
It determines the following characteristics:

- Describes a dynamic view of the organization
- It is composed of sub-processes, and these are composed by atomic action elements.
- It represents a graph of activities that describes a sequence of activities necessary to achieve an organizational objective.
- It has a transaction function that controls the flow of the process.
- It can involve several functional units.

2) Monolithic system

The monolithic architecture describes the ancestor of all software architectures. Through this approach, the system was consolidated into a single large file, in which the various components were combined into a single, inseparable program.
This monolithic system depends on a single database. It is composed of four essential elements: A user interface, business logics, a data interface and a database.
With this architecture, it is difficult to make improvements without taking the risk of affecting other functionalities. It takes time for maintenance and updates, which increases the deployment cycle time.

3) Microservices

The year 2014 was considered to be a revolutionary year for companies and the software architectures they use. Originally originating from Netflix and Amazon, they arise from the strong need to partition both software development teams and runtime components to foster agile development and horizontal scalability.
The term microservice describes a set of small autonomous services, each of which carries out its business process autonomously and communicates synchronously via an API or asynchronously through a lightweight messaging bus.
Microservices therefore can be deployed independently on fully automated deployment machines around enterprise capabilities.
Eight aspects characterize a microservice:

- Unique feature.
- Technological flexibility.
- Reduced team
- Targeted deployment
- Scalability
- Easy to test
- Reusability
- Autonomy

4) Dependency

It makes it possible to model the inter-process links which have a high impact on the maintenance and management of an organization's business processes.

5) Clustering

Data partitioning is a widely used technique in data analysis. Its objective is to divide a set of data into different homogeneous clusters, in the sense that the data of each subset share common characteristics, which most often correspond to proximity criteria that are defined by introducing measures and distance class between objects.
There is supervised clustering where the number of classes is known in advance and unsupervised clustering where the number of classes is unknown.

B. Related work

Compared to monolithic systems, microservices have become the software architecture of choice for business applications. There are a very large number of applications that migrate to a micro-services architecture.
Since enterprise developers are faced with the challenges of maintenance and scalability of increasingly complex projects, in [5], Escobar et al. proposed a model-based approach to analyze the current application structure and the dependencies between business capability and data capability. This approach aims to break down an application developed in J2EE into micro-services through diagrams resulting from the analysis of data belonging to each EJB

(Enterprise Java Beans) using the clustering technique.
In [4], Dojic et al. presented an approach allowing the reconstruction of an integration platform that is based on an SOA approach into a new microservices-oriented platform. This new platform makes it possible to cover the gaps related to the number of messages that must be processed as well as the number of new integrations that must be supported by the system. This new approach helps ensure flexibility and agility in maintaining, upgrading and deploying finished software products.
In [2], Chen et al. have proposed an approach that performs top-down, microservice-driven analysis. This approach is essentially based on data from the business logic. They developed an algorithm that generates the decomposition of the system into candidate microservices which will provide more rational, objective and easy to understand results afterward thanks to objective operations and data extracted from the business logic.
In [8] , Baresi et al. have proposed a semi-automatic approach allowing the discovery of microservices. Indeed, a Restful API is proposed in order to receive as input a business model components which describes the flow of requirements with its inputs, and outputs and the microservices are generated afterwards by the processing of this model. This approach is based on the calculation of the semantic similarity of the functionalities described through the openAPI specifications. By leveraging a reference vocabulary, the approach generates potential candidate microservices as fine groups of cohesive operations.

In [13], Gysel et al. proposed a semi-automated service identification model according to predefined categories, based mainly on requirements artefacts. Through approximation algorithms and through the use of weights, the determined services would be directed to a specific category. This approach defines the concept of coupling criteria maps using 16 different instances grouped into four categories: Cohesion, Compatibility, Constraints and Communications.
Munezaro et al. in [20] uses domain-driven design (DDD) patterns as input to determine potential microservices. First, developers determine a domain using a pervasive language. Domain experts break down the boundaries of each system responsibility and represent it as a business capability, where a business capability is something a system does to achieve a specific business goal. Each business feature represents a micro-service.
In [21], Sellami et al. have proposed a method for identifying candidate micro-services from the source code of a given application. This approach is based on the calculation of similarity and dependence between the different classes of the system according to the interactions and the terminology of the domain used in the code and to do this, the authors have thought of an algorithm which is based on the density in order to determine recommended micro-services by identifying potential classes.

Despite the business process is a central and crucial element in the evolution of the company, only four works that took the business process as an input system to discover the appropriate micro-services.
In [1], Amiri proposed a technique to discover potential micro-services from a set of BPs. To this end, they considered two aspects of structural dependency and data dependency.
The approach is essentially based on three steps: First, a TP relationship that shows the structural dependence of activities within a business process. Next, a TD relationship is defined to show the dependency of activities based on their used data objects.
Recently in [3],Daoud et al. was proposed to remove and deal with the limits of the approach of Amiri already mentioned in their work [1]. The essential goal of the approach is to automatically identify micro-services based on two dependency models (control and data) and using collaborative clustering. To do this Daoud et al. proposed formulas for calculating direct and indirect control dependencies as well as proposed two strategies for calculating data dependency.
In [15], Saidi and al. proposed an extension of the control model of Daoud and al.[3] They proposed four calculation formulas to calculate the dependence taking into account the case of loop$_{sequence}$, loop$_{And}$, loop$_{Xor}$, loop$_{Or}$ in order to calculate the dependence matrix of control to subsequently generate the appropriate micro-services. In [14], Saidi et al. presented an approach based on association rules to calculate the correlation between the attributes of the set of activities and to determine a dependency matrix based on the strong and weak associations.
The only paper that addressed the problem of identifying micro-servives from a set of business processes is the approach of Amiri and al. [1].
For this reason, our main objective in this paper is to take several independent business processes as system input and identify the candidate micro-services using three different clustering algorithms. The authors in [17] described an approach which is based on global K-means algorithm. This incremental approach makes it possible to add one center cluster through a deterministic search technique made up of N execution of the algorithm starting from the appropriate initial position.

III. Our approach for identifying microservices

A. Our case study

The post-production process involves a huge number of professionals - editors, sound engineers, etc. It includes the raw video editing process, sound mixing, visual effects, color correction and grading, and final soundtrack development and placement.
Post-production refers to all the tasks associated with cutting out raw footage, assembling that footage, adding music, dubbing, sound effects, to name a few.
Indeed, the post-production process is described as being a collaborative system, over a few months or even a year, depending on the size and requirements of the project.
Figure 1 shows a set of independent image post-production processes.
A Business process is a series of logically linked activities in order to achieve a well-determined organizational goal.

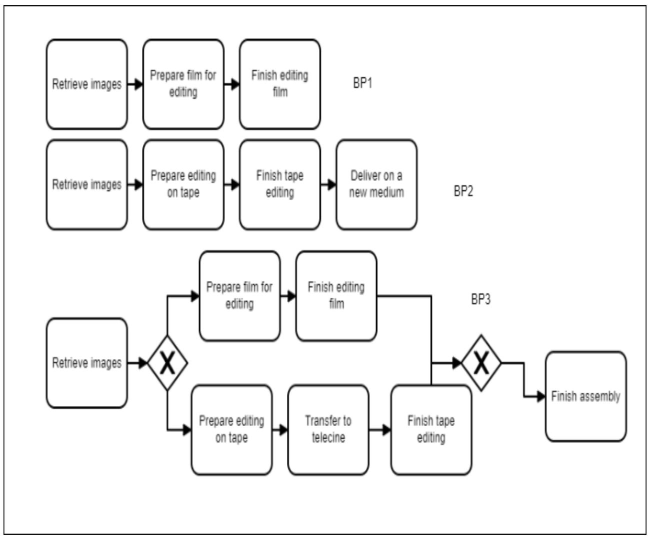


Figure. 1: Three independent BP of the picture post-production process

”Logically linked” means that there is some dependency between these activities: either a structural dependency, or a data dependency... Indeed, the business process model in general takes as input data of any form, adds value to it by using resources (human, material, ...) in order to achieve an objective by providing an output (product or service).
Structurally, a process model structurally is a graph composed of nodes of type activity, gateway and arcs which are based on these elements. Activities describe the tasks performed in the process. Gateways are used to represent alternate and parallel branches and merges. They can be of type OR or XOR (inclusive execution, exclusive execution) and AND.
Our example in figure 1 is represented in BPMN.

B. *Foundations*

Compared to the work presented in the related work section, business process models will be our main source for identifying candidate microservices. These processes (like our example in figure 1) are representative processes of the tasks [6] of the company independently of the human and technical means. The microservices we aim to determine from a set of business processes should be fine-grained, strongly cohesive (explains how the activities that build a given microservice go together) and loosely coupled (describes how the microservices can be replaced without affecting the proper functioning of the overall system). According to our proposed approach, we were able to identify two types of dependencies which are given below.

- **Structural Dependency:** This dependency refers to the structure of our business model. Indeed, if 2 activities are linked via a structural dependency, they will form a fine-grained micro-service with a loose coupling. Otherwise, they form separate microservices.[1]
- **Data Dependency:** Technological development continues to grow, the multitude of data sources is increasingly diverse and varied. The representation and presentation of information become even more abstract. The need to equip oneself with tools for analyzing and extracting these colossal collections of data becomes more than vital. The goal is to discover interesting associations or correlations between items in these large collections and databases. These elements can be: attributes, objects, individuals, Items ... etc.
An association rule takes the form X⇒Y, X in association with Y, which means that transactions or queries that contain the set of objects X tend to include the objects of the set Y Finding association rules is an important process in data mining.
This dimension so is based on association rules to determine the low and high correlation between the different attributes of a given pair of activities[14].

C. *The main steps of our approach*

We identify three essential steps in our proposed architecture (Figure 2)

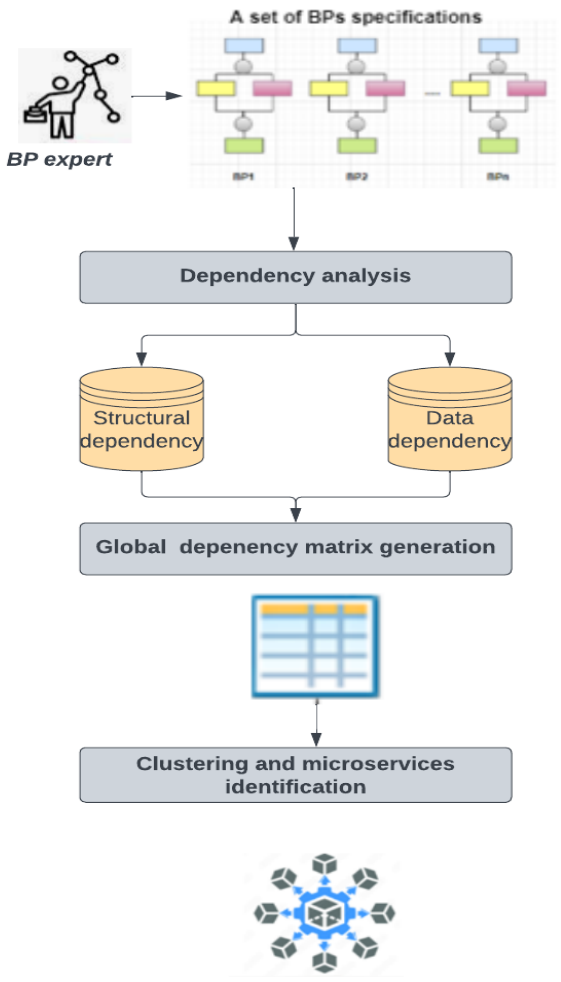

A set of BPs specifications
BP expert
BP1
BP2
BPn
Dependency analysis
Structural dependency
Data dependency
Global depenency matrix generation
Clustering and microservices identification

Figure. 2: Our architecture

- **Dependency analysis:** This first step allows us to identify the specifications of each BP model taken as input to the system and allows us to determine the dependence, according to two dimensions: A structural dimension (by analyzing the structure of the model)

and a given dimension (by using the technique of association rules in order to determine the strong and weak associations between attributes of each activity of our BP). Indeed, we will determine for each BP its own structural dependency matrix. So we will treat each BP as an undirected graph and we will determine the adjacency matrix linked to each model.
In the same way, we extract the matrices of each business process according to the second dimension (data) and by analyzing the model in terms of correlation between the shared attributes of each activity.
We can say that for n processes, we will have 2n dependency matrices.

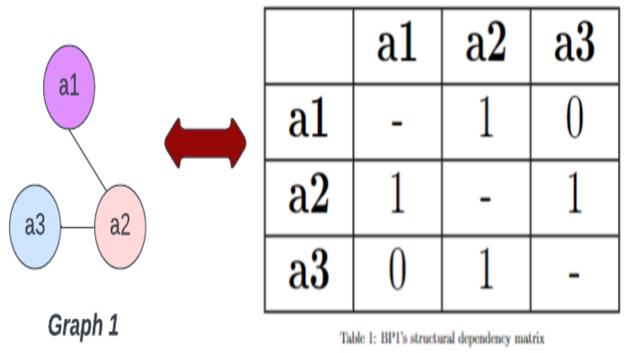
- **Global dependency matrix generation:** In this second step, we will base ourselves on all the matrices calculated in the previous step so that we can determine our global dependency matrix.
To do this, formulas have been proposed for the aggregation of these matrices. This part will be described in detail below.

- **Clustering and micro-services identification:** We will base ourselves in this last step on three clustering algorithms which take as input the global dependency matrix generated so that we can determine the potential micro-services. Each cluster is formed by a set of activities that form a candidate microservice.

D. Micro-services identification

1) A. Matrix aggregation formula

- **Control dependency formula**
Let G be an undirected graph generated from a BP represented in BPMN. This graph has a set of K vertices (as shown in figure 3, figure 4 and figure 5).
M=(a_{ij}) where a_{ij} describes the number of arcs that connect vertex i to vertex j.
For graphs with at most one edge or edge between two vertices, we have $a_{ij} \in \{0, 1\}$.
To therefore evaluate a structural dependence between two given activities (ai, aj), if there is a direct arc or a logical connector that directly links these two activities, we will assign the value 1 otherwise the value will be 0.
In the end, by applying this calculation method we can generate the adjacency matrix of each model separately (in our case we will have three adjacency matrices) and then we add up to generate a single output matrix by aggregating these three adjacency matrices.
red Note: black By aggregating if there are two or more different dependency values in all of our three models, we apply the formula below:
Dep(a_i, a_j)= Max (val1,val2..valn).
If we take the case of our first BP already represented previously in our motivation example in figure 1 and try to analyze the dependence in terms of structural dependence, the first adjacency matrix generated for the first BP is calculated in the figure 3.
The other two matrices of BP2 and BP3 are shown in

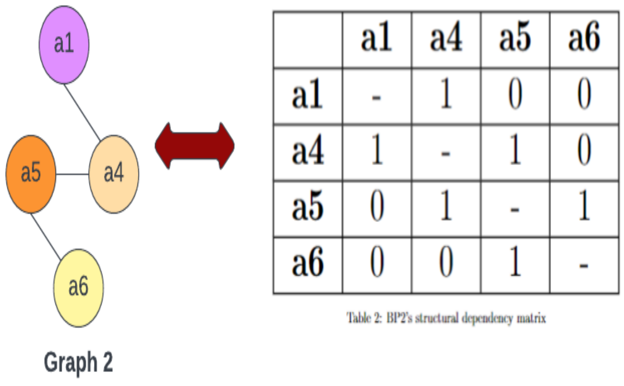


	a1	a2	a3
a1	-	1	0
a2	1	-	1
a3	0	1	-

Table 1: BP1's structural dependency matrix

Figure. 3: BP1's structural dependency

Figure 4 and Figure5.



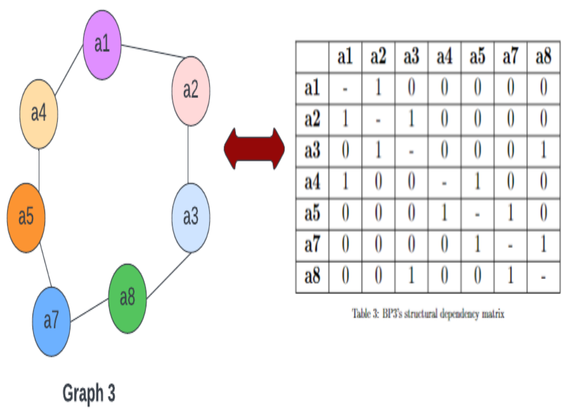
	a1	a4	a5	a6
a1	-	1	0	0
a4	1	-	1	0
a5	0	1	-	1
a6	0	0	1	-

Table 2: BP2's structural dependency matrix

Figure. 4: BP2's structural dependency

Our global structural dependency matrix of the three Bps is represented by the table 1

- **Data dependency formula:**
The activities of our BPs, which are interdependent, represent a strong dependence in terms of structural and logical relations which control the routing and the order of execution of these activities and in terms of association between attributes.
Indeed, to migrate towards an architecture based on microservices, we must guarantee strong cohesion and loose coupling.



	a1	a2	a3	a4	a5	a7	a8
a1	-	1	0	0	0	0	0
a2	1	-	1	0	0	0	0
a3	0	1	-	0	0	0	1
a4	1	0	0	-	1	0	0
a5	0	0	0	1	-	1	0
a7	0	0	0	0	1	-	1
a8	0	0	1	0	0	1	-

Table 3: BP3's structural dependency matrix

Figure. 5: BP3's structural dependency

	a1	**a2**	**a3**	**a4**	**a5**	**a6**	**a7**	**a8**
a1	-	1	0	1	0	0	0	0
a2	1	-	1	0	0	0	0	0
a3	0	1	-	0	0	0	0	1
a4	1	0	0	-	1	0	0	0
a5	0	0	0	1	-	1	1	0
a6	0	0	0	0	1	-	0	0
a7	0	0	0	0	1	0	-	1
a8	0	0	1	0	0	0	1	-

Table 1: Global structural dependency matrix

The association rules technique will allow us to identify weak and strong associations. In other words, we will determine the data dependency between activities via the correlation between the different attributes of our system.
Indeed, the relationships discovered can be modeled in the form of association rules or a set of frequent elements.
Let I = (i1, i2, ... in) be a set of binary attributes distinct from the database, and A = (a1, a2, .. an) be a set of activities.
An activity being a subset of elements I such that A$\subseteq I$.
Let D be the database containing all the activities. Each activity a is represented by a binary vector with a[i]=1 if the activity shares the attribute, otherwise a[i]=0.
A non-empty subset X = $\{i1, i2, ...\}$ of A is called itemsets, and we denote it I.
The length of I is given by the value of k corresponds to the number of items contained in X, it is noted: K-itemsets.
An association rule is a set of 2-tuple elements (X, Y) of A representing an implication of the form $X \rightarrow Y$ with $X \subset I, Y \subset I$.
It is generally expressed by: if (x1, x2, ... xn) then (y1, y2, .. yn).
Through this dimension, each microservice has its own database to minimize communication. This is because activities that share attributes will most likely be categorized into the same microservice.
In this dimension, so we will reuse what we have done in [14] in order to measure the dependence of data between a couple of given activities.
For each pair of activities ai and aj the value of Dep (ai, aj) is same in all the processes (if the process has both activities), because an activity even in different processes use the same set of attributes, therefore we will use a single binary representation containing all the activities of our process models and we will apply the algorithm for generating the final dependency matrix proposed in [14].

The table 2 analyzes the three BP models in terms of data.
To generate the data dependency matrix described in table 2,

	a1	**a2**	**a3**	**a4**	**a5**	**a6**	**a7**	**a8**
a1	-	4.44	6.57	38.03	20.05	37.18	44.43	38.03
a2	4.44	-	6.57	38.03	44.43	37.18	44.43	38.03
a3	6.57	6.57	-	38.03	22.18	44.43	44.43	38.03
a4	38.03	38.03	38.03	-	44.43	44.43	44.43	44.43
a5	20.05	44.43	22.18	44.43	-	44.43	44.43	44.43
a6	37.18	37.18	44.43	44.43	44.43	-	44.43	44.43
a7	44.43	44.43	44.43	44.43	44.43	44.43	-	44.43
a8	38.03	38.03	38.03	44.43	44.43	44.43	44.43	-

Table 2: Data dependency matrix

we used the a priori algorithm implemented in [14]. Indeed, we set the minimum support value to 0.5 and the minimum confidence value of 0.7 to generate the set of association rules that will be used later by the dependency calculation algorithm [14].

2) B. Micro-services generation :

After calculating the two matrices across the two dimensions (structural and data), we will use an aggregation formula so that we can determine the final dependency matrix of our example.
Dep$_G$= $Sum\left(Mi\left(ai,aj\right),Mj\left(ai,aj\right)\right)$
Our final dependency matrix is shown in Table 3 . After calculating the final matrix, we will use three different clustering algorithms (partitional clustering, hierarchical clustering, and distribution-based clustering) to generate our potential microservices.

	a1	**a2**	**a3**	**a4**	**a5**	**a6**	**a7**	**a8**
a1	-	5.44	6.57	39.03	20.05	37.18	44.43	38.03
a2	5.44	-	6.57	38.03	44.43	37.18	44.43	38.03
a3	6.57	6.57	-	38.03	22.18	44.43	44.43	39.03
a4	39.03	38.03	38.03	-	45.43	44.43	44.43	44.43
a5	20.05	44.43	22.18	45.43	-	45.43	45.43	44.43
a6	37.18	37.18	44.43	44.43	45.43	-	44.43	44.43
a7	44.43	44.43	44.43	44.43	45.43	44.43	-	45.43
a8	38.03	38.03	39.03	44.43	44.43	44.43	45.43	-

Table 3: Final dependency matrix

Table 4 summarizes the microservices generation results for each algorithm used. Each cluster describes a candidate micro-services.

Technique	class number	classified activities's number	largest class size
k-means algorithm	4	8	3
agglomerative algorithm	2	8	4
GMM algorithm	4	8	3

Table 4: classification result of the different clustering methods

IV. Experimentation

A. Implementation

Our proposed tool is based on five essential modules as it is modeled on our proposed architecture (figure 6).

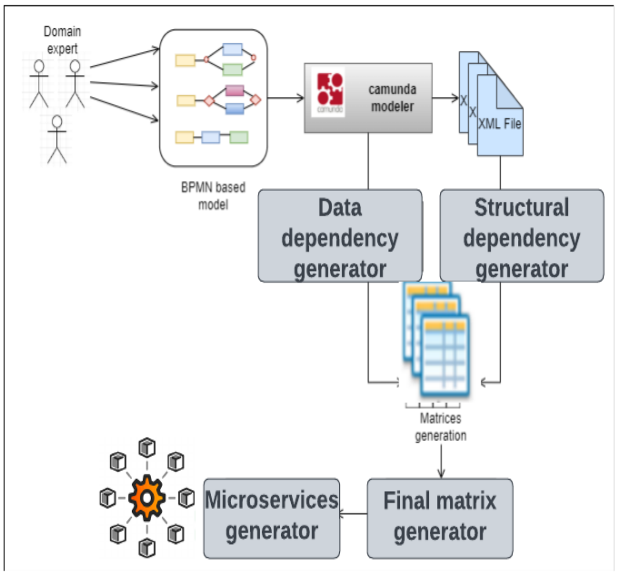


Figure. 6: Micro-services identification architecture

- **Camunda modeler :** It is the tool that will allow us to model our system input and to generate two types of output: either a graphic model of BP, or an XML file of BP created and which will be used later as input in the second module.
- **Structural dependency generator:** Based on the XML file generated by **Camunda modeler**, the structural dependency module calculates the dependencies between each pair of activities (ai, aj). These dependencies are represented in the form of an adjacency matrix using the method described above.
- **data dependency generator:** Each BP created is essentially based on a set of artifacts and attributes. Extracting frequent patterns is considered our first step in generating association rules.
  It allows us to extract the context of the set of binary attributes (I). This step takes as input a database and a minimal support to give as output a set of frequent elements with their supports.
  First, we calculate the support of the elements and remove those that do not reach the minimum support. Then, we calculate the support of the itemsets of level (n + 1) and we remove those which have a support lower than a minimal support. Therefore, frequent patterns are iteratively computed in ascending order according to their sizes.

```
Algorithm 1: Itemset generation
Input: Binary database BD, minimal support minsupp
Output: Frequent itemset $\mathcal{K}_i$
begin
    $\mathcal{K}_i=\emptyset$; i=0
    $\mathcal{L}_1$= The condidate itemsets with size 1 in B
    $\mathcal{K}_1$= The frequent itemsets of $\mathcal{L}_1$
    if $\mathcal{K}_{i+1}$= not empty then
        $\mathcal{L}_{i+1}$= Candidate-gen (Ki);
        $\mathcal{K}_{i+1}$= frequent itemset of $\mathcal{L}_{i+1}$;
        $i++$
    return $\cup\, \mathcal{K}_i$
```

  We have implemented Algorithm 1 to determine the itemsets.
  Since we are working on different BP models, this module makes it possible to define the correlation determined between the attributes of the three proposed business models in order to determine the activities which will be classified in the same cluster and those which will be classified in a different cluster. This module is essentially based on the method already proposed in [14].
- **Final matrix generator:** For a set of n BP, we will have n matrices at the level of the first dimension (structural dependency) and the second dimension, which is based on the extraction of strong and weak associations between activities.
  Therefore, we proposed to make an aggregation of n matrices generated for the first dimension and suddenly we will have in output a single matrix instead of n matrices.
  For the given dimension, if a given couple of activities (ai, aj) is the same in the other variants of BP, then it will be the same dependency value, otherwise, this value is recalculated by applying the technique proposed by Saidi and al in [14].
  As a result, we will have two matrices. We proposed to take the "**Sum**" of the two dependencies calculated for a couple of activities (ai, aj) in order to generate the global dependency matrix.
- **Micro-services generator:** This module is essentially based on the final matrix calculated in the previous module "Final matrix generator" by applying three different clustering algorithms to determine our potential fine-grained micro-services, with loose coupling and strong cohesion.

B. Experiments

First, we chose to calculate the appropriate cluster number using the "Elbow" method: Elbow is the point where the rate

of decrease in average distance, i.e. SSE, will not change significantly with increasing number of clusters.
According to Elbow result, the appropriate number of clusters in our case is equal to 2 (figure7).

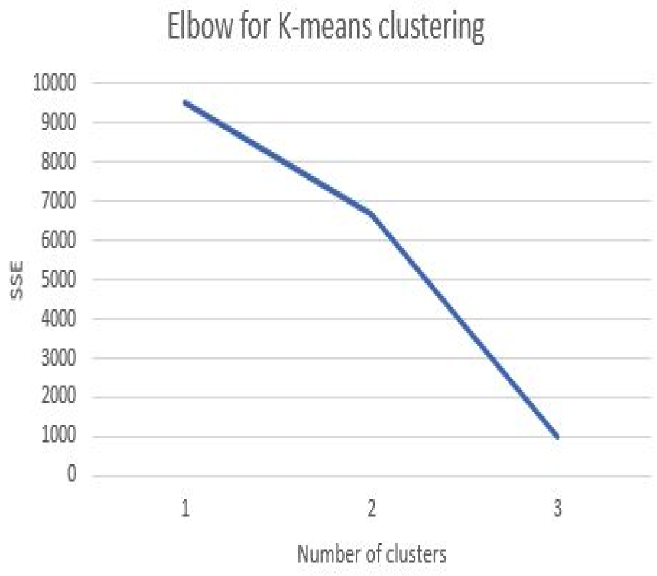
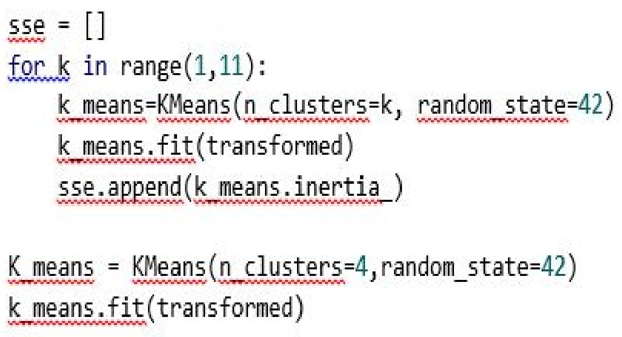


Figure. 7: Elbow

- **Partitional clustering:** There is a classic heuristic for this problem, often referred to as kmaverage methods, which is used for most applications. The problem is also studied as a classical optimization problem, with for example approximation algorithms. In particular, kmeans is used in unsupervised learning where observations are divided into k partitions. Dynamic clusters are a generalization of this principle, each partition being represented by a kernel which can be more complex than the average. The classic kmeans algorithm is the same as the LloydMax quantization algorithm.
  From the implementation of the K-means algorithm (figure 8), we were able to determine four microservices.
  The first microservice is formed by two activities (a0,a6), the second is described by three activities (a4,a5,a7), the penultimate is composed by a single activity (a1) and the last microservice is described by two activities (a2,a3).

- **Hierarchical clustering:** In the IT field, particularly in the fields of automated data analysis and classification, the notion of hierarchical grouping involves various clustering techniques and comes in two main families: ”bottom-up” and ”descendant” techniques. .
  The so-called ”top-down” method starts from a general solution to a more specific solution. The methods in this category start with a single cluster, including the whole, then divide at each step according to certain criteria until a series of individual clusters is obtained. There are two types of hierarchical clustering, Agglomerative and Divisive.
  Indeed, the agglomerative algorithm (figure 9) allowed

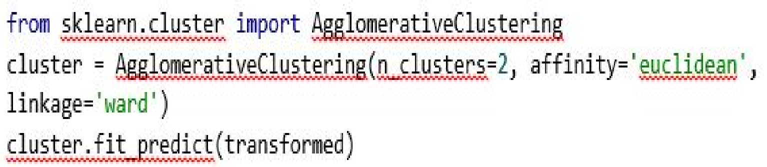


```
sse = []
for k in range(1,11):
    k_means=KMeans(n_clusters=k, random_state=42)
    k_means.fit(transformed)
    sse.append(k_means.inertia_)

K_means = KMeans(n_clusters=4,random_state=42)
k_means.fit(transformed)
```

Figure. 8: K-means

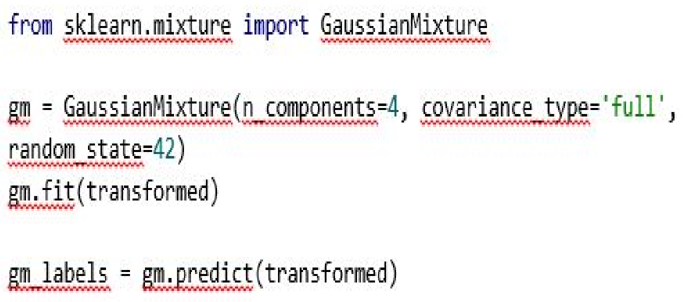
us to identify only two microservices. The first microservice is formed by four activities (a0,a4,a6,a7), the second is described by four activities also (a1,a2,a3,a5).



```
from sklearn.cluster import AgglomerativeClustering
cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean',
linkage='ward')
cluster.fit_predict(transformed)
```

Figure. 9: agglomerative algorithm

- **Distribution-Based Clustering:** Gaussian mixture models (GMM) assume that there are a number of Gaussian distributions, and each of these distributions describes a microservice. Since, the implementation of the GMM algorithm, we were able to determine four microservices.
  The first microservice is formed by three activities (a4,a7,a5), the second is described by two activities (a2,a3), the penultimate is composed by a single activity (a1) and the last microservice is described by two activities (a0,a6). The following figure (figure 10) shows the implementation of our GMM algorithm.



```
from sklearn.mixture import GaussianMixture

gm = GaussianMixture(n_components=4, covariance_type='full',
random_state=42)
gm.fit(transformed)

gm_labels = gm.predict(transformed)
```

Figure. 10: GMM algorithm

According to the comparison we made (figure 11), we find that the K-means algorithm gives better results of microservices generation compared to the others.

According to Dunn index (figure12), it can be concluded that for a given business process model, the given dimension is always richer in information compared to our first structural dimension. Regarding the comparison result for the aggregation of the two dimensions together, we notice that it is much better and more informative and suddenly the quality of generation of microservices will be better.

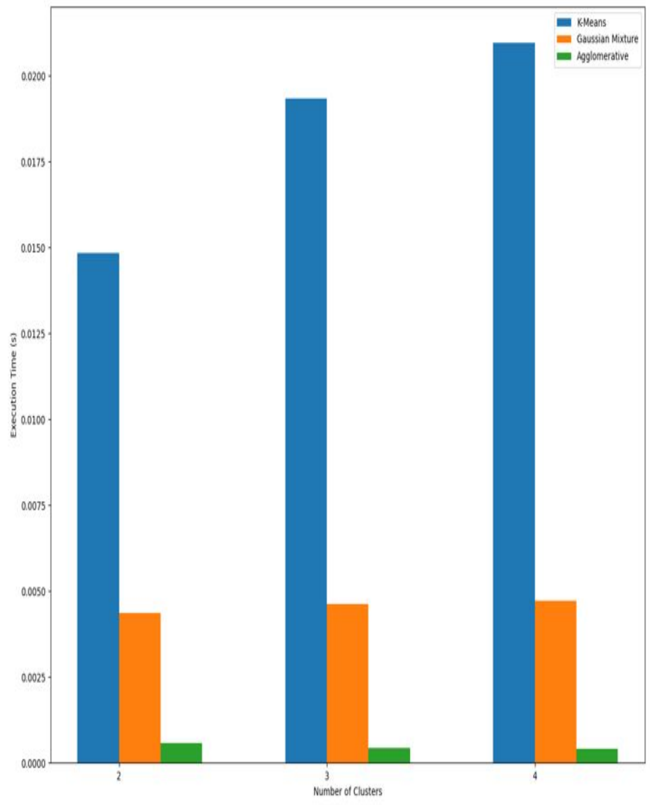


Figure. 11: Qualitative comparison

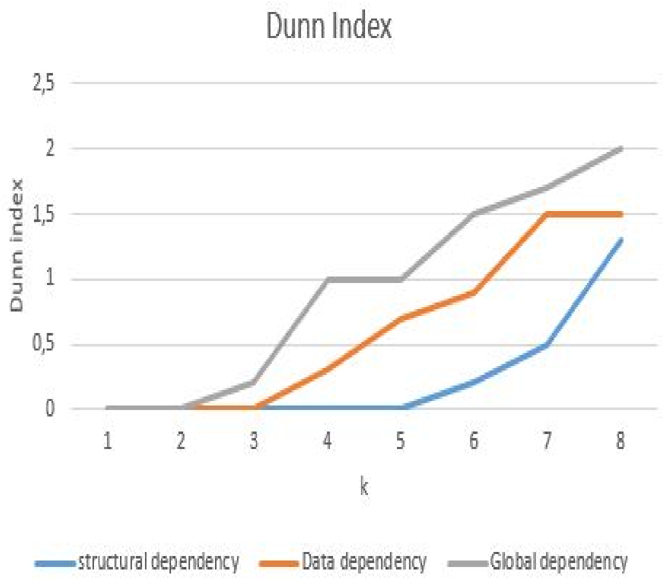


Figure. 12: Dunn index

V. Conclusion

Unlike monolithic systems, microservices were defined to overcome the shortcomings of monoliths and to evolve with continuously changing market demands.
Indeed, these microservices break down a monolithic system into a set of autonomous and independent services. This migration will allow us to optimize our resources, to improve collaboration and above all to rationalize the business processes of a company.
In this context, we thought about using a set of business processes as a monolithic system and aimed to divide our suggested system into small, self-contained services that can be deployed and individualized separately.
This proposed approach calculates the dependence between a given couple of activities taking into account the structural and data aspect of a set of independent BPs. Subsequently, and by implementing three different clustering algorithms, we were able to determine our candidate micro-services.
As future work, we first aim to treat the dependence between a couple of activities taking into account other aspects such as the semantic aspect and the security aspect with a set of BPs and then we will try to adapt these dimensions with other types of system input such as configurable process models.

References

[1] Amiri, M. J. (2018, July). Object-aware identification of microservices. In 2018 IEEE International Conference on Services Computing (SCC) (pp. 253-256). IEEE.

[2] Chen, R., S. Li, and Z. Li. ”From monolith to microservices: a dataflow-driven approach. In 2017 24th Asia-Pacific Software Engineering Conference (APSEC)(pp. 466–475).” (2017).

[3] Daoud, M., Mezouari, A.E., Faci, N., Benslimane, D., Maamar, Z., Fazziki, A.E. (2020). Automatic Microservices Identification from a Set of Business Processes. In: Hamlich, M., Bellatreche, L., Mondal, A., Ordonez, C. (eds) Smart Applications and Data Analysis. SADASC 2020. Communications in Computer and Information Science, vol 1207. Springer, Cham.

[4] Djogic, E., Ribic, S., and Donko, D. (2018). Monolithic to microservices redesign of event driven integration platform. 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 1411-1414.

[5] Escobar, D., Cárdenas, D., Amarillo, R., Castro, E., Garcés, K., Parra, C., and Casallas, R. (2016, October). Towards the understanding and evolution of monolithic applications as microservices. In XLII Latin American computing conference (CLEI) (pp. 1-11). 2016 .

[6] Ferchichi, A., Bourey, J.P., Bigand, M.: Contribution à l'integration des processus metier:application a la mise en place d'un referentiel qualite multi-vues. Ph.D. thesis, Ecole Centralede Lille; Ecole Centrale Paris (2008)

[7] Indrasiri, K., and Siriwardena, P. (2018). Microservices for the Enterprise. Apress, Berkeley.

[8] Baresi, L., Garriga, M., and Renzis, A. D. : Microservices identification through interface analysis. In European Conference on Service-Oriented and Cloud Computing (pp. 19-33). Springer, Cham.2017.

[9] Kherbouche, M. O. : Contribution à la gestion de l'évolution des processus métiers (Doctoral dissertation, Université du Littoral Côté d'Opale).2013.

[10] PPonce, F., Márquez, G., and Astudillo, H. (2019, November). Migrating from monolithic architecture to microservices: A Rapid Review. In 38th International Conference of the Chilean Computer Science Society (SCCC) (pp. 1-7). IEEE. 2019.

[11] Richardson, C.: Pattern: monolithic architecture. Dosegljivo: https://microservices. io/patterns/monolithic. html (2018)

[12] Estanol, Montserrat. "Artifact-centric business process models in UML: specification and reasoning." (2016).

[13] Gysel, M., Kölbener, L., Giersche, W., Zimmermann, O. (2016, September). Service cutter: A systematic approach to service decomposition. In European Conference on Service-Oriented and Cloud Computing (pp. 185-200). Springer, Cham.

[14] Saidi, M., Daoud, M., Tissaoui, A., Sabri, A., Benslimane, D., Faiz, S. (2022). Automatic Microservices Identification from Association Rules of Business Process. In: Abraham, A., Gandhi, N., Hanne, T., Hong, TP., Nogueira Rios, T., Ding, W. (eds) Intelligent Systems Design and Applications. ISDA 2021. Lecture Notes in Networks and Systems, vol 418. Springer, Cham.

[15] Saidi, M., Tissaoui, A., Benslimane, D., Faiz, S. (2022). Automatic Microservices Identification Across Structural Dependency. In: , et al. Hybrid Intelligent Systems. HIS 2021. Lecture Notes in Networks and Systems, vol 420. Springer, Cham.

[16] Cheung, Yiu-Ming. "k-Means: A new generalized k-means clustering algorithm." Pattern Recognition Letters 24.15 (2003): 2883-2893.

[17] Likas, Aristidis, Nikos Vlassis, and Jakob J. Verbeek. "The global k-means clustering algorithm." Pattern recognition 36.2 (2003): 451-461.

[18] Levcovitz, A., Terra, R., and Valente, M. T. (2016). Towards a technique for extracting microservices from monolithic enterprise systems. arXiv preprint arXiv:1605.03175.

[19] Mazlami, G., Cito, J., and Leitner, P. : Extraction of microservices from monolithic software architectures. In 2017 IEEE International Conference on Web Services (ICWS) (pp. 524-531). IEEE.2017.

[20] Josélyne, M. I., Tuheirwe-Mukasa, D., Kanagwa, B., and Balikuddembe, J. (2018, May). Partitioning microservices: A domain engineering approach. In Proceedings of the 2018 International Conference on Software Engineering in Africa (pp. 43-49).

[21] Sellami, K., Saied, M. A., and Ouni, A. (2022, June). A hierarchical dbscan method for extracting microservices from monolithic applications. In Proceedings of the International Conference on Evaluation and Assessment in Software Engineering 2022 (pp. 201-210).

[22] Heinonen, Riku. "Algorithmic Identification of Microservice Candidates." (2023).

[23] Qian, Lifeng, et al. "Microservice extraction using graph deep clustering based on dual view fusion." Information and Software Technology 158 (2023): 107171.