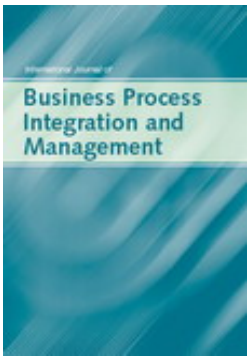


International Journal of Business Process Integration and Management

ISSN online: 1741-8771 - ISSN print: 1741-8763
https://www.inderscience.com/ijbpim

Microservices extraction through set of business processes variants

Malak Saidi, Anis Tissaoui, Sami Faiz

DOI: 10.1504/IJBPIM.2024.10065254

Article History:

Received:	01 July 2023
Last revised:	01 July 2023
Accepted:	26 March 2024
Published online:	25 July 2024

Microservices extraction through set of business processes variants

Malak Saidi*

National School for Computer Science,
Manouba, Tunis, Tunisia
Email: malaksaidi16@gmail.com
*Corresponding author

Anis Tissaoui

Faculty of Law, Economics and Management,
Jendouba, Tunis, Tunisia
Email: anis.tissaoui@fsjegj.rnu.tn

Sami Faiz

Higher Institute of Multimedia Arts,
Manouba, Tunis, Tunisia
Email: sami.faiz@isamm.uma.tn

Abstract: Managing multiple variations and versions of the same business process is a common practice in many companies. Indeed, the executions of these processes differ from each other according to human and contextual factors or also according to deliberate managerial decisions. These multiple variants describe the different paths that the activities can take according to these mentioned variations, which offer better flexibility, efficiency and adaptability for any organisation. However, these variants constitute a monolithic system with components that are strongly coupled. The latter can become more complex as it develops, with intertwined processes and tight dependencies between features. Thus all the variants of BP will be bulky and difficult to manage. Additionally, changes or updates in a monolithic system may require significant changes across BP, which may lead to regression risks or maintenance challenges. In this context, we aimed to propose an approach to migrate our monolithic system to a system with components that are loosely coupled, strongly cohesive and fine-grained. It is a multi-model approach representing a control and data dependency model and which takes as system input a set of process variants. We used three different clustering algorithms to generate our candidate micro-services.

Keywords: multiple variations; monolithic system; control and data dependency; micro-services.

Reference to this paper should be made as follows: Saidi, M., Tissaoui, A. and Faiz, S. (2024) 'Microservices extraction through set of business processes variants', *Int. J. Business Process Integration and Management*, Vol. 11, No. 3, pp.186–198.

Biographical notes: Malak Saidi received her Master's in Computer Science from the University of Management, Economics and Law of Jendouba (Tunisia) in 2018. She is currently a PhD student at the Higher National School of Computer Sciences, Manouba, Tunisia. She is a member of DocSys Lab Of FSJEGJ, Jendouba. Her current research interests lie in the area of business process and microservices identification. She has published papers in international and national conferences: SITIS 2018, ICOST 2020, HIS 2021, ISDA 2021, Systems Engineering 2022 and ISDA 2022.

Anis Tissaoui is an Assistant Professor of Computer Science with Jendouba University, Tunisia and a member of DOCSYS Research Team at VPNC Research Laboratory. He received his PhD in Computer Science from Paul Sabatier University, France in 2013. His research interests include knowledge acquisition and modelling, natural language processing and terminology-based knowledge engineering. His work benefits from collaborations with linguists and researchers in human factors. His contributions include methods and platforms for ontology engineering from text, tools for pattern-based semantic relation extraction, and knowledge representation to add lexical data to ontologies, with applications in semantic annotation and information retrieval.

Sami Faiz received his PhD in Computer Science from Orsay University (Paris 11). He is currently a Full Professor in Computer Science with the Higher Institute of Multimedia Arts of Manouba and a member of the Laboratory of Remote Sensing and Spatial Information Systems. He published more than 150 papers in specialised conferences and journals. He is also an author of three books in the framework of GIS.

This paper is a revised and expanded version of a paper entitled 'Automatic microservices identification across structural dependency' presented at International Conference on Hybrid Intelligent Systems, Springer International Publishing, Cham, December 2021.

1 Introduction

It is very common that any organisation, whether for profit, government or others, is defined as a system where value is created based on its business process model (BP). Indeed, the life cycle of these organisations is characterised by increasingly frequent phases of change due to the continuous search for competitiveness (Baresi et al., 2017).

Consequently, the task of managing the evolution of each organisation becomes a tedious task and requires a very rapid adaptation of their BPs in order to increase its agility and fluidity.

By definition, a BP represents a set of activities that are strongly correlated to achieve a well-determined organisational objective.

Thus, and despite the great desire of organisations to remain proactive, the monolithic nature of their process models (Ponce et al., 2016) places them before issues related to the performance of their services at the cost of the technical infrastructure and the cost of development and maintenance.

A monolith describes a single block of strongly connected components that complicates the process of adapting to functional, non-functional, and structural changes that organisations undergo. On the contrary, the micro-services architecture (Djogic et al., 2018; Baresi et al., 2017) has been invented since 2014 for the development of modern applications and with the aim of overcoming the shortcomings of these monolithic systems such as the lack of agility, the problem of scaling up and the strong interdependence of components.

This new architectural solution allows a monolith to be broken down into components that are thinner, highly cohesive and loosely coupled to scale easily to changing market demands (Chen et al., 2017; Indrasiri and Siriwardena, 2018).

So far, the micro-services identification 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 (Gysel et al., 2016; Levcovitz et al., 2016; Mazlami et al., 2017). 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 (2018), Daoud et al. (2020) and Saidi et al. (2021a, 2021b) are the only ones to have adopted BPs in this identification exercise.

The business process does not generally exist as singular entities, but rather as a family of variants that must be collectively managed (Rosa et al., 2017; Ayora et al., 2016; Cognini et al., 2018). As is the case, for example, for multinational organisations that use different process variants depending on the country or according to different regions of the same country. Similarly an insurance claims process may run differently between different claim processing centres.

Indeed, each variant may present differences in the sequence of activities, the decision rules, the data flows or the resources involved. These variants provide greater flexibility, better adaptation to changes and optimisation of resources, thus contributing to better efficiency and adaptability of the company.

For these reasons, we aim to propose in this paper a multi-model approach which aims to deal with the case of several variants of business processes in order to analyse control and data dependencies based on our previous work (Saidi et al., 2021a, 2021b) at first and to calculate the final dependency matrix based on our formulas proposed in a second step and finally generate the micro-services. To achieve our objectives, three main challenges have been addressed in our study related to existing micro-services architecture and solutions in practice:

1 *Challenge 1:* How can we identify micro-services from a set of process variants to meet specific needs or different contexts?

2 *Challenge 2:* How can we determine the appropriate level of granularity for each micro-service by modelling the control dependency linked to an independent set of BP variants?

3 *Challenge 3:* How can we determine a good strategy aimed at partitioning the data of a set of BP variants in order to reduce communication and simultaneous access to the same database while keeping a high cohesion between the generated micro-services?

The rest of this paper is organised 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 formalises the

control and data dependencies models. Section 4 presents the implementation of our proposed approach. Finally, we conclude with some future work.

2 Related work

The development of robust monolithic systems has reached its limits since 2014, because the implementation of changes in the current large, complex and rapidly evolving systems would be too slow and inefficient. Currently, there are a very large number of applications that migrate to a micro-services architecture.

In Chen et al. (2017), the authors proposed a top-down partitioning approach that is essentially based on a data stream in order to overcome the shortcomings of the initial system and migrate to a micro-services-based approach. As a first step, they propose to generate a data flow diagram (DFD) based on a natural language description. In the second step, they propose to transform the initial DFD into a purified DFD based on data operations. Finally, the purified DFD will be transformed into a decomposable DFD through a proposed algorithm to extract micro-services.

In Josélyne et al. (2018), the authors used domain-driven design (DDD) patterns to provide an approach for identifying appropriate micro-services. First, developers define a domain using a pervasive language. Domain experts determine the boundaries of each system responsibility and represent it as a business capability, where a business capability is something a system does to generate an output. Each business feature is a micro-service.

In Djogic et al. (2018), an approach for an architectural reconstruction of an integration platform that was initially based on an SOA architecture into a new micro-services-oriented platform. This new architecture solves the problem related to the large number of messages that must be processed as well as the number of new integrations that must be supported.

Romani et al. (2022) proposed a data-centric technique to determine candidate micro-services and migrate legacy software systems to a micro-services architecture. They used the World Web Dictionary as an illustrative example, elbow and k-means for the identification process.

Since enterprise developers are faced with the challenges of maintenance and scalability of increasingly complex projects. In Escobar et al. (2016), the authors proposed a model-based approach to analyse 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 Enterprise Java Beans (EJB) using the clustering technique.

In Sellami et al. (2022), the authors thought of an approach for identifying potential micro-services from the source code of a given application. This approach is essentially based on the measurement of similarity and dependence between the different classes of the system based on the interactions and the terminology of the domain used in the code. The authors used so a density-based algorithm to generate a hierarchical structure of recommended micro-services while identifying potential classes.

Al-Debagy et al. (2022) proposed an approach based on two essential steps; the first step is to represent the source code as a class dependency graph. The second step describes the graph clustering algorithm to identify micro-services. This approach was tested with 8 different applications and using 11 clustering algorithms to find the most accurate and efficient algorithm.

In Baresi et al. (2017), the authors described a semi-automatic approach for the micro-services identification exercise. Indeed, a Restful API is proposed in order to take as input a business model component which describes the flow of requirements with its inputs and outputs and the micro-services are subsequently generated by processing this model. This approach is based on measuring the semantic similarity of features described through Open API specifications. The authors based their approach on a multitude of specifications and they compared the result with the results of software engineers and the Service Cutter tool.

In Gysel et al. (2016), the authors presented a service decomposition tool based on 16 coupling criteria from industry and literature. They proposed a semi-automated model for identifying services according to predefined categories, based mainly on requirements artefacts. Each category was objectively detailed certain consistent criteria that the service had to present for its terminals through approximation algorithms and the use of weights.

Although the business process is a central and crucial element for the evolution and success of the company, only four works that took the business process as input in the exercise of discovering the appropriate micro-services. In Amiri (2018), the author proposed a micro-services identification method that decomposes a system using the clustering technique. To this end, they modelled a system as a set of business processes and they considered two aspects of structural dependency and data dependency.

The approach is essentially based on three steps: first, a TP relation that shows the structural dependency of activities within a business process. Next, a TD relation is defined to show the dependency of activities based on their used data objects. In the end, these two dependencies are aggregated into a single dependency matrix T. Amiri's approach can be extended to provide a method for calculating the dependency of a set of business processes that is independent.

Recently, Daoud et al. (2020) proposed to remove and deal with the limits of the approach of Amiri already mentioned in his work (Amiri, 2018). 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. Then they used a

collaborative clustering algorithm to generate the candidate micro-services.

In Saidi et al. (2021b), the authors proposed an extension of the control model presented in Daoud et al. (2020) They proposed four calculation formulas to calculate the dependence taking into account the case of $\text{loop}_{sequence}$, loop_{And}, loop_{Xor}, loop_{Or} in order to calculate the dependence matrix of control to subsequently generate the appropriate micro-services. In Saidi et al. (2021a), the authors 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. This matrix will be taken as input to the K-means algorithm to identify candidate micro-services.

The use of a process variants becomes a crucial issue for any modern organisation since it makes it possible to better adapt to changes and specific requirements of the environment.

Thus, these variants aim to optimise operations and reuse existing processes by adapting them in a modular way in order to save time and save resources when designing new business process variants.

Despite the benefits of using the BP variant approach, Amiri (2018) and Daoud et al. (2021) are the only ones who have used a set of BPs as input in the micro-service identification exercise.

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.

In fact, when it comes to identifying micro-services, clustering is used to group activities of a BP that are similar into distinct clusters or groups. This makes it possible to effectively split a system into autonomous and coherent micro-services. The clustering techniques, such as hierarchical clustering, k-means or density-based clustering, can be used to perform this task of generating potential micro-services.

Cheung (2003) described an approach which is based on global K-means algorithm. This incremental approach makes it possible to add one centre cluster through a deterministic search technique made up of N execution of the algorithm starting from the appropriate initial position.

In Likas et al. (2003), the authors presented a new generalisation of the a k-means algorithm and they made an analysis of the penalised mechanism and they showed its exceptional clustering performance through an experiment results.

It can therefore be concluded that the case of several BPs taken as input in order to identify the candidate micro-services was treated only in a single work. We aim so in this paper to propose a new method based on what has been proposed in Daoud et al. (2020) and Saidi et al. (2021a, 2021b) to generate micro-services from a set of business process variants.

This article has been greatly extended from previous work (Saidi et al., 2022) which proposed an approach to identify microservices from the independent variants of business processes.

3 Our approach for identifying micro-services

3.1 Our case study

In the film industry, image post-production is the process that begins when filming is completed and deals with the creative editing of the film. Figure 1 shows several independent image post-production processes. A process model is a graph composed of nodes of type activity, gateway and arcs which are based on these elements.

Activities capture the tasks performed in the process. Gateways are used to model alternate and parallel branches and merges. They can be of type OR or XOR (inclusive execution, exclusive execution) and AND.

The first process model BP1 in the figure presents three activities (a1: retrieve images; a2: prepare film for editing; and a3: finish editing film). The second model describes a sequence of four activities, the first of which is an activity shared with the first model (BP1). On the other hand, the last process describes a model which contains a connector of the exclusive gold type (Xor). This connector links between two sequences of activities. This BP shares with BP1 the three activities (retrieve images; prepare film for editing; and finish editing film) and with BP2, its shares (retrieve images, prepare editing on tape and finish tape editing).

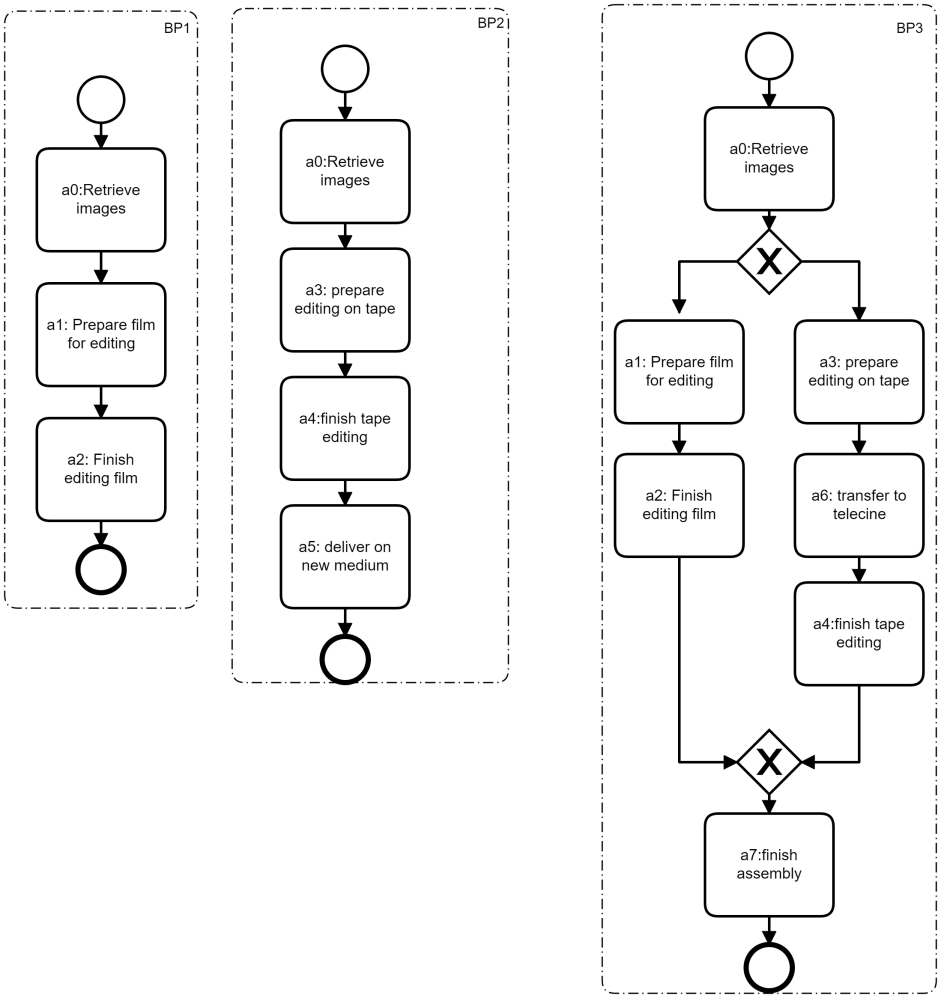
Our general idea is that from these three models of independent processes, but that they share common activities, we propose a method which is based on our two models control (by analysing the structural aspect of the process) and data (by analysing the correlation between the attributes of each pair of activities) to determine our candidate micro-services.

3.2 Foundations

It is well known that the business process represents the organised set of activities and software, material and human resources (Amiri, 2018), it is considered to be the central element and the backbone of each company. As a result, any economic success depends on the ability of its information system and its business process to easily integrate the changes imposed by the environment.

In this paper, we will be based on the business process as a monolithic system in order to break it down into appropriate micro-services using the dependency linked to a given couple of activities. These micro-services are fine-grained, loosely coupled and with strong cohesion.

Figure 1 An example of three independent BPs of the picture post-production process



We were able to identify four types of dependencies which are marked below.

- *Control dependency:* This dependency is essentially based on the order of execution (e.g., finish to start …) and the types of connectors that link the activities (xor, and …).

  Indeed, according to this model if a given pair of activities (ai, aj) is directly linked with a control dependency, it will probably form a highly cohesive micro-service. If not, these activities would form separate micro-services (Saidi et al., 2021b; Daoud et al., 2020).

  The identification of micro-services from a set of BPs taking into account the structural aspect of the model allows us to decouple specific functionalities from decisions and control mechanisms. This will facilitate the management, maintenance and scalability of our system, as well as it will promote reuse, modularity and flexibility, allowing isolation of responsibilities and finer management of workflows and control logics.

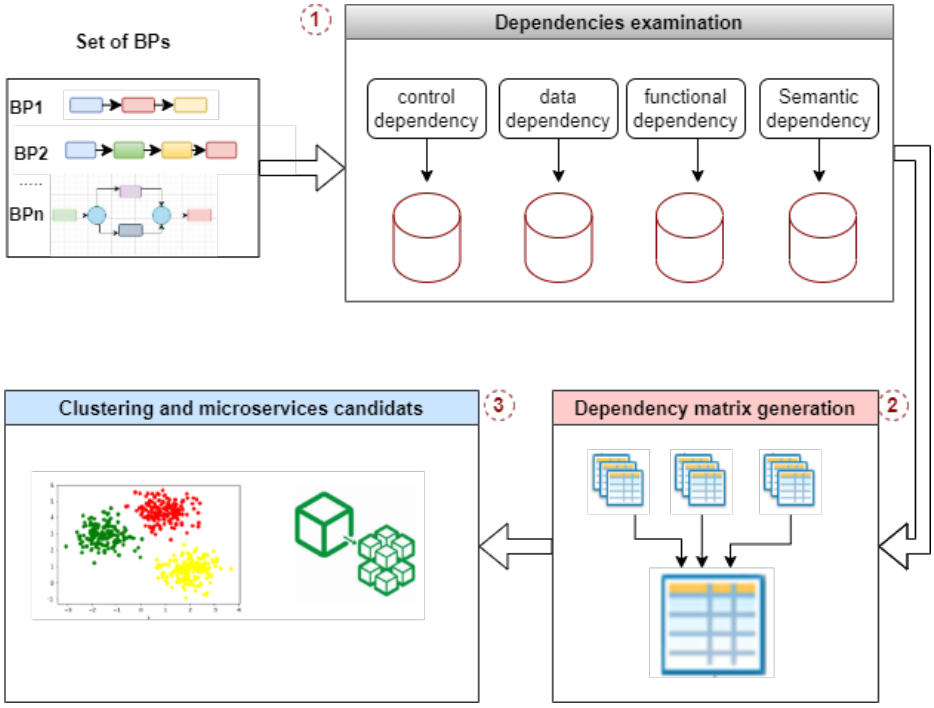
- *Data dependency:* From this model, we aim to represent the activities and attributes of our BP as a binary representation (if the attribute is present in the activity, we assign the value 1 if not 0). This representation will be our system input and from which we will use the association analysis method to discover hidden relationships between the attributes of a given activity pair.

  Subsequently, if these activities share the same attributes, it is very likely that they will be classified in the same micro-service. If not, they will be classified in separate micro-services (Saidi et al., 2021a).

  This dimension consists of analysing the data used and manipulated by the system in order to identify micro-services that have a high cohesion in terms of processing this data.

  The generation of micro-services from this dimension can be combined with other approaches such as the structural analysis of a BP in order to determine a more complete and precise breakdown of the system into micro-services.

Figure 2 Our proposed architecture (see online version for colours)



- *Functional dependency:* This method aims to break down the global domain into sub-domains based on the technique of DDD (domain-driven-design) so that each identified sub-domain will be considered as a micro-service. The really general idea of this approach is to stay consistent when designing micro-services with bounded context (BC). Indeed, any domain (knowing that a given BP represents a domain) is composed of several bounded contexts where each one encapsulates the associated functions in the model.

  Micro-services represent, therefore these bounded contexts being a good indication on a weak coupling with the strong cohesion of micro-services.

- *Semantic dependency:* From this dimension, we aim to identify micro-services by taking into account the semantic similarity of the names of each activity in order to group those that fall within the same application domain together.

In this work we are only interested in control and data dependency to identify appropriate micro-services from a set of business process models.

3.3 *The main steps of our approach*

Through Figure 2, we identify three essential steps in our proposed architecture.

- *Dependencies examination:* Through this step we will analyse the specifications of each process taken as system input and we will determine the dependence, according to two dimensions: control and association rules.

  For each business process, we will determine its own control dependency matrix analysing the different types of connectors in the model and then applying the appropriate formulas. In the same way, we extract the matrices of each business process according to the second type of dependence and by analysing the model in terms of correlation between shared attributes of each activity.

  Indeed, for n process, we will have 2n dependency matrices.

- *Dependency matrix generation:* In this step, we will be based on the matrices generated in the previous step in order to calculate the final dependence matrix. To do this, formulas have been proposed for the aggregation of these matrices. This part will be described in details below.

- *Micro-services candidate generation:* In this last step, we will be based on three clustering algorithm which takes as input the generated dependency matrix to identify the candidate micro-services.

  Each cluster will contain activities that form a micro-service candidate.

3.4 Micro-services identification

3.4.1 Analysing dependencies

- *Control dependency formula*

  The control dimension of a BP refers to the way activities are organised and interconnected. It allows us to decouple the different logical parts of a system into autonomous and independent services. This gives us increased modularity, better code reuse and easier system scalability.

  To do so, we will calculate the dependency matrix for each BP model separately by using the formulas given in Saidi et al. (2021b) and Daoud et al. (2020) and then we add up to generate a single output matrix.

  Note: If there is not an arc that links between a couple of given activities, we will assign the value ”0”.

  $\text{Dep}_c = \sum_{i=1}^{n} (M1(ai, aj), M2(ai, aj), ..Mn(ai, aj))$

  If we take the case of our example already represented in figure 1 and we try to analyse the control dependency.

  a *Control dependency matrix 1:* We calculated in this matrix (Table 1) the control dependency of the first BP.

Table 1 Control dependency matrix 1

	a0	*a1*	*a2*
a0	-	1/2	1/4
a1	1/2	-	1/2
a2	1/4	1/2	-

  b *Control dependency matrix 2:* We calculated in this matrix (Table 2) the control dependence of second BP.

Table 2 Control dependency matrix 2

	a0	*a3*	*a4*	*a5*
a0	-	1/2	1/4	1/8
a3	1/2	-	1/2	1/4
a4	1/4	1/2	-	1/2
a5	1/8	1/4	1/2	-

  c *Control dependency matrix 3:* We calculated in this matrix (Table 3) the control dependency of the third BP.

Table 3 Control dependency matrix 3

	a1	*a2*	*a3*	*a4*	*a5*	*a7*	*a8*
a1	-	1/4	1/8	1/4	1/16	1/8	1/32
a2	1/4	-	1/2	0	0	0	1/8
a3	1/8	1/2	-	0	0	0	1/4
a4	1/4	0	0	-	1/4	1/2	1/16
a5	1/16	0	0	1/4	-	1/2	1/4
a7	1/8	0	0	1/2	1/2	-	1/8
a8	1/32	1/8	1/4	1/16	1/4	1/18	-

  Our global control dependency matrix will be so as described in Table 4.

Table 4 Global control dependency matrix

	a0	*a1*	*a2*	*a3*	*a4*	*a5*	*a6*	*a7*
a0	-	1/2	1/8	1/2	1/4	1/8	1/8	1/32
a1	1/2	-	1/2	0	0	0	0	1/8
a2	1/8	1/2	-	0	0	0	0	1/4
a3	1/2	0	0	-	1/2	1/4	1/2	1/16
a4	1/4	0	0	1/2	-	1/2	1/2	1/4
a5	1/8	0	0	1/4	1/2	-	0	0
a6	1/8	0	0	1/2	1/2	0	-	1/8
a7	1/32	1/8	1/4	1/16	1/4	0	1/8	-

- *Association rules formula*

  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 Saidi et al. (2021a).

By analysing the three BPs in terms of data, the dependency matrices is as shown in Figure 3.

Figure 3 Data dependency matrix (see online version for colours)

```
> python dependecy_matrix.py --min-supp 0.5 --min-conf 0.7
       a_0    a_1    a_2    a_3    a_4    a_5    a_6    a_7
a_0   0.00   4.44   6.57  38.03  20.05  37.18  44.43  38.03
a_1   4.44   0.00   6.57  38.03  44.43  37.18  44.43  38.03
a_2   6.57   6.57   0.00  38.03  22.18  44.43  44.43  38.03
a_3  38.03  38.03  38.03   0.00  44.43  44.43  44.43  44.43
a_4  20.05  44.43  22.18  44.43   0.00  44.43  44.43  44.43
a_5  37.18  37.18  44.43  44.43  44.43   0.00  44.43  44.43
a_6  44.43  44.43  44.43  44.43  44.43  44.43   0.00  44.43
a_7  38.03  38.03  38.03  44.43  44.43  44.43  44.43   0.00
```

To generate this matrix described in Figure 3, we used the apriori algorithm implemented in Saidi et al. (2021a).

Indeed, we set the minimum support value to 0.5 and the minimum confidence value to 0.7 in order to generate the set of association rules that will be used later by the dependency calculation algorithm (Saidi et al., 2021b).

3.4.2 Micro-services identification

The sum of the two previously generated matrices (data and control) will be defined by performing the aggregation of the corresponding elements of each matrix. The aggregation is carried out by respecting the positions of the elements in the two matrices. To reformulate all this mathematically, we will use the element-by-element addition operation since our two matrices are two of the same size. Indeed, their sum creates a new matrix GM of the same dimension where each element is calculated as follows:

$$\text{gm}[i]\ [j] = Sum(Dep_c\,[i]\ [j],\ Dep_D\,[i]\ [j])$$

where $Dep_c[i]\ [j]$ is the element at position (i, j) of the control matrix, $Dep_D[i]\ [j]$ is the element at position (i, j) of the data matrix and $\text{gm}(i, j)$ is the element corresponding to position (i, j) in the global matrix GM.

Table 5 Final dependency matrix

	a0	*a1*	*a2*	*a3*	*a4*	*a5*	*a6*	*a7*
a0	0	4,94	6,695	38,53	20,3	37,305	44,555	38,061
a1	4,94	0	7,07	38,03	44,43	37,18	44,43	38,155
a2	6,695	7,07	0	38,03	22,18	44,43	44,43	38,28
a3	38,53	38,03	38,03	0	44,93	44,68	44,93	44,492
a4	20,3	44,43	22,18	44,93	0	44,93	44,93	44,68
a5	37,305	37,18	44,43	44,68	44,93	0	44,43	44,43
a6	44,555	44,43	44,43	44,93	44,93	44,43	0	44,555
a7	38,061	38,155	38,28	44,492	44,68	44,43	44,555	0

After generating our final matrix (Table 5), we will use three different clustering algorithms (partitional clustering, hierarchical clustering and distribution-based clustering) to generate our candidate micro-services. Indeed, an example of micro-services generation using K-means, agglomerative algorithm and GMM is summarised through Table 6 which each cluster describes a candidate micro-services.

Table 6 Micro-services identification

Clustering algorithm	*Clusters*
Partitional clustering	cluster 1 (a0, a6)
	cluster 2 (a4, a5, a7)
	cluster 3 (a1)
	cluster 4 (a2, a3)
Hierarchical clustering	cluster 1 (a0, a4, a6, a7)
	cluster 2 (a1, a2, a3, a5)
Distribution-based clustering	cluster 1 (a4, a7, a5)
	cluster 2 (a2, a3)
	cluster 3 (a1)
	cluster 4 (a0, a6)

4 Experimentation

4.1 Experiment protocol

4.1.1 Method's description

The method used consists of designing a micro-services-oriented architecture based on a set of business process variants. Our proposed architecture is be based on two identification strategies: structural and data. Indeed, it is composed of five essential modules as it is modelled on Figure 4.

- *Camunda modeller:* In order to model our system input, we will use Camunda which is designed specifically for the BPMN workflow. This tool actually allows us to generate two outputs. The first in the form of a graphical representation on which we will be based in the third module to describe the attributes linked to each activity which are necessary for the calculation of data dependency.

  From an XML file generated by this tool, we will apply the algorithm proposed in Daoud et al. (2020) and Saidi et al. (2021b) in order to generate the control dependency matrix.

- *Control dependency module:* This module therefore takes as input the XML file generated by Camunda and gives as output the control dependency matrix.

  Indeed, according to the analysis of the type of connector existing between an activities' couple (ai, aj), we will apply the formula which is appropriate to the case (sequence, Xor, or, and, loop …). We will reuse so the formulas proposed in Saidi et al. (2021b) and Daoud et al. (2020) to calculate this dependency.

- *Association rules matrices generator:* This module takes as input the graphical representation of BP and gives as output a data dependency matrix.

  Indeed, based on the method proposed in Saidi et al. (2021a), we will apply the proposed algorithm to calculate the interesting correlations between the different attributes of the three proposed business process and thereafter the activities will share data will be classified together.

- *Global matrix generator:* For n BP models, we will have n matrices at the level of the first dimension (control dependency) and the second dimension, which is based on the identification of strong and weak associations between activities. For this reason, we proposed to make an aggregation of n matrices generated for the first dimension and suddenly we will have as output a single matrix instead of several. For the second dimension, if a given pair of activities (ai, aj) is the same in the other variants of BP, then it will be the same dependency value, if not, this value is recalculated by applying the method already proposed in Saidi et al. (2021a).

As output, we will have two matrices. In order to generate our final dependency matrix, we proposed to take the '*Sum*' of the two dependencies calculated for a couple of activities (ai, aj).

Algorithm 1 Global dependency matrix

Input: $Dep_c\,[i]\,[j]$, $Dep_D\,[i]\,[j]$
Output: $GM\,[i]\,[j]$
begin
  if $n==m$ **then**
    for $i=1$ *to* n **do**
      for $j=1$ *to* m **do**
        GM$[i]\,[j] \leftarrow Dep_c\,[i]\,[j]+Dep_D\,[i]\,[j]$
  return GM
end

- *Micro-services identification module:* This module is based on the global matrix calculated in the previous module and using three different clustering algorithms (k-means, agglomerative and GMM) in order to identify our micro-services which are fine-grained and with low coupling and high cohesion.

  Table 6 summarises the classification results of the different clustering methods that we used.

4.1.2 Experimental plan and data

The experiment will be conducted by determining the set of clusters where each cluster contains the set of activities that will be executed together using our three clustering algorithms to guarantee fine granularity, weak coupling and high cohesion of micro-services.

Our first control dependency dimension is implemented in Java and the second data-oriented dimension will be implemented in Python.

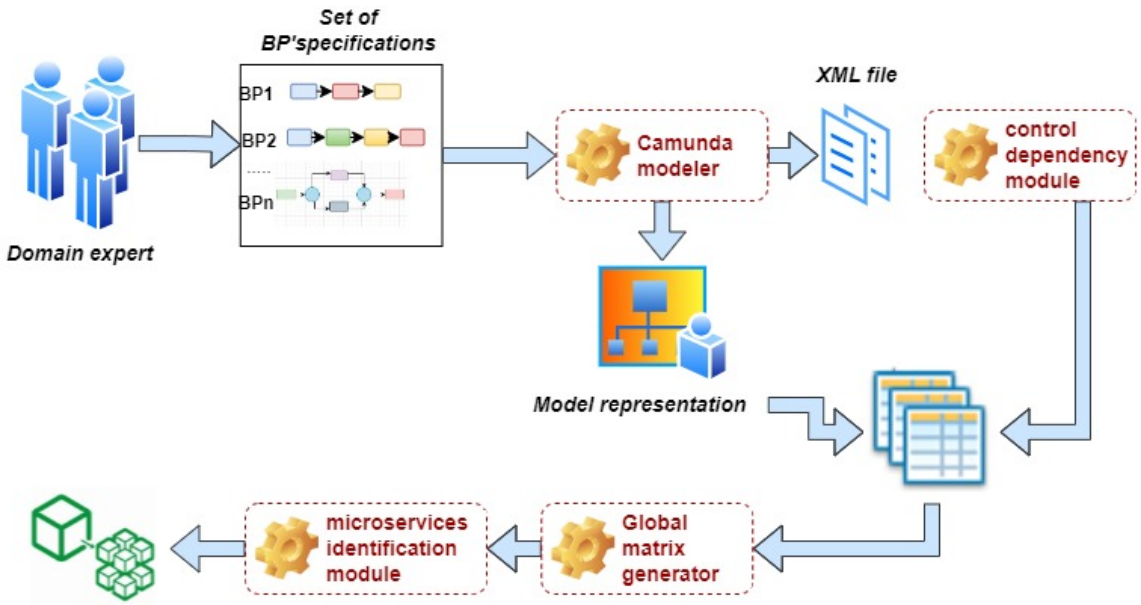
Regarding our dataset, it will be formed by three different process variants as described previously (Subsection 3.1).

In fact, the first BP is composed of a sequence of three activities (a0: retrieve images, a1: prepare film for editing, a2: finish editing film). The second is composed of a sequence of four activities (a0: retrieve images, a3: prepare editing on tape, a4: finish tape editing, a5: deliver on new medium). This variant shares activity a0 with the first BP. On the other hand, the third BP is made up of seven activities (a0: retrieve images, a1: prepare film for editing, a2: finish editing film, a6: transfer to telecine, a4: finish tape editing and a7: finish assembly) including a0, a1, a4 are activities in common with the two previous BP variants.

4.1.3 Validation and evaluation

The evaluation of the clustering results can be carried out using different criteria to measure the quality of the partitions obtained. In this context, we chose to use three different clustering algorithms (K-means, agglomerative and GMM) and to evaluate the results obtained by each algorithm. To do this, several metrics can be considered such as the silhouette index, which measures how similar the instances of the same cluster are to each other compared to the instances of other clusters and we will use Dunn's index, which evaluates the minimum distance between clusters and the maximum distance within clusters. By combining these different metrics and evaluation approaches, it will be possible to compare and evaluate the performance of the three clustering algorithms used (more details in the next section).

Figure 4 Micro-services identification architecture (see online version for colours)



4.2 Experiments

Clustering is a set of techniques used to divide data into groups or clusters.

In our work, we consider an activity of each BP as a distinct object. The activities that belong to the same cluster are supposed to be as homogeneous as possible to ensure the cohesion property of a group. On the other hand, activities belonging to different groups are supposed to be as distinct as possible to define a loose coupling of a group. Each cluster could be a candidate micro-service.

In our evaluation, we will use three clustering algorithms: partitional, hierarchical and distribution-based algorithms.

- *Partitional clustering:* Divides data objects into disjoint groups.

  K-means tries to minimise the intra-cluster inertia which measures the cohesion of the points inside each cluster.

  Our objective through this algorithm will therefore be to find the centroids which minimise the sum of the squares of distance between the points and their respective centroids.

  It is important to mention that this algorithm may converge to a local minimum, which means that it may not find the best global solution.

  So that we can solve this problem, we can run our algorithm several times with different random initialisations and then select the best solution in terms of intra-cluster inertia. With the result of the implementation of the K-means algorithm, we have identified four different clusters.

- *Hierarchical clustering:* Determines cluster assignments by creating a hierarchy. We chose in our case the agglomerative clustering. With the agglomerative algorithm, the number of clusters is generally not fixed in advance, but is determined by the algorithm itself according to the grouping criteria used.

  However, there are widely used techniques for determining the number of clusters in an agglomerative algorithm such as the elbow method. We used this method to run our algorithm for different numbers of clusters and then compute the intra-cluster dissimilarity each time.

  Thereafter we draw the curve according to the number of clusters and we seek the point where the addition of additional clusters no longer brings significant improvement. With the result of the implementation of the agglomerative algorithm, we have identified two clusters.

  Elbow method will not necessarily provide us with the exact and optimal number of clusters, but it provides indications on the appropriate number of clusters according to our data and our grouping criteria used.

- *Distribution-based clustering:* Gaussian mixture models (GMMs) assume that there are a number of Gaussian distributions, and each of these distributions represents a cluster. Therefore, the Gaussian model mixture algorithm begins with the initialisation of the model parameters. This involves specifying the number of clusters and estimating initial parameters, such as means, component weights, etc.). Once the model parameters have converged, data points are assigned to clusters based on their maximum membership probability. Each activity will be carried out in the cluster that best corresponds to it in terms of similarity according to the Gaussian distribution of the model.

The silhouette measurement is used in our evaluation as an internal validation method to assess the quality of the scores obtained. By comparing silhouette values between algorithms, one can get an indication of which produces more consistent and distinct partitions. This can help choose the most appropriate algorithm for a specific dataset.

Figure 5 Silhouette score in Python (see online version for colours)

```
Kmeanspl=metrics.silhouette_score(transformed, k_means.labels_,
metric='sqeuclidean')
Agglomerativepl=metrics.silhouette_score(transformed, model.labels_,
metric='sqeuclidean')
gmpl=metrics.silhouette_score(transformed, gm_labels, metric='sqeuclidean')
```

Indeed, the silhouette score is a measure used to calculate the quality of a clustering technique. Its value is between –1 and 1. Where a high value indicates that the object is well-classified to its own cluster and poorly suited to neighbouring clusters. If most of the objects have a high value, the clustering configuration is adequate.

- 1: means that the clusters are well separated from each other and clearly distinguished
- 0: means that the clusters are insubstantial, or we can say that the distance between the clusters is not significant
- –1: means that the clusters are assigned in the wrong direction.

By using this method (*silhouette score*) of interpretation and validation of consistency within data clusters, we were able to obtain a small comparison (Figures 6 and 7) on the quality of the classification of each clustering algorithm used.

Indeed, through the analysis of the results obtained, we can deduce that in our case the agglomerative algorithm generates so better results compared to the other two algorithms.

Also, by making a comparison in terms of execution time between the three algorithms and by varying the number of clusters (figure 8), we see that the K-means

algorithm has the highest value when the number of clusters is equal to 2, 3 or 4 on the other hand the agglomerative algorithm was executed in a minimum period of time.

Figure 6 Comparison's results

```
Silhouette Coefficient Kmeans: 0.229
Silhouette Coefficient Agglomerative: 0.347
Silhouette Coefficient GaussianMixture: 0.229
```

Figure 7 Qualitative comparison (see online version for colours)

Figure 8 Comparison in terms of execution time (see online version for colours)

We used the Dunn index validation metric which will allow us to measure the quality of micro-services.

A higher Dunn's index indicates that the instances in each cluster are tightly packed together, while the clusters are well separated from each other. This makes it easier to communicate the results to the stakeholders and to make a decision regarding the choice of the best partition.

In Figures 9, 10 and 11, we measured the Dunn's index of the clustering algorithm using matrices of control dependencies, data and the aggregation of the two dimensions together (the cooperative case). The results obtained in the three experiments clearly showing that that the Dunn's index in the case of the data dimension is then almost always better than the control Dunn's index. This means that for a given BP, the data dependency model is richer and more informative than the control dependency model.

Figure 9 Dunn index with K-means (see online version for colours)

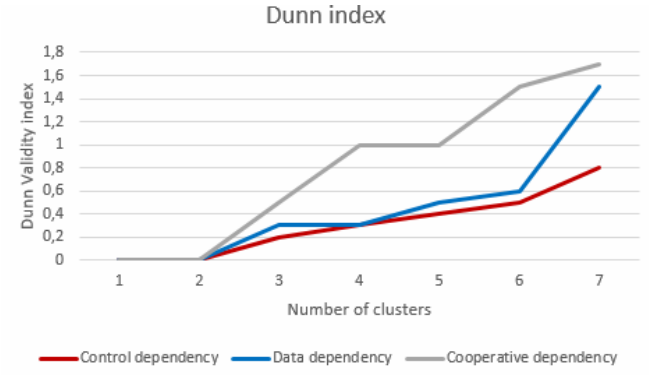


Figure 10 Dunn index with GMM (see online version for colours)

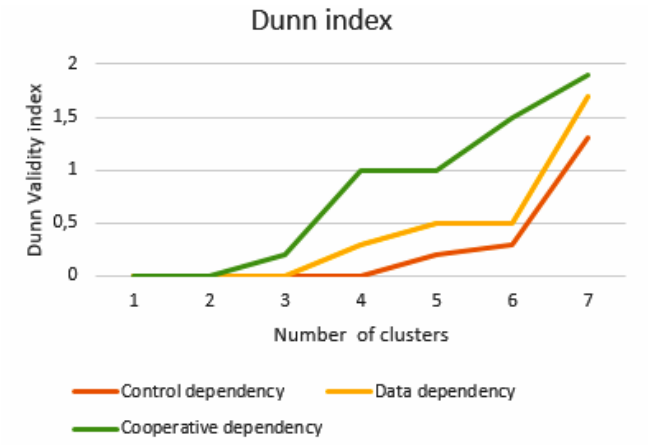
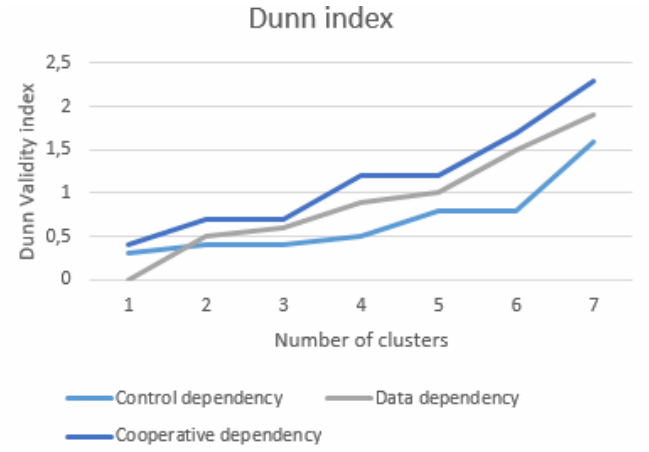


Figure 11 Dunn index with agglomerative technique (see online version for colours)



Indeed, the quality of the final generation of micro-services is often much better by combining the two dimensions together (control and data).

By analysing the results obtained by the three algorithms, we see that for the quality of a partition in terms

of separation between the clusters and the compactness of the clusters, the agglomerative algorithm gives better results compared to those generated by K-means and GMM.

5 Conclusions

Any organisation, whether non-profit, governmental or private, has been designed as a system where value is created through its business process model.

Indeed, the difference between a simple business process model and a set of variants of it lies in the ability to take into account the specific variations and conditions that can arise in the operational environment of a company. These variants offer greater flexibility and better adaptation to changes, thus contributing to greater efficiency and better adaptability of the company.

However, these process variants describe a monolithic system with components that are tightly coupled.

For these reasons, we have proposed an approach to decompose this system into appropriate micro-services based on the two dimensions control and data and using three different clustering algorithms.

As future work, we aim to improve our evaluation part by making a comparison between the results of the final micro-services generated using what we proposed in this approach (aggregation of models for a set of BPs) and the collaboration method (using collaborative algorithms) and to treat the dependency between a couple of activities in a configurable process model.

References

Al-Debagy, O. and Martinek, P. (2022) 'Dependencies-based microservices decomposition method', *International Journal of Computers and Applications*, Vol. 44, No. 9, pp.814–821.

Amiri, M.J. (2018) 'Object-aware identification of microservices', *2018 IEEE International Conference on Services Computing (SCC)*, July, IEEE, pp.253–256.

Ayora, C., Torres, V., de la Vara, J.L. and Pelechano, V. (2016) 'Variability management in process families through change patterns', *Information and Software Technology*, Vol. 74, pp.86–104.

Baresi, L., Garriga, M. and De Renzis, A. (2017) 'Microservices identification through interface analysis', *Service-Oriented and Cloud Computing: 6th IFIP WG 2.14 European Conference, ESOCC 2017, Proceedings*, 27–29 September, Springer International Publishing, Oslo, Norway, Vol. 6, pp.19–33.

Chen, R., Li, S. and Li, Z. (2017) 'From monolith to microservices: a dataflow-driven approach', *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*, IEEE, December, pp.466–475.

Cheung, Y.M. (2003) 'k-Means: a new generalized k-means clustering algorithm', *Pattern Recognition Letters*, Vol. 24, No. 15, pp.2883–2893.

Cognini, R., Corradini, F., Gnesi, S., Polini, A. and Re, B. (2018) 'Business process flexibility – a systematic literature review with a software systems perspective', *Information Systems Frontiers*, Vol. 20, pp.343–371.

Daoud, M., El Mezouari, A., Faci, N., Benslimane, D., Maamar, Z. and El Fazziki, A. (2021) 'A multi-model based microservices identification approach', *Journal of Systems Architecture*, Vol. 118, p.102200.

Daoud, M., Mezouari, A. E., Faci, N., Benslimane, D., Maamar, Z. and Fazziki, A.E. (2020) 'Automatic microservices identification from a set of business processes', *Smart Applications and Data Analysis: Third International Conference, SADASC 2020, Proceedings*, 25–26 June, Springer International Publishing, Marrakesh, Morocco, Vol. 3, pp.299–315.

De Alwis, A.A.C., Barros, A., Polyvyanyy, A. and Fidge, C. (2018) 'Function-splitting heuristics for discovery of microservices in enterprise systems', *Service-Oriented Computing: 16th International Conference, ICSOC 2018, Proceedings*, 12–15 November, Springer International Publishing, Hangzhou, China, Vol. 16, pp.37–53.

Di Francesco, P., Lago, P. and Malavolta, I. (2018) 'Migrating towards microservice architectures: an industrial survey', *2018 IEEE International Conference on Software Architecture (ICSA)*, April, IEEE, p.2909.

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)*, May, IEEE, pp.1411–1414.

Escobar, D., Cárdenas, D., Amarillo, R., Castro, E., Garcés, K., Parra, C. and Casallas, R. (2016) 'Towards the understanding and evolution of monolithic applications as microservices', *2016 XLII Latin American Computing Conference (CLEI)*, October, IEEE, pp.1–11.

Estañol, M. (2016) *Artifact-centric Business Process Models in UML: Specification and Reasoning*, Montserrat Estañol supervised by Professor Ernest Teniente Universitat Politècnica de Catalunya, Barcelona, Spain.

Gysel, M., Kölbener, L., Giersche, W. and Zimmermann, O. (2016) 'Service cutter: a systematic approach to service decomposition', *Service-Oriented and Cloud Computing: 5th IFIP WG 2.14 European Conference, ESOCC 2016, Proceedings*, 5–7 September, Springer International Publishing, Vienna, Austria, Vol. 5, pp.185–200.

Indrasiri, K. and Siriwardena, P. (2018) *Microservices for the Enterprise*, pp.143–148, Apress, Berkeley.

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

Levcovitz, A., Terra, R. and Valente, M.T. (2016) *Towards a Technique for Extracting Microservices from Monolithic Enterprise Systems*, arXiv preprint arXiv:1605.03175.

Likas, A., Vlassis, N. and Verbeek, J.J. (2003) 'The global k-means clustering algorithm', *Pattern Recognition*, Vol. 36, No. 2, pp.451–461.

Mazlami, G., Cito, J. and Leitner, P. (2017) 'Extraction of microservices from monolithic software architectures', *2017 IEEE International Conference on Web Services (ICWS)*, June, IEEE, pp.524–531.

Ponce, F., Márquez, G. and Astudillo, H. (2019) 'Migrating from monolithic architecture to microservices: a rapid review', *2019 38th International Conference of the Chilean Computer Science Society (SCCC)*, November, IEEE, pp.1–7.

Richards, M. (2015) *Software Architecture Patterns*, Vol. 4, O'Reilly Media, Incorporated, No. 1005 Gravenstein Highway North, Sebastopol, CA – 95472, USA.

Romani, Y., Tibermacine, O. and Tibermacine, C. (2022) 'Towards migrating legacy software systems to microservice-based architectures: a data-centric process for microservice identification', *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*, March, IEEE, pp.15–19.

Rosa, M.L., Aalst, W.M.V.D., Dumas, M. and Milani, F.P. (2017) 'Business process variability modeling: a survey', *ACM Computing Surveys (CSUR)*, Vol. 50, No. 1, pp.1–45.

Saidi, M., Daoud, M., Tissaoui, A., Sabri, A., Benslimane, D. and Faiz, S. (2021a) 'Automatic microservices identification from association rules of business process', *International Conference on Intelligent Systems Design and Applications*, December, Springer International Publishing, Cham, pp.476–487.

Saidi, M., Tissaoui, A., Benslimane, D. and Faiz, S. (2021b) 'Automatic microservices identification across structural dependency', *International Conference on Hybrid Intelligent Systems*, December, Springer International Publishing, Cham, pp.386–395.

Saidi, M., Tissaoui, A. and Faiz, S. (2022) 'From a monolith to a microservices architecture based dependencies', *International Conference on Intelligent Systems Design and Applications*, December, Springer Nature Switzerland, Cham, pp.34–44.

Sellami, K., Saied, M.A. and Ouni, A. (2022) 'A hierarchical dbscan method for extracting microservices from monolithic applications', *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering*, June, pp.201–210.

Tyszberowicz, S., Heinrich, R., Liu, B. and Liu, Z. (2018) 'Identifying microservices using functional decomposition', *Dependable Software Engineering. Theories, Tools, and Applications: 4th International Symposium, SETTA 2018, Proceedings*, 4–6 September, Springer International Publishing, Beijing, China, Vol. 4, pp.50–65.