Object-aware Identification of Microservices

Mohammad Javad Amiri
Department of Computer Science
University of California at Santa Barbara
Email: amiri@cs.ucsb.edu

***Abstract*—Microservices is an architectural style inspired by service-oriented computing that structures an application as a collection of cohesive and loosely coupled components, which implement business capabilities. One of today's problems in designing microservice architectures is to decompose a system into cohesive, loosely coupled, and fine-grained microservices. Identification of microservices is usually performed intuitively, based on the experience of the system designers, however, if the functionalities of a system are highly interconnected, it is a challenging task to decompose the system into appropriate microservices. To tackle this challenge, we present a microservice identification method that decomposes a system using clustering technique. To this end, we model a system as a set of business processes and take two aspects of structural dependency and data object dependency of functionalities into account. Furthermore, we conduct a study to evaluate the effect of process characteristics on the accuracy of identification approaches.**

***Index Terms*—Microservice; Identification; Business Process; Clustering**

I. Introduction

Microservice architecture is a style that is increasingly gaining popularity, both in academia and in the industrial world [1] by trying to overcome the shortcomings of centralized, monolithic architectures [2]. It can be defined as an approach for developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms [3]. Microservices architecture forms a software system as a group of fine-grained, cohesive, and loosely coupled services where each service implements a small business capability. The architecture is built on three simple principles [4]: (1) *Bounded Context*: focus on business capabilities. Related functionalities are combined into a single business capability which is then implemented as a service. (2) *Size*: If a service is too large, it should be refined into two or more services, thus preserving granularity and maintaining focus on providing only a single business capability. (3) *Independence*: This concept encourages loose coupling and high cohesion by stating that each service in microservice architectures is operationally independent of others.

Partitioning the system into microservices is usually performed intuitively, based on the experience of the system designers, however, if the functionalities of a system are highly interconnected, it is a challenging task to decompose the system into appropriate microservices [5][6], this problem is known as microservice identification. Although the problem has recently been studied, the existing methods [7][8][9] do not consider the underlying business functionalities of the system. This paper studies the problem of identifying microservices from the business processes point of view. A related but different problems exist in SOA domain where researchers use different techniques to discover [10], identify [11], and compose [12] services.

This paper identifies microservices from business processes. A business process consists of a set of activities performed in coordination in an organizational environment to accomplish a business goal [13]. Each activity in a business process plays the role of an operation in a microservice. The goal is to decompose business processes into fine-grained, cohesive, and loosely coupled components which are called microservices.

A business process can be partitioned by considering the structural relations between activities within the process: if there is a direct edge between two activities, these two activities are more likely to partition into the same microservice. Each activity also performs read and write operations on some data objects to execute where these data objects are shared among different activities. Since we want the resulting partitions (microservices) to be independent and loosely coupled, activities with similar data access should be partitioned into the same microservice.

Technically, we consider BPMN business processes with data object reads and writes. We then define two relations to show structural and object dependencies between activities, and aggregate these two relations. The final relation is clustered and each resulted cluster is identified as a microservice.

A key contribution of this paper is to show that it is possible to use underlying business processes of a system for microservice identification. Technically, this paper makes the following contributions: (1) Notions of "structural dependency" and "object dependency" between activities are defined, (2) Three approaches to identify microservices from a business process are introduced, (3) An Extended approach to identify microservices from a set of business processes is presented, and (4) The accuracy of different microservice identification approaches, i.e, user-driven, process-driven, object-driven, and the proposed method, for different processes is studied.

The remainder of the paper is organized as follows. Section 2 illustrates the problem with a concrete example. Section 3 defines a model for business processes. Section 4 introduces our approach to identify microservices. Section 5 performs a study on accuracy of the identification approaches, and Section 6 concludes the paper.

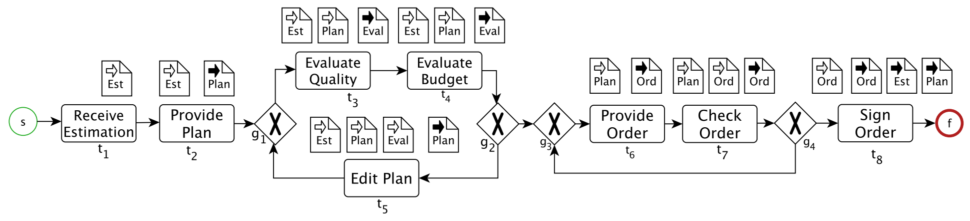


Fig. 1: The Process `PlanApproval` (`Est`: Estimation, `Eval`: Evaluation, `Ord`: Order)

II. Motivations

Fig. 1 shows a BPMN process model for a `PlanApproval` process in some part supplier company where the company needs yearly estimations for purchasing required parts. Let's assume the company wants to develop a system for its business and the developers are considering microservices as their software architecture. Here the problem is to identify appropriate microservices by decomposing each business process in a way such that the identified microservices are cohesive, loosely coupled, and fine-grained. In other words, each activity of a business process plays the role of an operation (functionality) in one microservice and we want to cluster all the activities of a business process. Each cluster is then introduced as a microservice.

To find such microservices, one may take the structural dependency of activities into account and cluster activities based on the edges in the process model: if there is a direct edge between two activities within a business process, those two activities are more likely two be clustered in the same microservice. Fig. 2(a) shows the microservices identified from the `PlanApproval` process using this approach where the process decompose into 4 microservices.

Another approach is to consider data object dependencies and put activities in the same cluster if they access to a similar set of data objects. We use object-driven approach to identify microservices of the `PlanApproval` process. The resulting microservices can be seen in Fig. 2(b).

Now the question is which one is the most appropriate approach to identify cohesive, loosely coupled, and fine-grained microservices? In the following sections, we try to propose a method to identify such microservices.

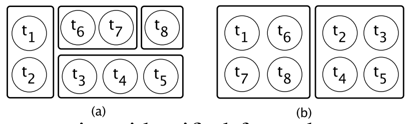


Fig. 2: Microservices identified from the `PlanApproval`

III. A Model for Business Processes

In this section, we introduce a model for business processes. Each business process comprises a set of tasks, gateways, events, and connectors. In BPMN [14], a process is modeled as a graph whose nodes and edges are of different types. In this paper, we focus on one type of edges corresponding to *sequence flow* in BPMN, and three types of nodes: *event*, *activity*, and *gateway*. Process schemas here are also BPMN processes in [15].

An *activity* node represents a unit of work. Each activity reads a set of objects and updates another set of objects. We consider two special events in BPMN: *start* and *end* events that signal the start and the end of a process respectively, and four kinds of frequently used gateways in BPMN: choice, merge, split, and join which are used to control the divergence and convergence of sequence flows.

Definition: A *process schema* is a tuple $P = (N, s, f, F, O, \rho, \omega)$ where N is a finite set of nodes, s and f are the *start* and the *end* nodes respectively, $F \subseteq N \times N$ is a set of *flow* edges, O is a finite set of objects identifier, and ρ and ω are two *data* mappings that assign each activity node a set of objects that are read or written by the node respectively.

Example: Consider the `Plan Approval` process (Fig. 1). Nodes s and f are the start and end nodes, t_i's ($1 \leqslant i \leqslant 8$) and g_j's ($1 \leqslant j \leqslant 4$) are activity and gateway nodes in N where g_1 and g_3 are merge gateways and g_2 and g_4 are choice gateways. `Est`, `Eval`, `Plan`, and `Ord` are the objects in O, and functions ρ, ω are used to assign objects to activity nodes. For example: $\rho(t_7)$={`Plan,Ord`}, and $\omega(t_8)$={`Ord,Est,Plan`}.

IV. Microservice Identification

In this section, a method to identify microservices from business processes is proposed. For each pair of activities in a process model we define two relations regarding their structural and data dependencies and then by aggregating these relations, the final relation between activities is defined. We then use that relation to cluster activities and identify microservices.

First, a relation T_P that shows the structural dependency of activities within a business process is defined. If there is direct edge between two activities or there is path between them containing only gateways, then those two activities are inter-connected.

Definition: Given a schema $P = (N, s, f, F, O, \rho, \omega)$, for each pair of activities $a_i, a_j \in N$, if $(a_i, a_j) \in F$ or there is a path $(a_i, n_1, ..., n_p, a_j)$ in P such that $\forall k \in [1...p] : n_k$ is a gateway, then $T_P(a_i, a_j) = 1$, else $T_P(a_i, a_j) = 0$.

Then, a relation T_D is defined to show the dependency of activities based on their used data objects where for each pair of activities, we assign weight 1 to the objects that are written by both activities, 0.5 to the objects that are written by one and read by the other activity, and 0.25 to the objects that are read by both activities. If an object is read and written by an activity we only consider the written (major) operation.

TABLE I: T_P of `PlanApproval`

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
t_1	0	1	0	0	0	0	0	0
t_2	0	0	1	0	0	0	0	0
t_3	0	0	0	1	0	0	0	0
t_4	0	0	0	0	1	1	0	0
t_5	0	0	1	0	0	0	0	0
t_6	0	0	0	0	0	0	1	0
t_7	0	0	0	0	0	1	0	1
t_8	0	0	0	0	0	0	0	0

TABLE II: T_D for the `PlanApproval`

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
t_1	0	1/4	1/4	1/4	1/4	0	0	1/2
t_2	1/4	0	3/4	3/4	5/4	1/2	1/2	3/2
t_3	1/4	3/4	0	3/2	5/4	1/4	1/4	1
t_4	1/4	3/4	3/2	0	5/4	1/4	1/4	1
t_5	1/4	5/4	5/4	5/4	0	1/2	1/2	3/2
t_6	0	1/2	1/4	1/4	1/2	0	5/4	3/2
t_7	0	1/2	1/4	1/4	1/2	5/4	0	3/2
t_8	1/2	3/2	1	1	3/2	3/2	3/2	0

TABLE III: T for the `PlanApproval`

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
t_1	0	5/4	1/4	1/4	1/4	0	0	1/2
t_2	1/4	0	7/4	3/4	5/4	1/2	1/2	3/2
t_3	1/4	3/4	0	5/2	5/4	1/4	1/4	1
t_4	1/4	3/4	3/2	0	9/4	5/4	1/4	1
t_5	1/4	5/4	9/4	5/4	0	1/2	1/2	3/2
t_6	0	1/2	1/4	1/4	1/2	0	9/4	3/2
t_7	0	1/2	1/4	1/4	1/2	9/4	0	5/2
t_8	1/2	3/2	1	1	3/2	3/2	3/2	0

Definition: Given a schema $P = (N, s, f, F, O, \rho, \omega)$, for each pair of activities $a_i, a_j \in N$, $T_D(a_i, a_j) = |\omega(a_i) \cap \omega(a_j)| + 0.5 \times |(\rho(a_i) \cap \omega(a_j)) \cup (\omega(a_i) \cap \rho(a_j))| + 0.25 \times |\rho(a_i) \cap \rho(a_j)|$

Finally, we aggregate these two relations to define the final relation T.

Definition: Given a schema $P = (N, s, f, F, O, \rho, \omega)$, for each pair of activities $a_i, a_j \in N$, $T(i, j) = T_P(i, j) + T_D(i, j)$.

Example: Continuing with the `PlanApproval` process, Table I shows the relation T_P for the process. For example, $T_P(t_4, t_6) = 1$, because there is a path t_4, g_2, g_3, t_6 in P where both g_2 and g_3 are gateway nodes. Table II shows the relation T_D for the process. For example, $T_D(t_1, t_2) = 1/4$, because the only shared object between t_1 and t_2 is `Est` where both activities read `Est`, and Table III show the relation T for the `PlanApproval` process. For example $T(t_6, t_7) = 9/4$, because $T_P(t_6, t_7) = 1$ and $T_D(t_6, t_7) = 5/4$.

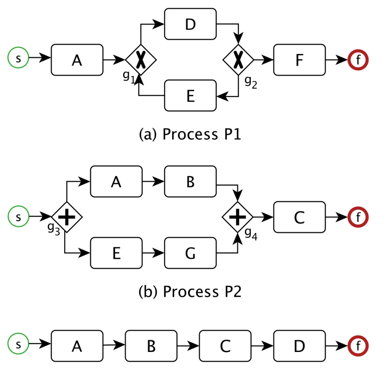
A microservice has a set of operations where each operation is an activity node in the corresponding process schema. To identify microservices, final relation (T) is clustered using a genetic algorithm and turbo-MQ [16] fitness function. The algorithm randomly partitions activities into K clusters and use Turbo-MQ to measure the fitness of the identified clusters. Then, in each iteration, the algorithm tries to increase the fitness by making changes in the previous partitions. This process continues till the fitness converges. To calculate Turbo-MQ, we sum up the *Cluster Factor* for each cluster k (CF_k) as it follows in Eq 1.

$$\text{Turbo-MQ} = \sum_{i=1}^{k} CF_i,\ CF_i = \begin{cases} 0 & \mu = 0 \\ \frac{2\mu_i}{2\mu_i + \sum_{i=1, i \neq j}^{k} (\delta_{i,j} + \delta_{j,i})} & \text{otherwise} \end{cases} \tag{1}$$

In this relation, CF_i is the cluster number i, μ_i indicates the number of intra-relations between activities within the cluster i and $\delta_{i,j}$ represents the number of inter-relations between activities in cluster i and cluster j.

Example: Continuing with the `PlanApproval` process, 3 microservices can be identified, `Receive Estimation` and `Provide Plan` are operations of the first identified microservice. `Evaluate Quality`, `Evaluate Budget`, and `Edit Plan` are operations of the second service, and `Provide Order`, `Check Order`, and `Sign Order` construct the third service.

We now proceed to identify microservices from multiple processes. Software systems are combination of different business processes where these processes may have shared activities or objects. In the previous part, we considered only a single business process and identified microservices from that process. However, our method can be extended to identify microservices from a set of business processes. We only need to consider all the processes together and redefine relations T_P, T_D, and T. Consider Fig. 3(a)-(c) where we have three processes P_1, P_2, and P_3 and there are totally 7 activities A to G in these three processes. Fig. 3(d) shows the data object reads and writes. As can be seen the set of all objects is $\{O_1, ..., O_5\}$. To compute $T_P(a_i, a_j)$ for each pair of activities a_i and a_j we just aggregate $T_{P_k}(a_i, a_j)$ of all process P_k in the set of business processes. For example, since there is a direct edge between activities A and B in both P_2 and P_3, $T_P(A, B) = 2$. For each pair of activities a_i and a_j the value of $T_D(a_i, a_j)$ is same in all the processes (if the process has both activities), because an activity even in different processes use the same set of objects, therefore the final relation T_D is the union of relations T_D of the processes. Finally, relation T can be define as before by aggregating T_P and T_D. The resulted microservices are shown in Fig. 3(e) where A and B construct the first microservice, C and D are the operations of the second microservice, and the third service consists of E, F, and G.



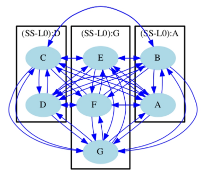
(a) Process P1

(b) Process P2

(c) Process P3

	Reads	Writes
A	O1	O1,O2
B	O1,O2	O1,O2
C	O2,O3	O3,O4
D	O1,O3	O3,O4
E	O3	O1,O5
F	O3,O5	O5
G	O5	O5

(d) object reads and writes



(e) Identified Services

Fig. 3: Identification of microservices from multiple processes

V. Experimental Evaluations

In this section, four experiments are conducted to evaluate the accuracy of the microservice identification approaches. We consider four parameters: number of activities (#A), number of gateways (#G), number of objects (#O), and number of processes (#P) and in each experiment we study the effect of one parameter on the accuracy of the identification approaches.

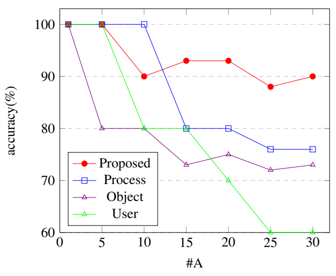


Fig. 4: # of Activities

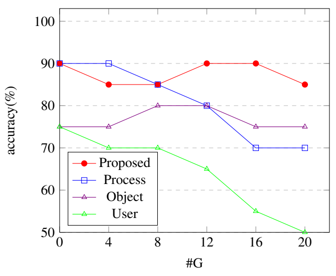


Fig. 5: # of Gateways

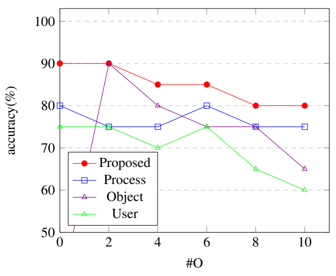


Fig. 6: # of Objects

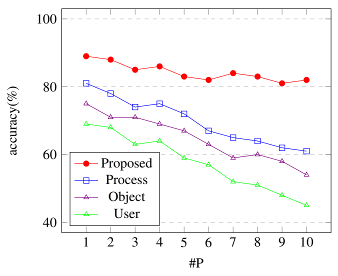


Fig. 7: # of Processes

Here, accuracy means the number of operations (activities) that are clustered in the correct microservice to the total number of operations. To find the correct microservices we asked a group of domain experts to identify microservices. We also take four approaches into account: (1) a user-driven method where 3 developers are asked to identify microservices, (2) process-driven where we consider only the structural dependency of activities within processes (relation T_p), (3) object-driven where we consider only the dependency of activities based on their used data objects (relation T_d), and (4) the proposed method which is the combination of methods (2) and (3).

In the first experiment, we identify microservices from a set of processes with a different number of activities, $1 \leqslant \#A \leqslant 30$, but same number of gateways ($\#G = 4$) and objects ($\#O = 3$). Fig. 4 shows the results of the first experiment where the accuracy of the proposed method is almost independent of the size of the process.

The next experiment studies the effect of process complexity (number of gateways) on the accuracy where a set of processes with $0 \leqslant \#G \leqslant 20$, but similar number of activities ($\#A \sim 20$) and data objects ($\#O = 3$) are taken into account. Fig. 5 demonstrates the results where the accuracy of the proposed method is almost independent of the complexity of processes. Here, the accuracy of the object-driven method is also independent of the process complexity, because it does not consider the structure of the processes.

In the third experiment, we study the effect of the number of objects on the accuracy where we consider a set of processes with $0 \leqslant \#O \leqslant 10$, similar number of activities ($\#A \sim 20$) and gateways ($\#G = 4$). Fig. 6 presents the results of this comparison where the proposed method is almost independent of the number of objects. Since we ignore the degree of importance of different objects, increasing the number of objects has more affect on the accuracy of the proposed method in this experiment.

Finally, in the last experiment, we observe the effect of the number of processes ($\#P$) on the accuracy in a system with multiple processes. The settings is similar as before, i.e., on average, in each process: $\#A \sim 20$, $\#G = 4$, and $\#O = 3$. Fig. 7 shows the results of this experiment.

VI. Conclusions

This article aims to provide a method to identify an independent collection of highly inter-related activities as microservices. To this end, dependencies between activities are measured not only regarding inter-connections between activities within business process models but also considering their dependencies in terms of accessing to same data objects. Our experiments show that the proposed method can identify cohesive, loosely coupled, and fine-grained microservices from a single business process, or a set of processes.

Although in the proposed method two aspects of process structure and object access are taken into account, the method can be easily generalized to other aspects such as requirements, resources, or ownerships.

References

[1] M. Mazzara, R. Mustafin, L. Safina, and I. Lanese, "Towards microservices and beyond", 2017.

[2] S. Hassan and R. Bahsoon, "Microservices and their design trade-offs: A self-adaptive roadmap", in 2016 IEEE International Conference on *Services Computing (SCC)*. IEEE, 2016, pp. 813–818.

[3] J. Lewis and M. Fowler, "Microservices: a definition of this new architectural term", *MartinFowler.com*, vol. 25, 2014.

[4] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina, "Microservices: yesterday, today, and tomorrow", in *Present and Ulterior Software Engineering*. Springer, 2017, pp. 195–216.

[5] S. Newman, "Building microservices: designing fine-grained systems". O'Reilly Media, Inc., 2015.

[6] N. Dmitry and S.-S. Manfred, "On micro-services architecture", *International Journal of Open Information Technologies*, vol. 2, no. 9, 2014.

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

[8] M. Gysel, L. Kölbener, W. Giersche, and O. Zimmermann, "Service cutter: A systematic approach to service decomposition", in *European Conference on Service-Oriented and Cloud Computing*. Springer, 2016, pp. 185–200.

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

[10] Z. Cong, A. Fernandez, H. Billhardt, and M. Lujak, "Service discovery acceleration with hierarchical clustering", *Information Systems Frontiers*, vol. 17, no. 4, pp. 799–808, 2015.

[11] M. J. Amiri, S. Parsa, and A. M. Lajevardi, "Multifaceted service identification: process, requirement and data", *Computer Science and Information Systems*, vol. 13, no. 2, pp. 335–358, 2016.

[12] M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for qos-based web service composition", in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 11–20.

[13] M. Weske, "Business process management architectures", *Business Process Management*. Springer, 2012, pp. 333–371.

[14] B. P. Model, "Notation (bpmn) version 2.0", *OMG Specification, Object Management Group*, pp. 22–31, 2011.

[15] M. J. Amiri and M. Koupaee, "Data-driven business process similarity", *IET Software*, vol. 11, no. 6, pp. 309–318, 2017.

[16] B. Mitchell, M. Traverso, and S. Mancoridis, "An architecture for distributing the computation of software clustering algorithms", in Proceedings of Working IEEE/IFIP Conference on *Software Architecture* IEEE, 2011, pp. 181–190.