Process Fragmentation: An Ontological Perspective

Asef Pourmasoumi[1,2(✉)], Mohsen Kahani[2], Ebrahim Bagheri[1], and Mohsen Asadi[3]

[1] Department of Electrical and Computer Engineering, Ryerson University, Toronto, Canada
{a.pourmasoumi,bagheri}@ryerson.ca
[2] Web Technology Lab, Ferdowsi University of Mashhad, Mashhad, Iran
kahani@um.ac.ir
[3] School of Interactive Arts and Technology, Simon Fraser University, Surrey, Canada
masadi@sfu.ca

Abstract. Process fragmentation provides the basis for re-usability and process improvement. Various re-searchers have already introduced different definitions for what constitutes a reasonable process fragment, and have offered algorithmic support for identifying such fragments. As we will show in this paper, some of these definitions suffer from ambiguity or imprecision. Therefore, the objectives of this paper are twofold: first, we provide an ontological assessment of the various process fragment definitions based on the well-known Bunge's Ontology and its process representational model, GPM. On this basis, we then extract the most important features of these definitions in order to formalize a precise definition for process fragments and propose a precise and non-ambiguous definition: *morphological fragmentation*. We present our work through a case study and report on our observations.

Keywords: Process model fragmentation · Ontological theory · Generic process model (GPM)

1 Introduction

Organizational mining focuses on discovering organizational structures, social networks, and resource allocation patterns [1]. Organizational mining was traditionally introduced within a single organization. The growing increase of IT infrastructure needs has led many organizations to reuse or share resources and processes leading to the introduction of *cross-organizational mining* [2]. Cross-organizational mining considers organization's IT infrastructures from two perspectives. In the first case, different organizations work with each other to perform the same process instances [3]. In the second case, different organizations are separately handling the same process while sharing experiences, knowledge, or a common infrastructure [2]. In this case, each organization could be executing a variant of the same process family, such as the sale process offered by Salesforce.com. In other words, these organizations use the common infrastructures of Saleforce.com for handling their processes. However, they do not execute the exact same process and often customize and build a variation of the sale process. These customized processes share many *commonalities* and some

K. Gaaloul et al. (Eds.): BPMDS 2015 and EMMSAD 2015, LNBIP 214, pp. 184–199, 2015.
DOI: 10.1007/978-3-319-19237-6_12

degree of *variability*. The analysis and mining of these commonalities and variabilities can lead to valuable insight for the organizations.

In [1], a good review of process mining from the organizational perspective and its existing challenges has been reported. One of the challenges that is of interest to our work in this paper is the identification and analysis of *common process fragments* from among multiple variants of the same business process within different peer organizations. To the best of our knowledge, most of the existing process fragment definitions have only considered the practical implications of their work and little, if any, theoretical analysis has been done [4]. In this paper, we investigate the use of ontological theories for the theoretical analysis of process fragmentation models. An ontological theory defines necessary constructs for describing the processes and structure of the world in general [5]. Ontological theories have been used to evaluate modeling languages in terms of the correspondence of ontological concepts to modeling constructs. Bunge's ontology ([5]) is a widely used ontological theory that has been used to evaluate several conceptual modeling languages [5] [6]. For example in [7], the authors used Bunge's ontology for evaluating BPMN and workflow nets. This ontology includes a set of high level constructs for representing real world phenomena. The evaluation of modeling languages is based on the assumption that an information system is an artifact that represents a real-world domain.

Bunge's ontology has also been used for rep-resenting process models. In [8], a generic model, called GPM, is derived from Bunge's ontology for semantically describing process models. GPM gives a formal abstract view of a process model in terms of state transitions that occur rather than using common notions such as control flows and activities [8]. In the other words, GPM can be viewed as a mapping between process models and the real world.

Since GPM is an ontological representation of real world process models, we presume that it is appropriate for analyzing the fragments of processes models that represent real-world domains. Therefore, our work is systematically grounded in concepts from Bunge's Ontology and GPM.

In this paper, we provide the following concrete contributions:

- We gather and classify some of the main process fragment definitions and systematically discuss their pros and cons. We present a comparative analysis of these definitions.
- We present a formal representation of process fragments derived from existing definitions. This formal representation can be used as a theoretical basis for comparing and designing new process fragmentation techniques.
- Using Bunge's ontological representation of process models (GPM) we theoretically present a process fragment definition, which covers the weaknesses of previous process fragment definitions.

The paper proceeds as follows. In Section 2, we briefly introduce Bunge's ontology and the Generic Process Model (GPM). We then introduce a running example, which will be used throughout the paper. In Section 3, we review different definitions of process fragments and discuss the pros and cons of each. Here, we are not focused on the algorithmic support for process fragmentation and are only analyzing the theoretical basis for the process fragment definitions. We theoretically define the notion of

process fragments using Bunge's ontology and GPM. Section 4 applies GPM for presenting a process fragment definition. In Section 5 we analyze the various fragment definitions through a case study. Finally, in Section 6 we conclude the paper and suggest directions for future work.

Table 1. Fundamental ontological constructs in the Bunge-Wand-Weber representational model [5]

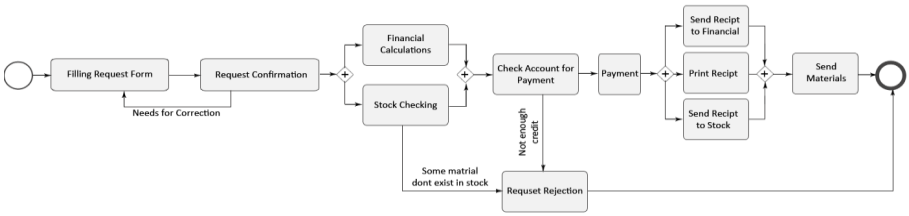
Fundamental Ontological Construct	Description
Thing	The basic unit in the Bunge's ontological model is thing and can be in two types: simple and compound. A compound thing is made up of other things.
Property	Each property can be described using a function (called attribute function) that maps the thing into some values. Things can have several properties.
State	The state of a thing is a vector of values for each property functions of that thing.
Transformation	A mapping from a domain containing states into a co-domain containing states is called transformation.

2 Preliminaries

2.1 Bunge's Ontology

Bunge's ontology has been widely used for evaluating several conceptual modeling languages [5] [6]. In this paper, we are interested in the theoretical analysis of process fragments through Bunge's ontology. Bunge models the world as a world of systems [9]. In Bunge's ontological model, the "*world is made up of substantial things which possess properties*" [9]. Since Bunge's ontology provides concepts for representing real world phenomena, it seems appropriate to be used for analyzing process models in real software systems, which represents real-world domains [7].

Bunge's ontological model contains four essential concepts: thing, property, state and event. Table 1 provides an overview of the fundamental ontological constructs [5]. Things are elementary units and can be specific instances of person, building, car, book and etc. A "*property*" of a thing can be any intrinsic or mutual (meaningful only in the context of two or more things) feature of it like height, color, weight, and shape. A "*state*" is the vector which contains the values of all property functions of a thing [10]. A "*state law*" makes a restriction on the values of the thing's properties to a subset that is considered lawful. The set of thing's states that conform to the state laws of the thing are called the "*lawful state space*". An "*event*" occurs when a change in the state of a thing can be seen. The set of all possible events of a thing is called its "*event space*" [10]. A set of things that possess a common property is termed a "*class*". Wand et al. have extended Bunge's ontology with 28 (real-world) constructs [10] [5].



(a) An example of purchasing process model.

Things	Properties (attributes)	State Space	Lawful Event Space
Specific instances of Request, Stock, Material, Employee, Applicant, Account	Request(ID, Date, Materials) Stock(ID, Materials, No), Material(ID, Name, Price), Employee(ID, Name, Grade), Applicant(Code, Requests), Account(ID, Credit, ApplicantID)	Request-Status(Filled, Confirmed, Rejected, Audited, Manager Confirmed) Material-Status (Available, Not Exist) Account-Status(A, B, C, D)	Request{ (Filled→Confirmed), (Filled→Filled), (Confirmed →Audited), (Confirmed → Stock Checked), (Stock Checked → Rejected), (Stock Checked → Manager Confirmation), (Audited →Manager Confirmation)}, Account-Status{(A→B), (B→C), (C→B), (B→A)}

(b) Ontological representation of mentioned purchasing process.

Fig. 1. An example of purchasing process

2.2 Generic Process Model (GPM)

The Generic Process Model (GPM) provides a process specification semantics based on ontological constructs [8]. GPM is considered as a framework for reasoning about process models according to their real-world meaning based on Bunge's ontology. GPM was used for various purposes such as analyzing the validity of process models, describing the conception of goals in business processes, and to interpret of control flow elements [8].

GPM focuses on the notion of *domain* by the use of Bunge's ontology concepts. A domain is represented by a set of state variables. The state of the domain can change for two reasons: first, due to the internal events which occur within the domain and second by external events which stimulate from outside the domain [8]. A state that changes due to actions in the domain is called "*unstable state*" and a state that only changes due to the actions of the domain's environment is called "*stable state*" [8]. Unstable states can be manifested as internal events and stable states as external events. A complete definition of GPM can be found in [8].

In [11], a process model is defined based on GPM as following:

Definition 1. [11]: *a process model is a tuple* $< I, G, L, E >$ *where,*

I: is a subset of unstable states of the domain (initial states),

G: is a subset of stable state (goal set),

L: is the set of state transitions,

E: is a set of relevant external events. ■

One of the advantages of this definition is that GPM explicitly addresses the goal of a process and checks the validity of a process design against its defined goal [11]. In this paper, we will use GPM for theoretically analyzing process fragments.

Table 2. A review of process fragment definitions

Group	References	Fragment Definition
G1	[16] [17] [18] [19]	Components with single input and single output control flow arc (SESE).
G2	[13]	SESE components which have connected nodes and their nodes have high label similarity.
G3	[20]	A portion of design process adequately created and structured for being reused during the composition and enactment of new design processes.
G4	[21] [22] [23] [24]	A connected portion of a process intended for reuse and contains no cycles and a single control flow linking up two distinct activities. It is made of at least one activity and several controls.
G5	[12]	A partition of a workflow model, and consists of a source transition, all the transitions are reachable from the source transition, and all the linking places of these transitions.
G6	[25] [26]	A logically different, smaller model part of input process model which extracted with the intention to distributing over different execution and controlling partners.
G7	[4]	A part of process models that contain process's elements such as activities, data flows, and controls.
G8	[27]	A connected graph with significantly relaxed completeness and consistency criteria compared to an executable process graph which contains a process start or end node and at least one activity and is not necessarily directly executable.

2.3 A Running Example

In this section, we present a running example of a "purchasing process" for better clarifying Bunge's ontological constructs.

Figure 1-a shows the flow of this process model. It starts by "filling request form" and ends with "send material" or "request rejection" activities. After the "filling request form" activity, the request will be checked and if it is confirmed then two actions can be done simultaneously: the stock will be checked and the financial calculation will be done. If the requested material exists in stock, then the user account will be checked and payment will be processed. Afterwards, the receipt will be printed and sent along with the material to the customer. This process is a simplified purchasing process. The fundamental Bunge's ontological constructs for this process are shown in Figure 1-b. We will use this example throughout the paper.

3 Process Fragmentation

Process fragmentation is referred to the act of categorizing process model elements such as activities, data flows, and control flows into groups [4]. The created groups are known as *process fragments*. *Process fragmentation* is the basis for techniques supporting reusability, parallel execution, management and analysis of process models [12]. It is also known as *process decomposition* or *process modularization* [13].

In this section, we systematically review some of the existing process fragment definitions from the literature and discuss their pros and cons. As mentioned earlier, we are only interested in the process fragment definitions in this paper and not in the algorithms that facilitate the identification of such fragments. By identifying the strengths and weaknesses of different process fragment definitions, one can propose a definition that would cover the pros of existing definitions and cover their cons.

3.1 A Review of Process Fragment Definitions

In order to identify the main work in process fragment definition, we first started by focusing on the main survey papers in this domain [4] [14]. We then gathered additional papers by searching for the keywords mentioned in these survey papers on reliable databases including *CiteSeerX*, *ScienceDirect*, ACM Portal, *SpringerLink* and *IEEEXplore*. We then extracted and classified these papers based on their fragment definition and then selected only those papers that had been cited more than 10 times ac-cording to Google Scholar. The result is shown in Table 2.

In order to analyze these definitions and identify their pros and cons, we required some comparative criteria. In [4], some classification criteria for process fragmentation techniques is provided. These criteria provide a foundation for the classification of process fragmentation algorithms and can be useful for the evaluation of these techniques. For example, some of the criteria state *why is the process model fragmented, how is the fragmentation performed, who performs the fragmentation, when is the fragmentation performed in the process model lifecycle*, among others. Most of these criteria are independent of the fragment *definition* and are with respect to different aspects of the fragmentation *algorithms*. Therefore the criteria introduced in [4] cannot be directly applied for our work.

We suggest three criteria, *structural restrictions for input/output process models*, *ambiguity* and *determinism.* We don't claim that these criteria are complete, but they can highlight main weaknesses regarding the fragment definitions. These criteria are presented as research questions in the following:

Q1: Does the definition impose any structural restriction on the input process model or the output fragments?

Some definitions consider limitations on the input or output processes. For example, some definitions require the process fragment not to have any cycle or all transitions should be reachable from the start node.

Q2: Does the definitions have any elements of ambiguity or leave room for different interpretation?

Given some of the definitions do not have a theoretical representation and are written in natural language, there might be room for different interpretations of the definitions. For example, some definitions just state the portions of process models that are suitable for reusability are fragments. Such definitions have ambiguity and there could be different interpretations.

Q3: Is the definition precise and deterministic?

This criterion specifies whether the definition will always guarantee the extraction of the exact same process fragments for the same input process model or not. This criterion is different from ambiguity. In the case of determinism, an implementation of a non-deterministic definition can be viewed as being not a function and for the same input, could produce different outputs. So, a definition might not be ambiguous but be non-deterministic.

Table 3. Analyzing various definitions of process fragments based on three criteria: structural restriction, ambiguity, and determinism

Group	Structural Restriction	Ambiguity	Non-Determinism
G1	—	—	■
G2	■	—	—
G3	—	■	■
G4	■	—	■
G5	—	—	■
G6	—	■	■
G7	—	■	■
G8	■	—	■

Table 4. All features extracted from various definitions of process fragments

F#	Features
F_1	Process fragment has single input and single output control flow arc (SESE)
F_2	Process fragment must be connected.
F_3	Fragment's node must have label similarity.
F_4	Process fragment should contain no cycles.
F_5	It is made of at least one activity and, of several control (dangling or not) and data flows.
F_6	All the transitions must be reachable from the source transition and all the linking places of these transitions.

Table 5. Extracted features against the various definitions

Definition Group ID	F1	F2	F3	F4	F5	F6
G1	■	—	—	—	—	—
G2	■	■	■	—	—	—
G3	—	—	—	—	—	—
G4	—	■	—	■	■	—
G5	■	—	—	—	—	■
G6	—	—	—	—	—	—
G7	—	—	—	—	—	—
G8	■	■	—	—	■	—

In Table 3, the results of the evaluation of existing process fragment definitions based on the three criteria are shown[1]. As can be seen, most of the definitions are non-deterministic and have no structural restrictions. For example, the definition in Group 1 has no ambiguity and is clear, but is non-deterministic, because in this definition any part of process can be considered as a fragment and for the same input process, different fragments can be produced. Group 2 has a more precise definition and has structural restriction on process inputs (must be connected). This definition is clear and has no ambiguity. Since it uses precise label similarity, it is deterministic. The definition in group 3 do not have clear structural restriction on process input and the definition is not clear which parts of input can be taken as fragments. The definition in group 4 states that fragments should be connected, have no cycles, and have at least one activity and a single control flow linking up two distinct activities. It is clear but is non-deterministic because it does not work as a function. The definition in group 5 is explicit and has no structural restriction but again it is non-deterministic, since for the same input process different process fragments can be generated. The definitions in Groups 6 and 7 are descriptive

[1] The evaluation was conducted with the first author by investigating the coverage of the criteria for each definition. In the cases that there were uncertainties for the coverage, they have been discussed between the authors to reach conclusions about the coverage.

and contain ambiguity. Finally, in the definition in group 8, a fragment is a connected graph (structural restriction) that has a start and end node and at least one activity and is not necessarily directly executable. So, this definition is clear and has no ambiguity. It is non-deterministic because with the same process we can have lots of fragments that have start and end nodes and be also connected.

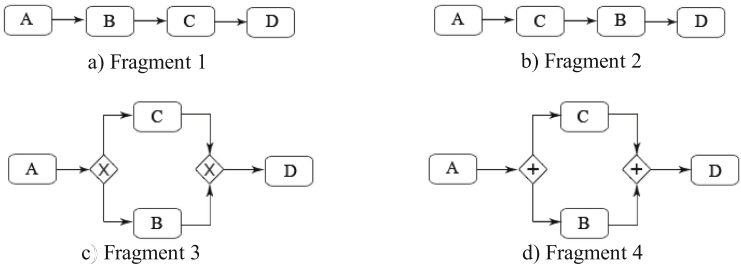


Fig. 2. Four fragments of *purchasing process* with different structures which operationally are identical. Activity A is "filling request form", B is "financial calculation", C is "stock checking" and D is about department management confirmation.

We further analyzed the various fragment definitions and identified the top six features that they had in common as shown in Table 4. These features represent the characteristics of the process fragments that are described by the definitions. Feature F_1 is the most widely seen feature for fragments. This feature implies that each fragment must have a single input and single output. Features F_2 and F_6 are concerned with the concept of connectivity. A node belongs to a fragment if it has a connection with at least one node within the fragment. Feature F_3 has a semantics point of view. It implies that the nodes that have more similar labels have a higher probability of belonging to the same fragments [13]. Natural language processing (NLP) tools can be used for detecting the similarity of the labels of the nodes in the fragments. Feature F_4 creates a structural limitation on the fragments that they should not contain cycles. Feature F_5 implies that in each fragment at least one action needs to be done.

In Table 5, we show each feature against each definition group. In other words, Table 5 shows for each definition group, what the features of their fragments are. In this table, the rows show each definition group's id (from Table 3) and the columns show the features (from Table 4). From this table, it can be understood that Features F_1 and F_2 are the most frequent features for process fragments within the literature, respectively.

Now, we exploit the features of Table 4 in order to provide a formal definition for a process fragment as follow:

Definition 2. *A process fragment* F *is a directed graph described as* $F(A, G, R, s, e)$ *where:*

1) $R \sqsubseteq (A \times G) \cup (G \times A) \cup (G \times G)$,
2) $|A| \geq 1$,
3) $\forall\, t \in (A \cup G)\ \exists\ v_0 = s, v_1, v_2, \ldots, v_k = t\ ((v_{i-1}, v_k) \in R, 1 \leq i \leq k)$,
4) $if\ \ \exists\ n_1, n_2\ (n_1 = s, n_2 = s)\ then\ n_1 = n_2$,
5) $if\ \ \exists\ n_1, n_2\ (n_1 = e, n_2 = e)\ then\ n_1 = n_2$,
6) $\nexists t \in\ (A \cup G)\ \big((v_t, v_i) and\ (v_k, v_t) \in R\big)\ and\ ((v_j, v_{j+1}) \in R,\ i \leq j < k)$ ■

A is the set of activities, G represents gateways and R is the set of control flow relations and s is single input and e is single output nodes. In this formalism, $|A| \geq 1$ (line 2) ensures feature F_5 and the expression in line 3 is enforces F_2 and F_6. Two if-clauses in this formalism (lines 4 and 5) together represent F_1 and the later expression is equal to F_4. In this definition we did not include feature F_3. The reason is that label similarity is based on the assumption that process names are always selected meaningfully and consistently [13]. For small processes or cross-organizational processes, this assumption is not necessarily always true.

4 Analyzing Process Fragments Using Bunge's Ontology

In Definition 2, we included the most prominent non-ambiguous features identified in the definition groups for process fragments. This definition is non-ambiguous and does not contain any ambiguous feature, has no structural limitation on inputs and outputs and is deterministic. Nevertheless, it would not be useful for identifying process similarity or multi-process analysis. This is due to the following reason: in Definition 2, two fragments within two process variants are shared, if and only if they are identical [15]. For example, in Figure 2, fragments 1, 2, 3 and 4 are not identical (Based on all of the definitions in Table2). Based on *Definition2*, since the R set of these four fragments are not identical, so they cannot be captured as common fragments between two process models. Therefore, with these definitions, common fragments are either complete matches or not considered at all.

In the example shown in Figure2, All fragments try to check a purchasing request; however, each achieves this in a different way. Activity A is filling a request by the customer for purchasing some goods. All fragments start with this activity. Activity B is the computation of tax and total price of goods and activity C checks the stock for requested goods. Finally, activity D is about department management confirmation. As seen in Figure 1, activities B and C can be placed in different relations to each other. This difference can be probably due to various branch managers' choices. In fact, these fragments are performing a similar task and would be considered to be very similar fragments that only have structural variances.

In the next section, we will theoretically define the notion of process fragments using GPM and derive a new conceptual definition for a process fragment that would address the above issue among others.

4.1 Process Fragment Based on GPM

Using Bunge's ontology, it can be inferred that there is a mapping between information systems and concepts in the real world. The concepts (domain) in the real world are made of sub-concepts (sub-domain). The structures and processes of the real world can be represented by constructs in the ontological models. So, we can define sub-processes or fragments of a process model based on its ontological mapping in the real world. In [8], the sub-domain is defined based on GPM as following:

Definition3. (*sub-domain*) [8]: *A sub-domain is part of the domain described by a subset of the set of domain state variables.* ■

It must be noted that using this definition, there might be many ways to divide a domain into sub-domains and not all of them will be meaningful. One of the main applications of process fragmentation is reusability. It means that process fragments can be used in the design of different processes models. In this case, process fragments should be run independently. In this respect, in the real world, partitioning of a domain into independent sub-domains is possible. Partitioning of a domain into independently-behaving sub-domains is the result of different actors existing in the domain. In [8], an independent sub-domain is defined as follows:

Definition 4. (*independent sub-domain*) [8]*: A sub-domain will be called independently behaving (or independent) in a given state (of the sub-domain) if the law projection on the sub-domain is a function for this state.* ■

In this case, the law projection is a function that depends only on the sub-domain's state variables. In the other words, the meaning of *Definition 4* is that each sub-domain behaves independently and ends on a stable state of the sub-domain. This stability can lead to the stability of the whole domain in the goal states or other states [8]. So, in this definition the stable states (initial and goal states) of the sub-domain are important.

We use *Definition 4* for creating a mapping between sub-domains in the real world and process fragments in information systems. So we can define process fragments based on GPM as follows:

Definition 5. *(process fragments): A subset of a process model is called a process fragment if it starts and ends with stable states and there exists at least one transformation inside it.* ■

Now, using GPM-based definition of process fragment (*Definition 5*), we can develop a model for process fragments that does not necessarily require a complete and exact match for finding similar fragments. As we will show, this allows us to assess the similarity of two process fragments beyond a binary match or no-match. In the next Section, the new notion for process fragment, building on Definitions 2 and 5, is introduced.

4.2 Morphological Fragments

In *Definition 5,* a process fragment is defined based on GPM. The main focus of this definition is on the stable states of fragments as a point of separability. Unstable states inside the sub-domain are responsible for the behavior of the sub-domain. It can be understood that two fragments f_1 and f_2 have equal behavior if they have equal stable states (I_1=I_2 and G_1=G_2) and also have equal *transformation* sets (In [8] transformation set of a process is equal to its Activities set). The order of transformation sets is equal to the way which processes execute. In the other words, with equal stable states and transformation sets, different transformation orders (law) for two fragments show that the fragments do similar tasks in different ways. In other words, they represent the same behavior but not necessarily the same structure. Now, we can define some measures for extracting common fragments from a family of process variants based on this observation.

Definition 6: Two fragments $f_1(A_{f_1}, G_{f_1}, R_{f_1}, s_{f_1}, e_{f_1})$ and $f_2(A_{f_2}, G_{f_2}, R_{f_2}, s_{f_2}, e_{f_2})$ are behaviorally similar, called morphological fragments, iff:

$$s_{f_1} = s_{f_2} \;\&\; e_{f_1} = e_{f_2} \;\&\; A_{f_1} = A_{f_2}$$ ■

In this definition, the start and end points are equal to the stable states in the *Definition 5* and the relation between internal nodes is ignored given the above explanation. The reason is that two fragments that have equal start/end points and have equal activities sets, and performing similar tasks, would be considered to be very similar fragments that only have structural variances hence, the term *morphological fragment*. For example in Figure 2, there are four fragments that check the purchasing request of a customer in different ways. These differences can affect the efficiency and effectiveness of the whole process. So detecting these fragments as common fragments among a family of process variants can lead to added value for organizations. All of these four fragments have equal start/end point and their internal activities are identical. So they are morphologically identical. The different relationships among internal activities show the different ways of doing same the task.

Table 6. Similarity patterns between two sub-processes

Behavioral Similarity		
Stable State	**Transformation Space**	**Class Name**
Full Similarity-Double Side	Full Similarity-Double Side	Full Similarity-Double side among Process
	Full Similarity-One Side	Partial Similarity among Process
	Partial Similarity	
	Dissimilarity	Complete Dis-similarity among process
Full Similarity-One Side	Full Similarity-Double Side	Complete Dis-similarity among process
	Full Similarity-One Side	
	Partial Similarity	
	Dissimilarity	
Partial Similarity	Full Similarity-Double Side	Complete Dis-similarity among process
	Full Similarity-One Side	
	Partial Similarity	
	Dissimilarity	
Dissimilarity	Full Similarity-Double Side	Complete Dissimilarity among process
	Full Similarity-One Side	
	Partial Similarity	
	Dissimilarity	

Definition 6 allows the identification of process fragments that are behaviorally similar but not structurally identical. It is now possible to define the degree of morphological similarity based on the degree of the transformation spaces similarity of the fragments. But before that, at first we use the definitions mentioned in [28] for a set of general similarity patterns between two sets of phenomena (By phenomena we refer to any possible observation that can be made about the domain or part of it).

Assume $A = \{a_1, a_2, ..., a_n\}$ is a set of phenomena belonging to domain D_1 and $B = \{b_1, b_2, ..., b_m\}$ is a set of phenomena belonging to domain D_2. We can see one of the following situations with respect to similarity between these two sets:

***Definition7* (Equivalent Sets of Phenomena) [28].** *Phenomenon A_1 is equivalent to A_2 (denoted as $A_1 \equiv A_2$), if and only if there is a unique mapping between elements in A_1 and elements in A_2.* ■

***Definition8* (Similar Sets of Phenomena) [28].** *Phenomenon A_1 is similar to A_2 with respect to p (denoted as $A_1 \equiv A_2$) if and only if there is a subset of A_1 (i.e., $A_1' \subset A_1$) and of A_2 (i.e., $A_2' \subset A_2$) which are equivalent $A_1' \equiv A_2'$. p is the equivalent subset i.e. $p = A_1' = A_2'$.* ■

***Definition9* (Completely Dissimilar Set of Phenomena) [28].** *Phenomenon A_1 is completely dissimilar to A_2 (denoted as $A_1 \neq A_2$) if and only if there are no subsets of A_1 (i.e., $A_1' \subset A_1$) and of A_2 (i.e., $A_2' \subset A_2$) that are equivalent.* ■

Based on Definitions 7-9, we can define the following similarity patterns between two different sets of phenomena:

- *Full similarity double side*: when the sets A_1 and A_2 are *equivalent* (i.e., $A_1 \equiv A_2$).
- *Full similarity one side*: when the sets A_1 and A_2 are *similar* (i.e., $A_1 \cong_p A_2$) and when we have either $A_1' \subset A_1$ and $A_1' \equiv A_2$ or $A_2' \subset A_2$ and $A_2' \equiv A_1$.
- *Partial similarity:* when the sets A_1 and A_2 are *similar* (i.e., $A_1 \cong_p A_2$) and there is no subset of one set that is equivalent to the other set.
- *Complete Dissimilarity:* when two sets are *completely disjoint*.

All of the above similarity patterns can occur between any two sets in the real world. In the case of *partial similarity* we can define the amount of similarity as:

$$S_p(A_1, A_2) = \frac{A_1 \cap A_2}{A_1 \cup A_2} \tag{1}$$

The value S_p is a positive value between 0 and 1 where values of S_p closer to 1 represent higher similarity between A_1 and A_2.

In Table 6, we show all of the similarity patterns that can happen between the two subset of process based on fundamental ontological constructs (we just show *stable state space* and *transformation space* constructs based on GPM, because we want to analyze behavioral similarities between process fragments and structural similarity is not our concern in this paper).

In Table 6, the first row is equivalent to our definition for morphological fragments. It means that two fragments are behaviorally identical if their *stable state space* and *transformation sets* are equivalent (have *full similarity double side* pattern). In the other words, the order of events does not matter. The only important point is that their internal event space (*transformation set*) is equivalent and they have equal start and end nodes (*stable state space*). So, the first row of highlighted part of Table 6 describes full similarity. Also, we can define some degree of similarity between morphological process fragments. If T_1 and T_2 be *transformation* sets of process fragments F_1 and F_2 respectively, then we can define the degree of the similarity between these two process fragments using α:

$$\alpha = \mathrm{S_p}(T_1, T_2) = \frac{T_1 \cap T_2}{T_1 \cup T_2} \tag{2}$$

The two fragments F_1 and F_2 have fully- similarity if their degree of the similarity is equal to 1 ($S_p(p_1, p_2)=1$). The reasons that why we do not consider the other parts of Table 6 as morphological fragments is discussed in Section 5.2.

5 Discussion

In this section, we analyze the running example of Section 2.3 based on the proposed morphological fragment definition. For this purpose, we compare this running example to three other purchasing systems that are similar to the system in our example. In this section, we analyze the running example of Section 2.3 based on the proposed morphological fragment definition. For this purpose, we compare this running example to three other purchasing systems that are similar to the system in our example.

Table 7 shows the ontological representation of all these four systems. Now, using the morphological fragment definition (Definition 6), we show how some structurally different sub-processes of these purchasing processes can be considered to be similar. These fragments are shown in Figure 3. All fragments start and end with the same activities "*filling request form*" and "*department manager confirmation*", respectively. Regardless of their order or composition, the set of internal activities of all these fragments are identical. However, the order and relationships of them are different. All of them check the validity of the purchase request and check the existence of the goods in stock, each in their own way. These differences can be due to various reasons and can effect efficiency and performance of the process. For example, one manager may decide that checking the stock and doing the financial calculations should be done in parallel and another might decide that it should be done sequentially. Undoubtedly this decision will affect various execution characteristics of the process, e.g. time to completion. By identifying similar morphological fragments, one can

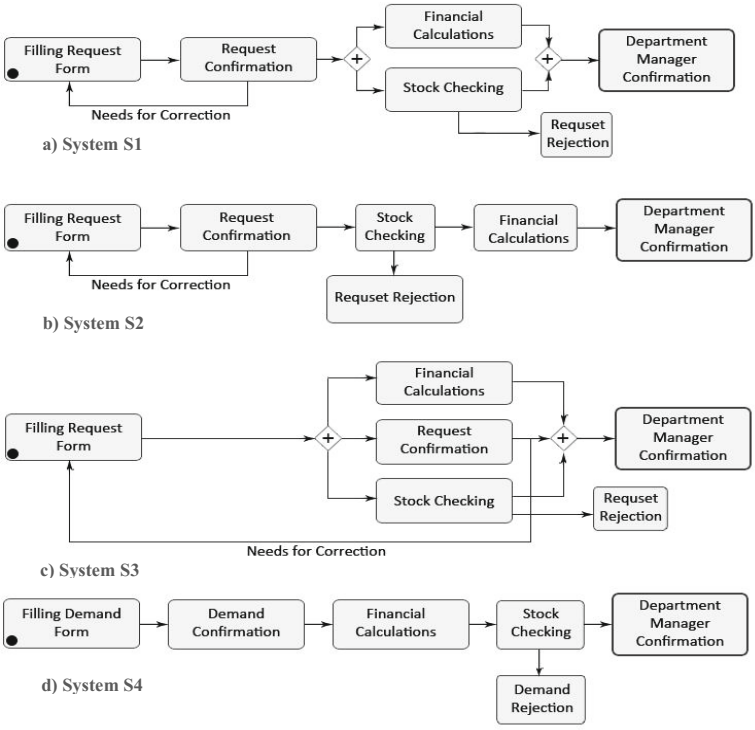


Fig. 3. Four similar fragments corresponding to systems that are shown in Table 3

determine various quantifiable measures for these fragments and use them for the purpose of process improvement. For instance, given the fragments start and end with the same activities, one can decide to replace one morphological fragment with another similar morphological fragment in the hopes to reduce the time to completion. As well, other criteria such as complexity, and cost can be used for optimizing processes through morphological fragments. It should be noted that existing definitions of process fragments as given in Table 2 do not support for this important point.

5.1 Analysis of Degree of Morphological Similarity

The use of the morphological fragment definition can lead to insights for organizations that cannot be otherwise obtained if a strict fragment definition is employed. Under real world scenarios, the number of *exact* fragments that can be mined may not be numerous; therefore, one can use the degree of morphological similarity (α in Equation 2) in order to relax the requirements to some extent. Two fragments are morphologically *identical* if their α similarity is equal to 1. Likewise, two fragments are completely dissimilar if $\alpha = 0$. Otherwise, there is a degree of morphological similarity where degrees closer to 1 represent higher similarity between the identified fragments.

With regards to the criteria shown in Table 3, we can note the following points:

- The morphological fragment definition imposes structural restriction on the process input, i.e., the input process should be connected.
- The definition does not have any ambiguity and is clearly defined; therefore, does not leave room for different interpretations.

The definition is deterministic and would lead to the same set of morphological fragments given a similar value for α.

5.2 Stable States of Morphological Fragments

There are two reasons why we only consider the first tow of Table 6 and not the other three other rows of Table 6 in our morphological fragment definition:

- First, the primary goal of eliciting morphological fragments is business process improvement and/or reusability. For this purpose, the most important property of morphological fragments is *composability*. Therefore, having the same input and output activities for morphological fragments can significantly facilitate replacement and composition of process fragments.
- Morphological fragments do not necessarily guarantee full goal compatibility but rather they point to sub-processes that are likely to be related to similar goals/objectives but this might not be necessarily the case. It should be noted that we would relax attention to accuracy and full compatibility for the sake of finding more potential matches.
- In many cases, if there is partial similarity between two fragments, it is possible by taking a window of smaller size on the main process, to reach to *full similarity-double side*. For example, imagine that we have two fragments: $F_1 =< A,B,C,F,E,D,G,H >$ and $F_2 =< M,H,C,D,E,F,G,A >$. These fragments are not morphological fragments because their stable states are not equals. However, if take a window of smaller size on these fragments, these morphological fragments can be achieved: $F_1' =< C,F,E,D,G >$ and $F_2' =< C,D,E,F,G >$.

6 Conclusion

In this paper, we have systematically analyzed various definitions of process fragments and compared them based on three criteria: structural restriction on input, ambiguity and determinism. We further formally analyzed the definitions based on Bunge's ontological model and its process representational model, GPM. We extracted the most important features of these definitions and formalized a precise definition for process fragments, called *morphological fragments*. Morphological fragments can be valuable for cross-organizational mining and especially useful for process improvement in peer-organizations when the processes are similar but not completely identical.
As future work, we are interested in pursuing the following three areas:

- Designing algorithms that can automatically detect morphological fragments from existing families of process models.
- Extracting common morphological fragments directly from collections of event logs as opposed to formal business process models.
- Providing a more operational definition for morphological fragments. In this case, the goal is to find fragments, which are operationally identical even if the set of activities are not the same.

References

1. Zhao, W., Zhao, X.: Process Mining from the Organizational Perspective. Advances in Intelligent Systems and Computing **277**, 701–708 (2014)
2. van der Aalst, W., et al.: Process mining manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part I. LNBIP, vol. 99, pp. 169–194. Springer, Heidelberg (2012)
3. Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: Mining configurable process models from collections of event logs. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 33–48. Springer, Heidelberg (2013)
4. Mancioppi, M., Danylevych, O., Karastoyanova, D., Leymann, F.: Towards classification criteria for process fragmentation techniques. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part I. LNBIP, vol. 99, pp. 1–12. Springer, Heidelberg (2012)
5. Wand, Y., Weber, R.: On the Deep Structure of Information Systems. Information Systems Journal **5**, 203–223 (1995)
6. Evermann, J., Wand, Y.: Ontology based object-oriented domain modelling: fundamental concepts. Requirements Engineering **10**(2), 146–160 (2005)
7. Jan, R., Marta, I., Michael, R., Peter, G.: How good is BPMN really? Insights from theory and practice. In: Ljungberg, J., Andersson, M. (eds.) Proceedings 14th European Conference on Information Systems. Goeteborg, Sweden (2006)
8. Soffer, P., Kaner, M., Wand, Y.: Assigning Ontological Meaning to Workflow Nets. Journal of Database Management **21**(i3), 35 (2010)
9. Bunge, M.: Treatise on basic Philosophy. In: Ontology I: The Furniture of the World, vol. 3. Reidel, Boston (1977)

10. Wand, Y., Weber, R.: On the Ontological Expressiveness of Information Systems Analysis and Design Grammars. Journal of Information Systems **3**, 217–237 (1993)
11. Soffer, P., Yehezkel, T.: A state-based context-aware declarative process model. In: Halpin, T., Nurcan, S., Krogstie, J., Soffer, P., Proper, E., Schmidt, R., Bider, I. (eds.) BPMDS 2011 and EMMSAD 2011. LNBIP, vol. 81, pp. 148–162. Springer, Heidelberg (2011)
12. Tan, W., Fan, Y.: Dynamic workflow model fragmentation for distributed execution. Computers in Industry **58**(5), 381–391 (2007)
13. Reijers, H.A., Mendling, J., Dijkman, R.M.: Human and automatic modularizations of process models to enhance their comprehension. Inf. Syst. **36**(5), 881–897 (2011)
14. Reijers, H.A., Mendling, J.: Modularity in process models: review and effects. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 20–35. Springer, Heidelberg (2008)
15. Pourmaoumi, A., Kahani, M., Bagheri, E., Asadi, M.: Mining common morphological fragments from process event logs. In: CASCON, (2014)
16. Leymann, F.: Workflows make objects really useful. EMISA Forum **6**(1), 90–99 (1996)
17. Basu, A., Blanning, R.: Synthesis and decomposition of processes in organizations. Information Systems Research **14**(4), 337–355 (2003)
18. Vanhatalo, J., Volzer, H., Leymann, F.: Faster and more focused control-flow analysis for business process models through sese decomposition. In: Proceedings of the 5th International Conference on Service-Oriented Computing, Vienna, Austria (2007)
19. Vanhatalo, J., Völzer, H., Koehler, J.: The Refined Process Structure Tree. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 100–115. Springer, Heidelberg (2008)
20. Seidita, V., Cossentino, M., Hilaire, V., Gaud, N., Galland, S., Koukam, A., Gaglio, S.: The meta model: a starting point for design processes construction. International Journal of Software Engineering and Knowledge Engineering **20**(4), 575–608 (2010)
21. Eberle, H., Unger, T., Leymann, F.: Process fragments. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2009, Part I. LNCS, vol. 5870, pp. 398–405. Springer, Heidelberg (2009)
22. Seidita, V., Cossentino, M., Chella, A.: A proposal of process fragment definition and Documentation. In: Cossentino, M., Kaisers, M., Tuyls, K., Weiss, G. (eds.) EUMAS 2011. LNCS, vol. 7541, pp. 221–237. Springer, Heidelberg (2012)
23. Zemni, M.-A., Hadj-Anouane, N.B., Yeddes, M.: An approach for producing privacy-aware reusable business process fragments. In: Proceedings of the 2012 IEEE 19th International Conference on Web Services, p. 659-661, June 24-29, 2012
24. Assy, N., Chan, N.N., Gaaloul, W.: Assisting business process design with configurable process fragments. In: IEEE SCC, pp. 535–542, (2013)
25. Hens, P., Snoeck, M., De Backer, M., Poels, G.: Process Fragmentation, Distribution and Execution Using an Event-Based Interaction Scheme. J. Syst. Softw. **89**, 170–192 (2014)
26. Khalaf, R., Kopp, O., Leymann, F.: Maintaining data dependencies across BPEL process fragments. International Journal of Cooperative Information Systems **17**, 259–282 (2008)
27. Schumm, D., Leymann, F., Ma, Z., Scheibler, T., Strauch, S.: Integrating compliance into business processes: process fragments as reusable compliance controls. In: Proc. of the MKWI 2010 - Integriertes ERM in automatisierten Geschäftsprozessen (2010)
28. Asadi, M., Gasevic, D., Wand, Y., Hatala, M.: Deriving variability patterns in software product lines by ontological considerations. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012 Main Conference 2012. LNCS, vol. 7532, pp. 397–408. Springer, Heidelberg (2012)