ABSTRACT

Business processes have become essentials on internal operations within organizations. However, these require efficient storage mechanisms that allow their extraction and reuse. This paper proposes a method of storing and retrieval business process models, formally represented as graphs and whose search is based on control flow patterns.

Keywords
Business process, graph representation, control flow patterns.

1. INTRODUCTION

Use of business processes to describe internal operations within organizations has been an alternative widely adopted in recent years. The creation of specialized software tools called BPMS (Business Process Management Systems) [1] has enabled the modeling of business processes using diagrams that contain the tasks, messages and execution flows between such operations, making the administration and organization of these processes agile and efficient. Processes created from these tools have generated the need to store and maintain this information in repositories that allow reuse them. An example of this need is exposed in [2] which use a repository that contains about 500 process models consumed by local councils from Netherlands government. However, this number is small compared with the thousands of processes that could have a multinational company.

The large number of processes created by these modeling tools generates the need of a major degree at information organization, and here is where different authors are making efforts to identify efficient methods of storing and retrieval in order to manage this amount of processes.

By other hand, it is stressed that many of these efforts are targeted around SOA, which is an architecture that defines the encapsulation, reuse and composition of services providing a great interoperability between their components [4]. Despite the inherent advantages of SOA, it does not provide tools for automatic reuse of processes, for this reason it is necessary to create a repository that solves this kind of issues. This paper attempts to address this limitation by defining a storing and retrieval method of business processes models supported by a formal representation based on graphs.

The paper is organized as follows: Section 2 presents the repository of business process models (BP models) proposed, and the storing and retrieval method used. Section 3 describes related works. And Section 4 focuses on the conclusions and future works.

2. BP MODEL REPOSITORY

This section addresses BP models publication and querying procedures. The first one adds a new BP model in the repository, in order to increase the number of stored BP models and the second one, makes a search in the repository from a query BP model to obtain a set of BP models ordered according to control flow patterns detected on query BP model taken as reference.

2.1 Architecture

The reference architecture of present work (Figure 1) is defined based on the modular description presented in [5] and it contains five general layers: Presentation Layer, BP Parser, BP Management Layer, Repository Management Layer and Storage Layer.

2.1.1 Presentation Layer

This layer contains a web interface for publishing and querying a business process model.

2.1.2 BP Parser

This layer converts a BP model in a formal model represented by graphs. It receives as input a BP model described in WSML (Web Service Modeling Language) which contains a set of tasks, messages and execution flows. The output delivers a graph (BP graph) in which the tasks and events are represented as nodes, gateways as nodes type AND Split, Join; OR Split, Join; XOR Split, Join; together with all their nested tasks and events. (This article works on the eleven gateways defined by the BPMO 1.4 standard [6]). In addition, control flow connectors are represented as edges and block patterns WHILE and REPEAT are represented in the same way as gateways (Figure 2).
2.1.3 BP Management Layer
This layer is composed by the modules: management, analysis and similarity search of patterns. Below, we describe each one of them.

2.1.3.1 Pattern Management
This module specifies a set of reference patterns that obtain aspects related to control flow dependencies between diverse tasks. These patterns are defined in [7] and are described using graphs according to notation used in BP Parser module and can be edited according to user preferences (Figure 3).

Consider two graphs $G$ and $G'$; a graph isomorphism between $G$ and $G'$ is a bijective mapping $f: V \rightarrow V'$ such that $\alpha(v) = \alpha'(f(v)) \forall v \in V$. For any edge $e = (u, v) \in E$ exists an edge $e' = (f(u), f(v)) \in E'$ such that $\beta(e) = \beta'(e')$ and for any edge $e' = (u', v') \in E'$ exists an edge $e = (f^{-1}(u'), f^{-1}(v')) \in E$ such that $\beta(e) = \beta'(e')$. If $f: V \rightarrow V'$ is an isomorphism between graphs $G$ and $G'$, and $G'$ is a subgraph of another graph $G''$, i.e. $G \subset G'$, then $f$ is called a subgraph isomorphism from $G \subset G'$.

Patterns detection is done by a representation of a directed graph, considering the control flow of tasks and events within the process, to achieve this was necessary to use an algorithm called VF2 based on the work presented in [8].

2.1.3.2 Pattern Analyzer
This module is responsible for an exact detecting of the reference patterns contained in the Query Graph (BP Graph) that represents the business process query (BP Query Graph), see (Figure 3). In this case, patterns are represented by nested substructures between nodes AND, OR, XOR Split, Join. Below, we describe the concept in which is based the detection of substructures:

Patterns detection is done by a representation of a directed graph, considering the control flow of tasks and events within the process, to achieve this was necessary to use an algorithm called VF2 based on the work presented in [8].

2.1.3.3 Similarity Pattern Search
In order to not discard BP models from repository that are similar to BP query, this module makes an inexact subgraphs detection using the language GLIDE (Graph Linear DEscriptor) [9]. The expressions in Glide, called graph regular expressions, allow the description of portions of graphs, and approximate query, and let make various approximate queries identifying different locations of the control flow patterns inside BP Graph analyzed.
Figure 4 shows the approximate detection of substructures nested between nodes AND, OR, XOR Split.Join. The BP Query Graph is composed of a sequence that contains three control flow patterns:

1. AND_Split, Task, WebServiceTask, AND_Join (1)
2. XOR_Join, XOR_Split, GoalTask (2)
3. XOR_Join, WebServiceTask, XOR_Split (3)

These patterns can be described independently through GLIDE as follows:

(1) AND_Split%1/
   Task/AND_Join/WebServiceTask
   %1/
(2) XOR_Join%1/
   XOR_Split/GoalTask
   %1/
(3) XOR_Join%1/
   WebServiceTask/XOR_SPLIT
   %1/

Where nodes are expressed only with their labels and are joined by a '/' character representing the edges. The branches are grouped using nested parentheses and cycles can be viewed as a cutting edge and a label with an integer number. The nodes of cutting edges are represented by their labels, followed by the characters '%', '/', and the integer number. If the same node belongs to several cutting edges, the node label will be preceded by a list of '%' and the corresponding integer numbers.

According to above, a user can structure an exact query (1) (2) (3), using the language GLIDE as follows:

```
AND_Split%1/
Task/AND_Join(Task)
   XOR_Join%2/
   XOR_Split(XOR_Join)
      WebServiceTask/XOR_SPLIT
%3/
   )
   /GoalTask%2/
   )
/WebServiceTask%1/
```

Or an inexact query (2)...(1), where there is at least one route between the patterns (2) and (1). This route is expressed through the character '*':

```
XOR_Join%1/
XOR_Split(AND_Split%2/
   Task/AND_Join/WebServiceTask%2/
   )
/GoalTask%1/
```

Another kind of queries can be performed using wildcards as: '*', matches any single node, '*' matches any sequence of nodes, '?' matches at most one node, '+' matches any sequence of one or more nodes [9]. We can see that according to the sequence of substructures is possible to get an exact similarity result and also an inexact similarity result with respect to BP graphs stored in the repository.

### 2.1.4 Repository Management Layer
This layer contains drivers required to enable the creation, read, update and delete (CRUD) in each one of the repositories. In BP models (WSML documents) were used the WSMO and WSMO4J APIs. In BP Graphs was used GraphBlast [10] and for RDBMS were used drivers such as ODBC or JDBC.

![Figure 4. Similarity Pattern Search Module Operation](image)

### 2.1.5 Storage Layer
This layer contains three repositories. The first one known as ORDI [11] is responsible for storing WSML documents, this works together with WSMO and WSMO4J APIs. The second one called Berkeley DB [12] stores the BP Graphs and is directly related to GraphBlast. And the third one uses a relational database which contains the WSML documents location routes, and relates every document to every BP graph represented in order to make agile the delivery of similarity results to user, this consideration was taken into account from the recommendations described in [3]. In addition, is stored all data concerning to patterns detection of a given BP model, such as the number and type of identified patterns.

### 3. RELATED WORKS
Below are shown related works for processes storing in a repository. In [13], is described BPL (Business Process Library) as a result of the first design and prototype of the business process library of SUPER project (Semantics Utilized for Process management within and between Enterprises). BPL provides basic operations of a database management system (CRUD) and also allows semantic queries making inference on the information contained in the database through an ontological reasoning machine. In [14] is introduced an OASIS repository called ebXML (Electronic Business using eXtensible Markup Language) Registry / Repository. This allows store any data type including: descriptions of Web services, documents, binary data and XML data, queries are made through XML or SQL. In [15], is presented the IBM BPEL Repository which is a repository that stores BPEL business process and XML documents. Allows querying XML files as objects EMF (Eclipse Modeling Framework) using an
Moreover, given that this work represents the BP Models using graphs, this, in order to reduce the problem of processes recovering in a graphs recovering problem, next we present the related works from the perspective of graph searching and mining. Authors such as [17], [18] focus on graphs recovering based on mining frequent graph patterns. In this case a subgraph is frequent if its support (occurrence frequency) in a given data set is not less than a minimum support threshold. These techniques are based on depth first search (DFS) algorithms and substructure patterns decomposition. In [19] are described constraint-based graph pattern mining techniques. These constraints reduce the search space and can be classified into different categories.

The authors of [9, 10, 20, 21], define graph indexing methods, in which, given a graph database and a query graph, find all graphs which contain this query graph. Indexing mechanisms of substructures are used of a graph query, excluding all graphs that do not contain these substructures. Searching process is performed through an isomorphism algorithm and taking topics related to graphs searching in a graphs database viewed from two perspectives: i) Graph Search: Finds all graphs containing the query graph and ii) Graph Containment Search: Finds all graphs contained by query graph. These two techniques allow exact and inexact detection of predefined patterns within the query graph through substructures graphs searching.

From the description of previous works were selected for our proposed repository, the methods, techniques and researches defined in [8] for the matching algorithm, [9] for the method of inexact patterns searching and [10] as a mechanism for exact patterns searching.

4. CONCLUSIONS
This article introduces a method of storing and retrieving business process models supported in a formal representation based on graphs. Firstly, was addressed the need of using repositories for automatic processes reuse in SOA, then was described the proposed architecture for the BP Repository. The problem of processes recovering was reduced to a subgraph isomorphism detection problem using as a pattern detection technique GraphBlast and GLIDE as a query language for approximate querying results. Besides, is proposed a Pattern Analyzer in which the graph model representation (BP Graph) of the business process model (BP Model) is indexed in order to improve business process model searching using indexing techniques. The next step of this work is to further personalize the searching process using semantic discovery, which refers to inferences about the concepts on which the user could make his search, through applying ontological criteria that include relations, properties, functions and constraints and propose as a future work the develop of control flow patterns ontology in order to improve these discovery techniques.

5. ACKNOWLEDGMENTS
We want to thank to Dr. Rosalba Giugno from University of Catania, for provide GraphBlast and for her unconditional help and support.

6. REFERENCES

[17] X. Yan and J. Han. 2002. gSpan: Graph-Based Substructure Pattern Mining, ICDM'02.
[18] X. Yan and J. Han. 2003. CloseGraph: Mining Closed Frequent Graph Patterns, KDD'03.
[19] F. Zhu, X. Yan, J. Han, and P. S. Yu. 2007. gPrune: A Constraint Pushing Framework for Graph Pattern Mining, PAKDD'07.