Abstract: In this paper, we propose a novel approach that applies structural data mining to the problem of managing a large BPM collection in order to (1) retrieve similar business processes based on meta-level semantics given a context of new process definitions and (2) discover process patterns (and anti-patterns) prevalent in the organization’s practice when a domain reference model is being constructed. In the suggested approach, the semantics of BPM specifications are preserved in a conceptual graph format that comprise vertices and edges, along with their instantiations and interrelationships. A structural data mining tool is then applied to deal with a certain degree of informality inherent in business process models while looking at similar sequence patterns based on similarities in sequences of events, processes, and control structures. Our structural matching approach can complement other methods like classification based on descriptors or similarity measures with attributes of business processes. To validate and demonstrate the feasibility and effectiveness of our approach, we conducted a series of experiments with business model collections in a financial service company. The results show that with a modest effort of tuning search process, our approach can be a promising solution for dealing with large process models.

Keywords: business process models, process reuse, structural data mining, process repository

1. Introduction

As business process models have been constructed and accumulated over time, organizations now face a new challenge of managing a huge volume of business process model collections. It is natural that reuse of business process models, either in instance level or in reference model level, becomes common in practice in order to support such management activities although it often relies on human recollection. System engineers or subject matter experts recall business process models that were previously constructed in a domain similar to their current project. They search relevant business process models from their memories or a repository of archived documents, and then apply retrieved models to the current context. Reusing business process models, akin to other analysis and design artifacts in systems development, can bring various benefits to organizations. Reusable business process models can facilitate communications between system engineers and clients instead of starting from scratch, thereby help organizations capture correct business process requirements and identify possible improvements. In this way, they can expedite the process of business process management to quickly respond to changing business environments. In addition, organizations may obtain best-practice business processes because reusable process models have been already validated and successfully integrated in a similar domain.

The most pressing challenge in managing reuse of business models is its inherent complexity stemmed from an overwhelming number of variations of the same semantics (i.e., naming difference, different level of abstraction and granularity, various formalisms). As solutions to these problems, managing business process model collections appears to have redrawn attention from researchers in recent years. It is mainly because of standardization efforts for business process modeling languages and shifted interests toward Web-based, SOA (Service-oriented architecture) business applications. Yet most studies related
to management of process model collections place their focus on utilizing “bigger chunks” of business processes like process templates, domain reference models, or interoperable services coupled with business processes, with little attention to how to construct these chunks by finding reusable business processes based on similarities among actual activity and control sequences.

In this paper, we propose a novel approach that applies structural data mining to the problem of managing a large BPM collection in order to retrieve similar business processes based on meta-level semantics given a context of new process definitions when a domain reference model is being constructed. In the suggested approach, the semantics of BPM specifications such as Petri-net, BPMN, EPC, and UML Activity Diagrams (e.g., event, task, sub-process, gateway, sequence, message, and data object) are preserved in a conceptual graph format that comprise vertices and edges, along with their instantiations and interrelationships. Since the process patterns may use synonyms, e.g. clients, customers, consumer, etc., the synonyms are also defined in a conceptual graph. A structural data mining tool is then applied to deal with a certain degree of informality inherent in business process models while looking at similar sequence patterns based on similarities in sequences of events, processes, and control structures. Our structural matching approach can complement other methods like classification based on descriptors or similarity measures with attributes of business processes.

The paper is organized as follows. Section 2 reviews related research including process repository and process reuse. Section 3 explains structural matching techniques. Section 4 describes our experiment results. Finally, Section 5 concludes the paper with a discussion of the result.

2. Related work

Process reuse issue has been studied in many areas in BPM research. Many commercial tools and academic works provide some reuse support for business processes based on reusable asset management or knowledge management perspective. Major ERP vendors provide BPM tools such as SAP’s NetWeaver and Oracle’s Oracle Workflow, in which customers utilize workflow or business scenario templates and process patterns. To manage large collections of business process, there are a handful of research works in process repository. MIT process handbook is a well-known example of process repository that contains a comprehensive library of business process models [4]. Several commercial BPM tools support manual reuse of process models in process repositories [2]. To effectively manage process repository, the process reuse issue should be addressed. To facilitate reuse of process models, it provides several functions such as searching, browsing, and sorting of process models.

3. Structural matching

This section explains an approach to reuse of business process models based on structural matching techniques. A graph-based representation of structural information combined with a substructure discovery technique has been successfully applied in knowledge discovery area [2, 6, and 7]. In our paper, to find reusable process fragments in process repository, we used Subdue algorithm developed by [2]. Section 3.1 explains the Subdue algorithm in detail.

Process models can be expressed in several ways such as BPMN, EPC, or UML Activity diagram notations, etc. Since those process models are extended forms of directed graph, they are easily transformed in to directed conceptual graphs that consist of vertices and edges. In Section 3.2, an example of the transformation of process models will be explained.

When the models are transformed into directed conceptual graphs, they are added into a business process repository along with the initial models. As an analyst begins defining a business process model or tries to find applications of a certain pattern, the analyst can utilize the Subdue algorithm to search for
similar structures in the library. The analyst then selects the most relevant business process model and adapts it to the current analysis problem.

3.1 Structural Data Mining

Subdue is an automated relational learner that discovers patterns in structured data sets [2 and 5]. In Subdue, information is stored as a graph of vertices and edges. They are defined as follows.

Definition 1 (Graph(G)) A graph is a pair \((V,E)\):
- \(V\) is a finite set of vertices, where \(\langle v \text{id label} \rangle \in V\)
- \(E\) is a finite set of edges
- \(E = E_u \cup E_d\), where \(E_u\) is a set of pairs of vertices (undirected edges) and \(E_d\) is a set of ordered pairs of vertices (directed edges).

Vertices usually refer to objects, attributes, and their values while edges represent relationships between the objects. The syntax for vertex description is \(\langle v \text{id label} \rangle\) where \(v\) is the vertex number and \(label\) is the name of that vertex. Edges are coded with \(\langle u \text{id1 id2 label} \rangle\) or \(\langle d \text{id1 id2 label} \rangle\). The former represents an undirected edge \((u)\) between vertex \(id1\) and \(id2\); the latter means a directed edge \((d)\) from vertex \(id1\) to \(id2\). Examples of the Subdue graph are shown in Fig.1 and Table 1.

Subdue’s search algorithm figures out repetitive substructures called concepts in graphs [2]. The algorithm begins with initializing the search queue with a uniquely labeled vertex of a graph. Following the beam search strategy, it expands its search by including adjacent edges and associated vertexes in all possible ways, yielding potential substructures. When a repeating substructure is found, it is replaced with a placeholder vertex pointer to its substructure, thereby compressing the whole graph. Each candidate substructure is evaluated by a compression score based on the Minimum Description Length (MDL) principle. The basic idea of the principle is the best concept (substructure) describes the whole data set with a minimal description length, i.e. the length in number of bits of the graph representation when compressed by the substructure [1]. When the compression ability of a candidate substructure is better than others, the structure is preserved in the best substructure queue. Iterating this process results in a hierarchical classification lattice whose lower-level concepts are included in the higher-level concepts. The iteration can be limited by two parameters: breadth of search (beam) and number of expansions (limit). The search terminates when it reaches a user specified limit on the number of substructures extended or when the search space is exhausted.

In Subdue, a threshold value determines when two structures are similar enough to match. The analyst can set this threshold parameter from 0.0 to 1.0. The value 0.0 means a complete match and 1.0 the maximum tolerance level. The similarity metric of two structures is computed as transformation cost / structure size, where transformation cost is the number of graph transformations required to make the structures isomorphic. The transformations are adding or deleting either an edge or a vertex, changing a label on an edge or a vertex, and reversing the direction of an edge. A transformation has a cost of 1. Two structures are identified as similar ones when the similarity value is less than the threshold [5].

To use the capability that deals with synonyms, a list of predefined synonyms should be defined beforehand. They can substitute different vertex or edge labels that carry the same meaning. This functionality enables Subdue’s potential to utilize the benefits of a domain ontology or lexicon.

3.2 Representation of Business Process Models

As described in the previous section, business process models need to be represented as directed conceptual graphs in order for the structural data mining algorithm to find a similar match. The
transformation of business process models into conceptual graphs takes place at two different abstraction levels: metamodel and instance level. Since key elements in metamodel types in BPMN, EPC models, and UML activity diagrams share similar semantics, converted process graphs can be used together regardless of differences in the modeling notations. In this paper, we consider an example of EPC models which are used in SAP R/3 and ARIS system. EPC chain can be defined as follows.

Definition 2 (Event-driven process chain) An event-driven process chain is a five-tuple \((E,F,L,T,A)\):
- \(E\) is a finite set of events,
- \(F\) is a finite set of functions,
- \(L\) is a finite set of logical connectors,
- \(T \in \{\text{AND, XOR, OR}\}\) is a function which maps each connector onto a connector type,
- \(A\) is a set of arcs.

The building blocks of an EPC are functions, events, and logical connectors. Functions are relevant to activities which need to be executed. Event refers to the situation where functions start or end. Logical connectors are used to connect functions and events. Fig.1 (a) shows an example of EPC model [3]. In the example, boxes refer to functions, hexagons are events, and circles represent logical connectors.

To convert EPC models into Subdue graph, we define the transformation coding scheme for the EPC building blocks. Table 1 summarizes the representation scheme for EPC notations. Note that, the coding scheme is designed, considering (1) separation between metamodel elements and instances and (2) maintaining an atomic value for each vertex and edge, the coding scheme for EPC notations is designed.

<table>
<thead>
<tr>
<th>Table 1. Transformation of EPC into Subdue Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPC Semantics</td>
</tr>
<tr>
<td>Function</td>
</tr>
<tr>
<td>(V 1\ Function)</td>
</tr>
<tr>
<td>(V 2\ Handle)</td>
</tr>
<tr>
<td>(V 3\ order)</td>
</tr>
<tr>
<td>Event</td>
</tr>
<tr>
<td>(V 1\ Event)</td>
</tr>
<tr>
<td>(V 2\ Receive)</td>
</tr>
<tr>
<td>(V 3\ order)</td>
</tr>
</tbody>
</table>

In our scheme, the metamodel elements are coded with vertices in a conceptual graph. Each vertex has a label like Event, Function, AND-Connector, XOR-Connector, OR-Connector. The connecting objects such as sequence flows and connectors (AND, XOR, and OR) are coded as edges between two vertices that represent these metamodel elements. Then a function instance has two vertices describing the nature of the function with a verb and an object. They are linked with ActionOf and ActionObjectFor edges respectively. Event instances are coded in a similar fashion. They are linked with TypeOf and ActionObjectFor edges respectively.

Fig.1 (b) illustrates a detailed example of the transformation in which the model in Fig.1 (a) is converted. It shows the translated vertices and edges in the conceptual graph format. In the figure, the gray-highlighted part represents the structural information of the process, and the white-colored elements are the instance-level information.
3. Evaluation: Case Study

To demonstrate the feasibility of our approach, a preliminary case study was taken with a relatively small process model library. The process models in the library were collected from the specification documents and examples available on related Web sites. The majority of them were constructed for SMEs’ processes and from industries such as finance, retail, construction, and manufacturing. The library used in the case study consists of 122 process models in simple flowcharts or EPC notation. To use in the structural mining, eight people independently coded the models into subdue graphs. It is intended to reflect a more realistic situation in which process models are authored by different people at different times. As a result, the library is transformed into a conceptual graph that contains more than 3000 vertices and 4200 edges. Then we constructed seven queries, partially completed process models that would be run against the library. For the illustration purpose, the queries were drawn from the existing library and modified as follows.

1. Variations in notations with the same semantics: It is not uncommon to see a process model library is composed of models in various notations. To check if our approach can assist in finding similar cases in different notations, we modified simple flowchart models to EPC-based ones and vice versa.

2. Variations in labels in activities with the same semantics and notations: Naming difference is one of the main reasons that make retrieval of similar cases more complicated. For this, we modified some of the labels in the original models.

3. Variations in labels in activities with the same semantics and notations given a dictionary of synonyms: A possible solution to process pattern mining is to utilized ontology. In Subdue, a similar feature can be provided with a pre-defined set of lexicons. In addition to (2) queries, we tested usability of lexicons by providing synonym lists.

Table 2 summarizes the results of the preliminary case study. For each query, the threshold value is initially set as 0.0 and then incremented by 0.1 until 0.9. In each trial, relevant, best matched graphs were retrieved at the threshold between 0.4 and 0.9, expect the most complicated query “Handle a customer complaint” that has more than 25 vertices and 30 edges. The queries that test variations in notations required a relatively high threshold values, 0.7–0.9. Compared to this, other queries were found at 0.4–0.8. The results suggest that structural matching technique can be applied to find relevant, reusable business process models. Yet, in order for the tool to be practical, each search must be tuned with a proper threshold. It should be also noted that for complicated queries with a high threshold, the computation time may exceed more than several minutes in a personal computer CPU environment because the algorithm itself is polynomial. This concern can be resolved by adjusting other search options such as limit, beam, number of vertices in a structure, etc.
Table 2. Case study results

<table>
<thead>
<tr>
<th>Variations in notation</th>
<th>Query</th>
<th>Complexity of query</th>
<th>Best match found at threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Open a new bank account</td>
<td>High</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td>Process a refund</td>
<td>Low</td>
<td>0.7</td>
</tr>
<tr>
<td>Naming difference</td>
<td>Cancel a bank account</td>
<td>Medium</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>Handle a customer complaint</td>
<td>High</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>Process a withdrawal</td>
<td>Low</td>
<td>0.6</td>
</tr>
<tr>
<td>Naming differences with synonyms</td>
<td>Generate an invoice</td>
<td>High</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>Process a deposit</td>
<td>Low</td>
<td>0.4</td>
</tr>
</tbody>
</table>

4 Conclusion

This research proposes structural data mining approach to business process reuse that exploits rich semantics of business process modeling formalisms. Business process models are translated as conceptual graphs that comprise vertices and edges. The coding scheme is quite flexible and extensible; it can express core semantics of existing business process specifications. By applying the structural matching technique, the approach can deal with a certain degree of informality inherent in business process models while looking at similar sequence patterns. This study can be applied to many reuse-related situations, namely retrieval of reusable process models given a problem, uncovering sequence patterns among process models, and suggesting the instances of (anti-) patterns for learning purpose.

Future work includes developing the prototype of the tool support based on our approach, validating its effectiveness in a field or lab experiment setting. Additional methods for search tuning also need to be explored to increase search performance, including ontological support of important concept matching and Subdue’s supervised learning feature.

Acknowledgement

This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government (NRF-2011-0010561).

References