A REFERENCE ARCHITECTURE FOR MANAGING BUSINESS PROCESS VARIANTS

Ruopeng Lu and Shazia Sadiq
School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, Australia

Keywords: Business process management, flexible workflows, process evolution, business process analysis.

Abstract: Business process management systems (BPMS) have been prevalent in business information systems, yet still striving to cope with emerging demands from current business environments. It is particularly challenging in managing knowledge intensive business processes, which has partially led to the demand for more complex BPMS functionality such as instance adaptation and streamlined process evolution. On the other hand, various process analysis and discovery techniques have been developed as an important component in BPMS. In this paper, we present a technology framework that supports process discovery from preferred work practices in a flexible process management system. The framework supports instance adaptation and a systematic approach towards process evolution/improvement.

1 INTRODUCTION

In recent years, there have been many efforts towards providing agile business process management (BPM) support. Business process management systems (BPMS) have been recognised as a substantial extension to the legacy of workflow management systems (WFMS). The BPM life cycle (van der Aalst et al, 2003b) identifies that apart from process **design**, **deployment** and **enactment**, supporting process **diagnosis** differentiates BPMS from traditional WFMS. The process diagnosis phase which refers to a wide range of BPM activities, has been the prime arena for both academia and industry, especially for business process analysis (BPA) and process discovery (van der Aalst et al, 2003b; Casati, 2005). Generally, these post-execution activities are to identify and resolve operational process problems, discover preferred work practices, and provide business intelligence.

Nevertheless, in the dynamic environment of collaborative and e-business today, it is essential that BPM technology supports the business to adapt to changing conditions, where different process models could be derived from existing ones to tailor individual process instances. It is evident that work practices at the operational level are often diverse, incorporating the creativity and individualism of knowledge workers and potentially contributing to the organization's competitive advantage. This diversity needs to be both encouraged and controlled. A major difficulty in this regard is the requisite knowledge, that drives the diverse practices at an operational level, is only tacitly available. This knowledge constitutes the corporate skill base and is found in the experiences and practices of individual workers, who are domain experts in a particular aspect of the overall operations.

There have been proposals from both academia (Sadiq & Orlowska, 2005; van der Aalst et al, 2005; Rinderle & Reichert, 2006) and industry (Ultimus, 2004, ILOG, 2006, Tibco, 2006) to transfer process modelling efforts from business owners and business analysts, to domain experts who have the knowledge in performing activities in the process. The common practice in these proposals has been the support for deploying partial process models, which contain some repetitive procedures that require less or no flexibility for execution, but also contain loosely coupled process activities that warrant a high level of customization. When an instance of such process is instantiated, a complete process model is specified by the domain expert, where the set of loosely-coupled activities is given an execution plan according to instance-specific conditions, possibly some invariant process constraints, and their expertise. We shall refer to such approaches as **instance adaptation**.

We believe that current BPM solutions only provide limited capability for instance adaptation.

Lu R. and Sadiq S. (2007).
A REFERENCE ARCHITECTURE FOR MANAGING BUSINESS PROCESS VARIANTS.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - ISAS*, pages 416-421
DOI: 10.5220/0002400304160421

Techniques are required where part of the modelling effort is transferred to domain experts who make design decisions at runtime.

At the same time, instance adaptation also imposes new problems and challenges on existing techniques for process diagnosis activities. During instance adaptation, the process model of each process instance may be uniquely designed. An executed process instance reflects a variant of realising the same process goals, and is called a **process variant**. One of many challenging problems is how to avail the knowledge from a diversity of process variants as an information resource. This can be considered as a particular area of **process discovery**.

Process discovery often refers to so-called best practices, or the best way to perform a particular type of business process (O'Leary & Selfridge, 2000). The measures of best practices are manifold. For example, from the resource utilization perspective, a process variant is regarded as the best practice if the least number of performers are required. While in some other perspective, the process model with the largest number of instances can be the best since it is the most trusted. These measures correspond to business metrics (Casati, 2005). The way to capture, manage and diffuse knowledge associated with the best practices can be found in many knowledge-based systems.

The contribution of this paper is to present a reference architecture that is in accordance with these new demands, for the retainment and retrieval of process variants from a flexible business process modelling and execution framework that supports instance adaptation.

The rest of this paper is as follows. Section 2 presents a brief overview on the proposed framework supporting instance adaptation and process discovery. In section 3, a reference architecture for managing process variants is introduced, and its major components and functionality are presented in section 4. The related work is reviewed in section 5, followed by the conclusion in section 6.

2 FRAMEWORK OVERVIEW

We utilize a framework for business process modelling and execution that attempts to achieve the balance between flexibility and control (Lu et al, 2006a). The framework consists of two major components: (1) A constraint-based process modelling approach, called Business Process Constraint Network (**BPCN**); and (2) a repository for case specific process models, called Process Variant Repository (**PVR**).

In BPCN, business process requirements are extracted and transformed into a minimal set of process constraints, which are represented in a way that is readable by human and supporting analysis and validation for correctness. Instance adaptation takes place when a customised process model for process instance is constructed (or completed) at runtime (Lu et al, 2006a). The BPCN execution environment allows for the generation of potentially a large number of customized process variants. Figure 1 shows the process models of four different process variants, all of which satisfies the same set of process constraints, namely, task T1 must perform before tasks T2, T5 and T6.

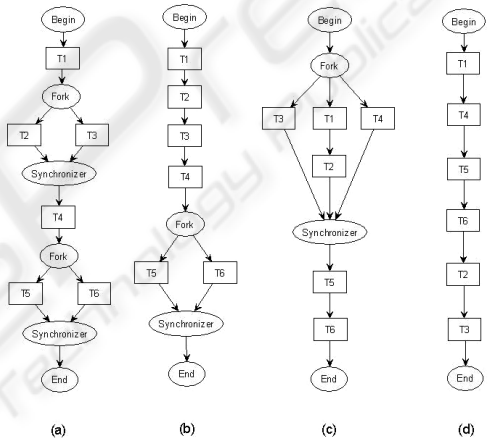

Figure 1: Examples for process models of process variants satisfying the same process constraints.

Although all these process variants achieve the same process goals, they may vary significantly in many aspects, such as process models and resource allocation patterns. It is important to note that the way that domain experts reason about the situation during process modelling cannot be truly reconstructed using computational techniques.

In our approach, the purpose of PVR is to capture, structure and subsequently extract the decisions that led to a particular design as far as possible. These design decisions are embedded in various process properties e.g., process data values. Over time, the repository can build into an immense corporate resource.

In the meantime, there are various occasions when precedents of process variants need to be retrieved. For example, during instance adaptation, domain experts may refer to a list of past process variants sharing common properties similar to

current situation. Using appropriate analysis techniques, a collection of sufficiently similar process variants can be generalized as the preferred/successful work practice, and consequently lead to process improvement/evolution. The retrieval process in PVR according to a user specified requirement is supported.

3 PROCESS VARIANT REPOSITORY

PVR provides a well-formed structure to store past process designs, as well as an instrument to utilize process variants as an information resource. The **retainment** of executed process variants in the repository and the subsequent **retrieval** of preferred process variants are the two major functions of PVR.

Figure 2 presents an overview of the PVR reference architecture. In what follows, we present an overview of various functions to manage PVR in stepwise order (as annotated in Figure 2). The first three steps correspond to the retainment process, while steps 4 – 6 correspond to the retrieval process.

Step1: An executed process variant is received from BPCN through the interface between PVR and BPCN. The process variant is to be retained in the repository according to the schema in PVR, and is referred to as a **case** of the business process. A set of **features** are extracted from the case in order to formulate a formal case **description** that can identify the new case (cf. Section 4). It is possible for a set of similar cases to share the same description.

Step2: The **feature index** is updated according to the description of the new case. The feature index in PVR (Lu et al, 2006b) is an organised structure of case descriptions, and is maintained by the index management component.

Step3: The new case is stored in the repository. This step builds the PVR population and with time is expected to build into a valuable corporate resource.

Step4: At a later time, a **query** is formulated to specify the case retrieval requirement. A query expresses user requirements in terms of case features. A query may represent a partial or complete description for a case, or multiple cases sharing same description. The query requirement is formulated with the help of the query processing component.

Step5: Case descriptions are searched utilizing the feature index to find matching cases according to query requirements. The goal is to return a set of sufficiently **similar** cases. Similarity between case descriptions and the query requirements is determined by a predefined similarity measure (cf. Section 4).

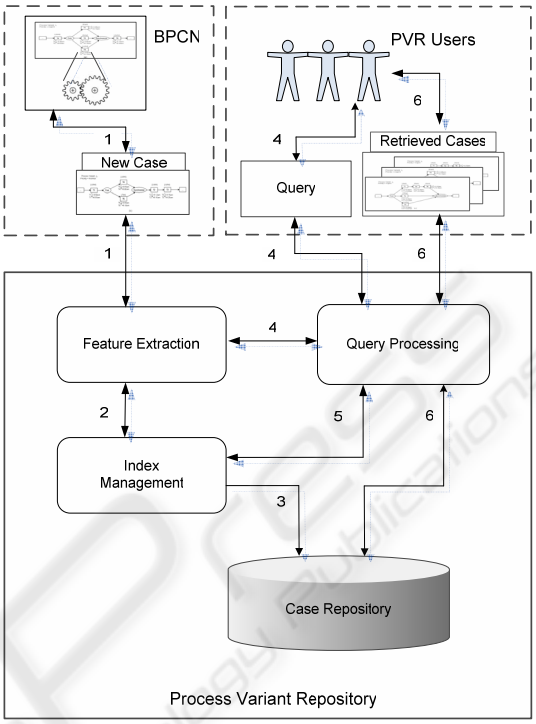

Figure 2: Reference Architecture for PVR.

Step6: The initial matching cases are retrieved from the repository, and the best matches are selected from the set of initial matches according to the degree of similarity compared to the query case, called the **similarity score**. The further selection process involves a more detail comparison, e.g., comparing the process model of a case against the query process model (if query requirements contain structural features).

4 PVR FEATURES AND COMPONENTS

At any time, the repository may be queried for precedents under specified criteria. In this section, we discuss the major components and functionality of the PVR reference architecture that support the above requirements.

Case Schema

A process variant is a complex object containing various design and executional properties. The case **schema** defines the structure and data content according to which process variants are stored. In PVR, a feature is an attribute-value pair used to

describe the property of a case. Dimension refers to the attribute part of the feature. We have identified the following dimensions of features that the schema should cover:

- **Structural** dimension contains the customized process model (typically graphical), which is customized for the process variant during instance adaptation;
- **Behavioural** dimension contains execution properties of the process variant, such as temporal properties of tasks involved in the process execution, the performers and their roles of process tasks, and the values of process-relevant data for the variant;
- **Contextual** dimension contains descriptive information (annotations) from the process modeller about the reasoning behind the design of a particular process variant, its goals and intentions. There is evidence that semantic technologies (description logic, ontologies etc.) may assist in the formalization of the contextual dimension.

Feature Index

In PVR, the structured collection of case descriptions reflects the concretized knowledge that led to the designs of different cases during instance adaptation. The feature index is used to differentiate different cases. Furthermore, as the number of cases can be potentially very large, the index also facilitates effective case retrieval in subsequence.

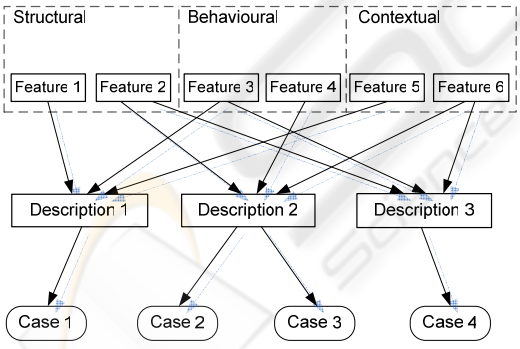

Figure 3: PVR feature index structure.

PVR uses a three-layer hierarchy to manage the feature index, as illustrated in Figure 3. The set of unique features collected from all retained cases are on the top layer, where each feature points to one or more case descriptions. A case description contains a minimal list of features required to identify a particular case, which typically covers all three dimensions. A case description points to one or more cases, where a case is the collection of all required features of a process variant according to the case schema.

The process of inserting new cases and adjusting the index structure constitutes the repository update approach. PVR update determines the way the case is indexed by extracting features from the case and computing the index values. The description of the new case is prepared by extracting relative information form the process variant according to a preference list of features designed by domain experts, which covers the three dimensions of features. A detail presentation of the feature index management approach can be found in (Lu et al, 2006b).

Query Formulation

Query is the interface between users and PVR, through which a wide range of search criteria for case retrieval is specified. Examples can be features in behavioural and contextual dimensions such as to find all cases with *execution duration of not more than 3 days*, or *performers of the role senior management were involved*. Such queries can mostly be satisfied using well established techniques. We are specifically interested in providing a facility to find process variants for queries that provide complex (structural) criteria. For example, find all case in which *activities T2 was performed immediately after T1, and T3 was performed in parallel with T1 and T2* (cf. Figure 4(b)).

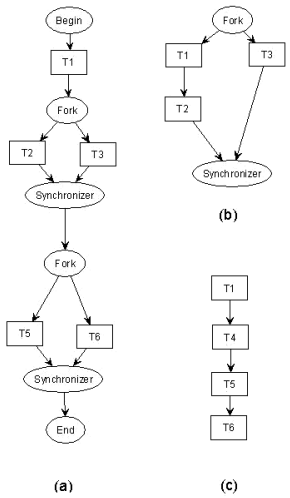

Figure 4: Queries containing (a) a complete process model and (b) (c) partial process models.

Furthermore, a hybrid of features in different dimensions (or so-called multi-aspect queries) also can be provided, e.g., find all cases where *activities T1, T2 and T3 were performed by a senior management role in sequence*, and *finished*

execution within 1 - 2 days. The issue here is how to characterize the structural features of process variants such that queries that require searching the repository on the structural aspect can be efficiently satisfied. The structural features of the query can resemble a complete process model (cf. Figure 4(a)), which specifies the exact structure required for the cases to be retrieved; or a partial process model (cf. Figure 4(b), (c)), which contains a fragment of the process model characterizing the desired structural properties to be retrieved.

Case Similarity

An essential concept for case management is the **similarity measure** for the cases to be compared, i.e., how to determine the degree of match between two cases, or between a case in repository and a desired case described in a query. The similarity measure provides quantitative measurement to calculate how similar two cases are.

Recall that each case is characterised by a selection of features in its case description, representing the factors that lead to the design of particular process variant. As a result, the case matching problem is transferred into the comparison of case features in each dimension. However, different types of features (in different dimensions) generally have specific semantics, the similarity measure for each dimension is thus to be defined separately with respect to its semantic appropriateness. The overall approach is:

- Define the similarity measure for each feature type in every dimension, i.e., for each type of feature, define a function which takes as input a pair of features to be compared, and outputs a similarity score between 0 (dissimilar) and 1 (complete match);
- Calculate the overall similarity by aggregating the similarity score from each dimension. A weighted approach can be used if some dimension is prioritized by assigning a higher weight than the others.

Consider comparing features in structural dimension as an example. If the process model in Figure 4(a) (referred to as 4(a)) is to be compared with those in Figure 1. 1(a) is the most similar to 4(a) since in 1(a) all structural constraints between the tasks in 4(a) are preserved and thus having the highest similarity score. 1(a) is said to be **subsuming** 4(a) since it contains more tasks than 4(a). If it is not required to have the same set of tasks in both models, the structural similarity score is 1 between 1(a) and 4(a). In addition, 1(a) is called a **complete match** to 4(a).

On the other hand, 1(b) is a **partial match** to 4(a) for partially preserving process constraints between the tasks in 4(a). To elaborate further, in 4(a), tasks T2 and T3 are in parallel branches and can be executed in any order. While in 1(b), T3 can only be executed after T2. As a result, 1(b) is a more restrictive model than 4(a), which excludes certain execution possibilities from 4(a), and hence is less similar to 4(a) than 1(a). For the same reason, 1(d) is a partial match to 4(a) and the similarity rank to 4(a) is lower than 1(b) since there is only one execution possibility in 1(d). Notably however, 1(c) is also a partial match to 4(a) but does not preserve the structural constraint between tasks T1, T2, T3 and T4, while containing extra execution possibilities.

Specifically, we have developed a methodology (Lu & Sadiq, 2006) for comparing both exact and partial matching cases based on complex control flow features, where a similarity function has been defined to produce the similarity score between 0 and 1.

Case Retrieval

In our framework, a progressive refinement approach is used for case retrieval (Steps 4 - 6 in Figure 2 is repeated). Given a query of case features (typically in multi-aspect queries), the retrieval process starts with searching for matching cases by using a subset of required features (e.g., structural features). Relevant features in the feature index are searched and compared with the query features, and a set of initial matching cases is retrieved from the repository including complete and partial match cases, as directed by the feature index. For partial match cases, we use a ranking mechanism to provide a quantitative measure for the similarity. A predefined threshold value by domain expert is used to filter the best matches (e.g., similarity score $\geq$ 0.75). A more restrictive search is then performed, where the query case and the best matches from the set of initial match cases are further compared with the remaining query features. This process can be repeated until the preferred case is selected. PVR users can be involved in this process, which is to decide which features to be used as comparison and in what order.

We take the structural matching as an example to illustrate the retrieval process. Suppose a given query contains the structural features in Figure 4(c) and requires to retrieve exact matching cases (i.e., threshold value = 1.0). A set of cases, in which the process models contain tasks T1, T4, T5 and T6 are first retrieved. Cases containing process model 1(d) (cf. Figure 1) are complete matches to the query case and are retrieved for further comparison, since process model 1(d) subsumes 4(c). Cases contain

process models 1(a), 1(b) or 1(c) are partial matches to the query case and hence not to be retrieved.

5 RELATED WORK

Business process analysis (BPA) involves monitoring and analysis of process execution patterns and performance. Process mining (van der Aalst, 2003a) is one such approach, which is to diagnose operational processes by extracting information from process execution logs. The proposed PVR reference architecture is different from process mining approach, with emphasis on supporting knowledge acquisition and process discovery in BPM. In particular, it targets the reuse of past instances of process execution to achieve new operational goals in similar situations.

Some approaches in existing literature such as Case-Based Reasoning (CBR) can be relevant to address the challenges in PVR management that we have found to link closely to our problem. There are a number of recent proposals for process management approaches based on CBR techniques (Madhusudan et al, 2004; Weber et al, 2005), which have demonstrated the possibilities to apply CBR techniques to achieve certain BPM goals.

6 CONCLUSIONS

Variations in work practice often represent the competitive differentiation within enterprise operations. In this paper, we have argued for the value of variants in BPM platforms. We have presented a reference architecture as a foundation for designing and implementing a process variant repository (PVR) that addresses advanced requirements for instance adaptation, process redesign and post-execution analysis. The presented components in PVR provide effective means of searching and matching process variants against a given query, and generate result sets that can be conveniently ranked. The results of the proposed reference architecture can provide deep insights into ongoing work practices, identify areas of process improvement, and contribute to systematic and well-informed process evolution.

REFERENCES

van der Aalst, W. M. P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M., 2003a. Workflow Mining: A Survey of Issues and Approaches. *Data & Knowledge Engineering,* vol. 47, pp. 237 – 267.

van der Aalst, W. M. P., t. Hofstede, A. H. M., Weske, M., 2003b. Business Process Management: A Survey. *In BPM 2003, Conference on Business Process Management*, Eindhoven, The Netherlands.

van der Aalst, W. M. P., Weske, M., 2005. Case handling: a new paradigm for business process support. *Data & Knowledge Engineering*, vol. 53, no. 2, pp. 129-162.

Casati, F., 2005. Industry Trends in Business Process Management: Getting Ready for Prime Time. *In DEXA 2005, 16th International Workshop on Database and Expert Systems Applications*, Copenhagen, Denmark.

ILOG 2006, Business Rule Management (BRM) with ILOG JRules 6, BRMS Without Compromise. White Paper, ILOG, www.ilog.com

Lu, R., Sadiq, S., Padmanabhan, V., Governatori, G., 2006a. Using a Temporal Constraint Network for increased flexibility in Business Process Execution. *In ADC 2006, Seventeenth Australasian Database Conference*, Hobart, Australia.

Lu, R., Sadiq, S., Governatori, G., 2006b. Utilizing Successful Work Practice for Business Process Evolution. *In BIS 2006, 9th International Conference on Business Information Systems*, Klagenfurt, Austria.

Lu, R., Sadiq, S., 2006. Managing Process Variants as an Information Resource. *In BPM 2006, 4th International Conference on Business Process Management*. Vienna, Austria, September 2006.

O'Leary, D., Selfridge, P., 2000. Knowledge Management for Best Practices. *Communication of ACM*, vol.43, no. 11.

Madhusudan, T., Zhao, L., Marshall, B., 2004. A Case-Based Reasoning Framework for Workflow Model Management. *Data Knowledge Engineering*, vol.50(1), 87-115.

Rinderle, S., Reichert, M., 2006, Data-Driven Process Control and Exception Handling in Process Management Systems. *In CAiSE 2006, 18th International Conference on Advanced Information Systems Engineering*, pp. 273-287.

Sadiq, S., Sadiq, W., Orlowska, M., 2005. A Framework for Constraint Specification and Validation in Flexible Workflows. *Information Systems*, vol.30, no.5, pp. 349-378.

Tibco 2006, Enhancing BPM with a Business Rule Engine. Tibco White Paper, Tibco, www.tibco.com

Ultimus 2004, Adaptive Discovery: Accelerating the Deployment and Adaptation of Automated Business Processes. White Paper, Ultimus Inc. www.ultimus.com

Weber, B., Rinderle, S., Wild. W., Reichert, M., 2005. CCBR-Driven Business Process Evolution. *In ICCBR 2005, 6th International Conference on Case-Based Reasoning*. Chicago, USA.