Enabling effective workflow model reuse: A data-centric approach

Zhiyong Liu a, Shaokun Fan b,*, Harry Jiannan Wang c, J. Leon Zhao d

a Faculty of Management and Economics, Dalian University of Technology, China
b College of Business, Oregon State University, United States
c Lerner College of Business and Economics, University of Delaware, United States
d Department of Information Systems, College of Business, City University of Hong Kong, China

ABSTRACT

With increasingly widespread adoption of workflow technology as a standard solution to business process management (BPM), much effort is focused on designing appropriate workflow models that satisfy the business requirements for a given company. Workflow models are designed to support complex process management in many business domains such as supply chain management, knowledge management and e-commerce [1,2]. How to efficiently design workflow models to satisfy particular business requirements is one of the key factors of BPM success [3].

Workflow models are used to capture domain knowledge about business processes in organizations. Many modeling methods have been proposed and applied by both academic researchers and business practitioners, such as Petri nets [4], UML activity diagrams [5], metagraphs [6] and the dataflow-based approach [7]. The dataflow-based approach is a relatively new method, in which designers develop workflow models by analyzing the input and output data of tasks and their dependency relationships [7]. Compared with other workflow modeling methods that require users to provide plenty of information about specific processes, such as description of business function, sequence of tasks and structured representation of business rules, dataflow-based modeling methods focus on the input and output data in the business processes. Thus, it is easier to gather information required for the dataflow-based modeling method, because input and output data items are usually contained in the documents that business users deal with on a daily basis.

When designing workflow models for a specific business context, workflow designers always confront with the difficulty of collecting domain knowledge [8]. A typical business process involves many functional departments of an organization, such as purchasing, production, and sales, and requires knowledge from these varied business domains. However, process designers usually possess expertise in modeling, system design or software engineering, but little business domain knowledge like sales or production. Process designers always devote considerable effort to collecting information from workers at the front line. Even though domain knowledge can be acquired from interviews, surveys and field studies, there is always a significant gap between designers’ and users’ understanding, which is caused by their different knowledge backgrounds.

Rather than starting from scratch, designers can gain valuable insights from the existing process records [9], which can serve as a valuable source of domain knowledge. To a certain degree, the existing models reflect the operational mechanisms of an organization. A formal approach is needed to utilize the knowledge embedded in existing workflow models in a systematic manner. Model reuse will certainly

1. Introduction

With increasingly widespread adoption of workflow technology as a standard solution to business process management (BPM), much effort is focused on designing appropriate workflow models that satisfy the business requirements for a given company. Workflow models are designed to support complex process management in many business domains such as supply chain management, knowledge management and e-commerce [1,2]. How to efficiently design workflow models to satisfy particular business requirements is one of the key factors of BPM success [3].

Workflow models are used to capture domain knowledge about business processes in organizations. Many modeling methods have been proposed and applied by both academic researchers and business practitioners, such as Petri nets [4], UML activity diagrams [5], metagraphs [6] and the dataflow-based approach [7]. The dataflow-based approach is a relatively new method, in which designers develop workflow models by analyzing the input and output data of tasks and their dependency relationships [7]. Compared with other workflow modeling methods that require users to provide plenty of information about specific processes, such as description of business function,

* Corresponding author at: 302 Austin Hall, Oregon State University, Corvallis, OR 97331, United States.
E-mail address: Shaokun.fan@OregonState.edu (S. Fan).

http://dx.doi.org/10.1016/j.dss.2016.09.002
0167-9236/© 2016 Elsevier B.V. All rights reserved.
improve the quality and efficiency of business process design. Business practitioners have documented many workflow models in various business domains based on their knowledge and experiences, called Process Reference Models (PRMs), such as SAP Process Reference Model and Oracle Best Practice Processes [10]. So far, workflow model reuse mainly relies on function or control structure. However, little research has been done on workflow model reuse based on the data dependency, which takes the data flow of workflow models as the focus of model management.

We adopt a data-centric perspective for workflow model reuse for three reasons. First, data flow information can be directly acquired from users. Data collection is the starting point of business requirement collection when building workflow models. The users are usually familiar with the data that they are using and producing on a daily basis. By contrast, the control flow information that reflects complex logic is not explicitly documented in working environments; it is summarized by users based on their domain knowledge. Second, dataflow of workflow models is more intuitive and less error-prone than control flow [11]. The control flow-based models require integrating designers’ in-depth understanding of model semantics and logic. Even the error rate of control flow models designed by professional companies is reported to be more than 10% [11]. Third, the data flow information is independent from workflow modeling paradigms and supports cross-paradigm model reuse. The solutions for workflow model reuse based on the control flow perspective are vitally influenced by the model specification paradigms [12,13]. The control flow-centric models from different sources can be expressed following different paradigms such as Petri nets [14], UML activity diagram [5] and BPMN. Considerable effort is spent on transforming control flow models and adjusting model management algorithms. Workflow models with different paradigms can be managed in a unified way following a data-centric perspective.

In this paper, we fill the research gap by proposing a framework for data-centric workflow model reuse (DWMR), which consists of model storage, model search and model composition via a data-centric approach. Our contributions are as follows. First, we define the formal data structure of a workflow model repository and propose a data-centric indexing method for workflow models based on data dependency relationships. Second, we propose a method for matching user requirements with candidate workflow models based on data similarity and develop a flooding algorithm to search for model groups that satisfy a particular user query. Third, we develop a formal method to compose candidate groups of models when needed to meet a specific user requirement. We evaluate the approach by applying it to a business case and comparing the DWMR approach with existing approaches and human experts.

2. Literature review

2.1. Workflow modeling

The control flow perspective of workflow modeling focuses on the sequence and coordination of tasks. The objectives of control flow modeling are controlling, monitoring, optimizing and supporting business processes [14]. Petri nets have become one of the most popular control flow-based modeling paradigms since van der Aalst [14] introduced it into the workflow modeling field. In addition to providing an appropriate language for workflow specification, Petri nets also serve as a powerful analytical tool for verification of workflow models. Several important concepts have been introduced to support the analysis of the correctness of control flow models, such as structuredness [15,16] and soundness [17]. Vanderfeesten et al. [3] introduced cohesion and coupling metrics to evaluate workflow model design. Van der Aalst has extended Petri nets to propose a workflow net concept and the YAWL language [18]. Some extended forms of Petri nets are also proposed to add more data information to workflow models, including colored Petri nets [19] and timed Petri nets [20].

Most of the modeling methods from the control flow perspective or graph-based perspective ignore the importance of data flow. They just focus on the coordination of tasks or rules, while the data requirements of models, data exchange between tasks and data dependency relationships are often overlooked or simplified.

2.2. Data flow perspective

The data flow perspective was first proposed in the software engineering field [21,22] and more recently introduced into the business process modeling field to detect data errors and correct workflow models [23,24]. Compared with traditional modeling methods that focus on the structural issues [17], data flow is considered as an important complement to the workflow specification. Sadiq et al. [24] identify the importance of data flow issues to workflow research and introduce modeling [25], specification and validation of data flow. They also illustrate the essential requirements for data flow modeling and seven types of data anomalies. Russell et al. [26] describe a series of workflow data patterns to show the different ways to represent and use data flow in workflow systems. Data flow analysis is a useful tool for workflow verification, which has been used to validate the correctness of control flows. Sun et al. [7] propose a formal data flow specification and develop algorithms to detect data anomalies based on data dependency. The data flow modeling method is further extended to support process integration [27]. Vanderfeesten et al. [2] introduced a product-based workflow design approach, which adopts a Product Data Model to capture the data dependency relationships. A formal workflow language based on Petri nets and nested relational calculus has been proposed to support data flow modeling [28].

2.3. Workflow model reuse

Research on management and reuse of models has a long history in fields such as software engineering [29,30] and operation research [31]. Existing models form an abundant source of domain knowledge and best practices. In the BPM field, they are referred to as Process Reference Models (PRMs) [32]. Many commercial companies have provided packages of their process designs and best practice for different industries, including SAP Process Reference Models and Oracle Best Practice Processes [10]. Reference Models help model designers avoid starting from scratch. These packages cover many business functions that can satisfy most of a company’s basic requirements. Because of the unique characteristics of different companies, Process Reference Models must be adapted to fit a particular context before being adopted [33]. Models with a higher abstract level are more generalizable, and more effort is needed to adapt them to meet the specific requirements of the end user.

Researchers have studied workflow model reuse from different angles. Wang and Wu [10] propose a collaborative approach to integrate and manage process reference models (PRMs). Their approach is based on a “spiral” model for organizational knowledge creation and leverages Web 2.0 technologies such as social tagging and classification to maintain PRMs. The model patterns (or model components) are also reusable. Zhuge [34] defines workflow components with four characteristics: independence, encapsulation, completeness and consistency. Thom et al. [35] classify workflow patterns into nine categories, and Altintas et al. [36] abstract workflows into several components according to their service functions. Cao et al. [37] define the workflow components, which consist of function, quality of service, control flow model and business domain.

The model design cases are also a reusable resource. Madhusudan et al. [38] propose a framework to reuse two categories of workflow cases, i.e., prototypical cases and instance cases. They also introduce a similarity-based flooding algorithm to support case retrieval. Similarity-based model matching is another important aspect of model reuse. Madhusudan et al. [38] support case retrieval by matching the NAME properties of nodes in models through language processing and string
matching techniques to get the initial similarities. Zhuge [39] calculates the matching degree of two activities with activity-distance. The similarity degree of processes is calculated using matching degree of activities, numbers of activities and numbers of sub-processes of the processes. Their research revealed the wide existence of reusable process units and tested the thresholds for similarity-based search.

Although previous research on workflow model reuse is fruitful, little research has explicitly considered the data-flow perspective, which is a unique feature of this paper. The main difference between our research and existing research are shown in Table 1. The relative advantages of our approach include: (1) utilizing the data flow information, which is a valuable resource and often ignored by existing research; (2) lowering the requirements for users to input requirements; and (3) supporting the search and composition of model groups.

3. Data-centric workflow model reuse (DWMR)

3.1. DWMR framework

We propose a framework for data-centric workflow model reuse, shown in Fig. 1.

3.1.1. Workflow model storage and organization

When a new workflow model is adopted by the company, the model should be organized and stored in the model repository. All models in the model repository should be encoded following a standard format.

3.1.2. Workflow model search

A data-centric search mechanism provides users with the most relevant models from the model repository to aid their model design work. The relevance of a workflow model is calculated based on the fit between models in the repository and user requirements.

3.1.3. Workflow model composition

When a group of models is found relevant to user requirements, the composition of these models should be presented to designers. The composed workflow model will be obtained by mapping the composed data dependency structure.

3.2. An illustrative example

We will illustrate the DWMR approach using two simplified banking processes. A process model repository has been built to store the existing models to facilitate new workflow model design. The two example processes are shown in Figs. 2 and 3 using extended UML activity diagrams as defined in [7].

Transfer application

After receiving a transfer application form, the bank first checks to make sure that all needed information is provided. If the application form is not complete, additional information will be requested. Then it verifies the application to ensure that the information is accurate. After the application form is verified, it will be formally accepted by the bank. Then the password from the customer is checked before allowing the customer to access the account balance. If there is enough balance, the transfer will be executed and the process is finished.

Credit payment application

After transaction information and authentication information are received from the client, the bank accepts the credit payment application. Then the customer’s credit history record and liquid assets record are checked to create a credit score. The transaction information and credit score are analyzed to give an overall evaluation such as risk level and amount adjustment. Finally, the application with the evaluation result is approved by the officer.

A new workflow model of the Loan Application process needs to be designed, which begins with receiving an application and ends with that application being approved. The requirement can be considered as modeling a process that produces the data entity of an approved application form with an input data entity of the initial application form. Rather than starting from scratch, we can reuse the two aforementioned models because the new Loan Application process is similar to the existing models of credit assessment and approval. If no individual model can fulfill the requirements, composing a group of candidate models may be required.

3.3. Model storage and organization

Business data is one of the most essential components of a business process. The actual input data required by a task includes not only the data entities, but also their status. Data entity represents the information object that can be manipulated by agents, such as “Application Form” and “Credit Score.” Note that the data entity in this paper refers to a data item or a workflow form, which is different with the data entity concept in database [43]. Status data reflects the state that the entity has reached. For example, an “Application Form” that is initially filled out by a customer is different from an “Application Form” that is approved by the bank in the example in Section 3.2. Status data reflects the lifecycle of a data entity in a whole business process. According to Workflow Management Coalition’s definition [44], “control data is very important for business process modeling and it is described as “state information about each workflow instance and state information about each activity instance” [44]. Thus, we include status information in data definition, because status data is always considered as a part of workflow data.

Definition 1. (Data) A data d in workflow model is defined as a 2-tuple, consisting of data entity and status, namely \( d = (\text{Data entity, status}) \). Example: data \( d = (\text{Application form, verified}) \) means “an application form which has been verified.”

Definition 2. (Workflow model) A workflow model (denoted as W) in the repository consists of both control flow information and data flow information. The control flow structure consists of the task set and the set of flows between tasks, while the dataflow structure describes a set of relationships between tasks and data, which contains tasks and

<table>
<thead>
<tr>
<th>Table 1: Comparison with existing approaches.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reusable units</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>Control flow based approaches</td>
</tr>
<tr>
<td>DWMR</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
their input and output data. Formally:

\[ W = \{ \text{Model name WN: Model description MD, Control flow structure } \Gamma \} \]

\[ = \{ (T, F); \text{Data set } D = \{ d_i | i = 1, 2, \ldots, n \}; \text{Task–data relationship } TD = \{ (t, D_{in-t}, D_{out-t}) | t \in T \} \} \]

WN is the unique identification of this model. Model description records the information provided by model designers to describe the function of this model. \( \Gamma \) is the control flow structure, which consists of tasks and the flows between them. \( T \) is a finite set of tasks in \( W \), \( T = \{ t_i | i = 1, 2, \ldots, m \} \), \( t \) is a specific task in \( T \) (\( t \in T \)). \( F \) is a finite set of the flows between tasks in \( T, F = U (t_i \times t_j), t_i \in T, t_j \in T, i \neq j \). The data items that are used or produced in this model form a data set \( D \). TD describes the relationship between input/output data and tasks; \( D_{in-t} \) and \( D_{out-t} \) are the sets of input and output data of task \( t \).

We have developed a formal method to deal with loops in workflow model. The main idea of this solution is to transform a cyclic model to a set of acyclic models, denoted as main path and feedback paths respectively. We introduce the method as follows:

Input: cyclic workflow models; output: a set of acyclic models, which are transformed from the main path and feedback paths of the cyclic models.

Preconditions: (1) All the input workflow models should be structured [17], which promises the errors caused by structure including deadlock and multiple instances are avoided. This is not a strong restriction, because most of the reusable models have been tested in application. Structuredness is a common requirement for most workflow modeling tools. (2) All the models must start with a start node and end with an end node. Every other node must be on a path from the start node to the end node. This is also a simple and common requirement, which ensures that the models are connected [14].

The cyclic model transformation algorithm is as follows. Step 1, omit all of the Split/Join nodes, and transform the models \( (M) \) to directed graphs \( (G) \) which consist of task nodes \( (N) \) and arcs. Step 2, search for a random path \( (P) \) from the start node \( (s) \) to the end node \( (e) \). Set a label with a number for every node on the path. The number for \( s \) starts with 0, and every immediate successor node in the path increases its
I search for a random path from Join nodes in the original model, and get the directed graph. Second, we work model is acyclic. Step 5, mapping the main path and feedback paths to the main path, and the remaining paths in the graph as feedback paths. Finally, we give mappings of the main path and feedback path as work models by adding the Split/Join nodes to the corresponding positions.

An illustrative example is shown in Fig. 4. First, we omit all the Split/Join nodes in the original model, and get the directed graph. Second, we search for a random path from s to e, namely P: s → A → E → F → I → J → D → e. A number is labeled for every node in P, from 0 to 7. Third, search for the paths (P’) that start from a node labeled with a smaller number and end with a node labeled with a bigger number. Add P’ to P. P’ includes A → B → F, F → G → J, A → B → C → G → J and A → B → C → D. Fourth, until no more P’ are found, denote the final P as the main path, and the remaining paths in the graph as feedback paths. Finally, we give mappings of the main path and feedback path as workflow models by adding the Split/Join nodes to the corresponding positions.

After dividing cyclic models to acyclic paths, we define rules to avoid the possible loops caused by the main path and feedback path pair (Fig. 5). After the model search and model composition stage, the feedback paths and the main path should be composed, to keep the model completeness.

Definition 3. (Data dependency relationship) In a workflow model, if a task produces data item d2 with data item d1 as input, then there is a direct data dependency relationship between d1 and d2. We say that d2 directly depends on d1, which is denoted as d1 → d2.

We should note that there is no transitivity for direct data dependency relationships. If d1 → d2 and d2 → d3, we cannot get the result d1 → d3. However, d1 is used to produce d2, and d2 is a precondition for producing d3. When d2 is temporarily unavailable, d1 may be a necessary input for the system to produce d3. The system will produce d2 with d1, and then produce d3 using d2. We call the relation between d1 and d3 an indirect data dependency relationship.

For a set of data items {d1} (i = 1, 2,...k), if d1 → d2...dn−1 → dn (n = 3, 4,...k), then there is an indirect data dependency relationship between d1 and dn. We define that dn indirectly depends on d1, denoted as d1 → dn. Transitivity is an important property of indirect data dependency relationships, namely if d1 → d2 and d2 → d3, then d1 → d3.

Definition 4. (Data dependency distance) Data dependency distance between data items d1 and d2 is defined as the length of the shortest path of data dependency relationships that generate B from d1. Given a direct data dependency relationship d1 → d2, the data dependency distance between d1 and d2 is defined as 1, which is denoted as Distance(d1, d2) = 1. The distance between a node and itself is defined as 0, Distance(d1, d1) = 0. For an indirect data dependency relationship d1 → d2, the minimum number of direct data dependency relationships

![Fig. 4. Example of transforming cyclic workflow model.](image-url)
that constitute this indirect data dependency relationship is defined as the data dependency distance between $d_1$ and $d_6$. The data dependency distance between $d_1$ and $d_2$ is determined by the shortest indirect data dependency relationship, namely $\text{Distance}(d_1, d_2) = \min(m, n)$. If there is not any data dependency relationship between $d_1$ and $d_2$, we define $\text{Distance}(d_1, d_2) = +\infty$.

In Fig. 5, $\text{Distance}(d_1, d_2) = 1$, because between $d_1$ and $d_2$ there is a direct dependency relationship. There are three paths from $d_1$ to $d_6$, where $\text{Length}(d_1 \rightarrow d_2 \rightarrow d_4) = 2$, $\text{Length}(d_1 \rightarrow d_4) = 1$, $\text{Length}(d_1 \rightarrow d_3 \rightarrow d_4) = 2$. The distance between $d_1$ and $d_6$ should be the minimum length of all the paths, then $\text{Distance}(d_1, d_6) = \min(2, 1, 2) = 1$.

We define the distance between two models as the minimum distance between two data items in these two models respectively, namely $\text{Distance}(M_1, M_2) = \min(\text{Distance}(A_i, B_j))$, $A_i \in M_1, i = 1, 2, \ldots, n$, and $B_j \in M_2, j = 1, 2, \ldots, m$. The distance of one data set is defined as the maximum distance between any two data items in this data set, namely $\text{Distance}(M) = \max(\text{Distance}(A_i, A_j) \mid \text{Distance}(A_i, A_j) < +\infty, A_i, A_j \in M, i \neq j, 1, 2, \ldots, m)$. The conditional distance of a data item $d$ is defined as the maximum distance of any two data items in $M$ that are involved in a given data dependency relationship set ($C$), denoted as $\text{Distance}_c(M, C)$, $C$ is a set of data dependency relationships, $C = \{(d_{m_{ij}} \rightarrow d_{out_{ij}})\}$. $d_{in} = \bigcup d_{in_i}$ and $d_{out} = \bigcup d_{out_{ij}}, i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$. Namely, $\text{Distance}_c(M) = \max(\text{Distance}(A_i, A_j), A_i, A_j \in M, \text{Distance}(A_i, A_j) < +\infty, \text{Distance}(A_i, A_j) < +\infty, i \neq j, 1, 2, \ldots, m)$.

In the data structure in Fig. 6(a), let us define a model $M1:\{d_1, d_2, d_1\}$ and model $M2:\{d_5, d_6, d_2\}$. The distance between $M1$ and $M2$ should be the length of the shortest path between any two nodes from $M1$ and $M2$ respectively. Obviously, the shortest path should be "$d_2 \rightarrow d_4 \rightarrow d_5"$ and "$d_2 \rightarrow d_4 \rightarrow d_5"$. Then, $\text{Distance}(M_1, M_2) = 2$. The distance of an individual model is defined as the length path existing in the model. In $M1$, the longest path should be $d_1 \rightarrow d_2$ and $d_1 \rightarrow d_3$, then $\text{Distance}(M1) = 1$.

**Definition 5.** (Data item description) Data item description consists of optional descriptions of the data item, dependency relationships and data-task relationships showing the tasks that use specific data as input or output. Formally:

$$\text{Data item} = \{\text{Data name WN}, \text{Model name WN}, \text{Data description DD}\}$$

- Direct dependency $DP_d = \{(d \rightarrow d_{i}) \mid d, d_{i} \in D, i = 1, 2, \ldots\}$
- Indirect dependency $DP_i = \{(d \rightarrow d_{i}) \mid d, d_{i} \in D, j = 1, 2, \ldots\}$
- Data-task relationship $DT = \{(d, T_1, T_n)\}$

$d$ is the identification of this data item. $W$ is the name of the model that $d$ belongs to. $DP_d$ and $DP_i$ are the set of direct and indirect data dependency relationships of $d$ respectively. $DT$ records the tasks related to this data item $d$. $T_1$ and $T_n$ are the sets of tasks that use $d$ as input and output respectively.

**Definition 6.** (Data dependency structure) Given a workflow model, a data dependency structure (DDS) is a graph structure in which nodes represent data and edges represent direct data dependency. DDS consists of data set and direct dependency between data, which is denoted as: $S = \{D, DP_d\}$.

For example, in the Transfer Application model, the data set $D = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8\}$, the direct dependency $DP_d = \{(d_1 \rightarrow d_2), (d_1 \rightarrow d_5), (d_2 \rightarrow d_4), (d_3 \rightarrow d_4), (d_4 \rightarrow d_5), (d_4 \rightarrow d_6), (d_5 \rightarrow d_7), (d_6 \rightarrow d_7), (d_7 \rightarrow d_8)\}$. We represent the data items with ovals and indicate the dependency relationship with arrows. The data dependency structure of this model is illustrated in Fig. 6(a). From this visual index of this model, we can intuitively identify the direct and indirect dependency relationships. The data dependency structure of the Credit Payment Application model is created similarly, as shown in Fig. 6(b).

**3.4. Model search**

We show a model search requirement template below. The user describes the possible input and output data for the required model. For the Loan Application process, the search requirement can be expressed as “(Loan Application form, initial) ↠ (Loan Application form, approved)”. The expected result could be an individual model that satisfies the user requirements or several models that collectively fulfill the requirements after appropriate composition. According to the different types of expected results, we further classify the data-centric model matching into Individual Model Matching and Group Model Matching.

**User requirement template for data-centric model search**

**User query:** “User input ↠ expected output”

**Expected result:** matched individual models or model groups.

Note that the users can require the dependency between input and output to follow a certain path, with a query like “User input ↠ intermediate data ↠ expected output”, simply denoted as “A ↠ B ↠ C”. For simplicity, we do not raise cases under this condition. The query will be decomposed as two sub-queries "A ↠ B" and "B ↠ C." The final result will be the composition of the models returned by the two sub-queries. The cyclic models are divided into a set of acyclic models in the model search stage, namely the main path and feedback paths. We propose two rules dealing with the main path and feedback paths. Rule 1: the main path and its feedback path(s) cannot be calculated in the same candidate model set during the model search process. Rule 2: if one of the paths of a model is involved in the final composed workflow
model, all the other paths should also be composed to keep the model complete.

3.4.1. Individual model matching

We show the user requirement template for individual model matching. Din (or Dout) is a set of data items the user requires to be the model’s input (or output), din and dout are data items, din ∈ Din, dout ∈ Dout. The user expects the target model to produce output data Dout using input data Din. If an individual model satisfies this requirement, then the dependency relationship “din ↠ dout” should exist in the data flow structure of this model.

User requirement template for individual model match

User query: Din ↠ Dout.

Expected result: individual model containing “din ↠ dout”, (din ∈ Din, dout ∈ Dout).

For example, the user requirement for the Loan Application process can be expressed as follows: Din = {(Loan Application form, initial)} and Dout = {(Loan Application form, approved)}. The data set similarity focuses on the proportion of the common data elements between the user’s data set and the model data set. The variable α (α ≥ 0) denotes the weight of the data set that includes the data items in the user requirements but not in the model. In practice, the variable α should be predefined. Different values can be assigned to α to search for the one leading to best performance, which is beyond the scope of this paper.

User requirement example of individual model match

User query: {(Loan Application form, initial)} ↠ {(Loan Application form, approved)}.

Expected result: W = {wi | “(Loan Application form, initial) → (Loan Application form, approved)” ∈ wi, DP, i = 1, 2, …}.

There may be more than one model that contains “din ↠ dout”. We measure the quality of the result models based on their similarities with user requirements. Extending the definition of similarity measure defined in [45], we define similarity as follows:

**Definition 7.** (Data set similarity) The data set described in the user requirements is denoted as DU(DU = Din ∪ Dout), the data set contained in a candidate model is denoted as DM. The data set similarity is defined as follows:

$$SIM_{Data} = \frac{|D_{In} \cap D_{Out}|}{|D_{In} \cap D_{Out}| + \alpha \cdot |D_{In} - D_{Out}|}$$

The data set similarity focuses on the proportion of the common data elements between the user’s data set and the model data set. The variable α (α ≥ 0) denotes the weight of the data set that includes the data items in the user requirements but not in the model. In practice, the variable α should be predefined. Different values can be assigned to α to search for the one leading to best performance, which is beyond the scope of this paper.

**Definition 8.** (Data dependency similarity) The data dependency relationship set that is described in the user requirements is denoted as DP, which is a set of dependencies like “din ↠ dout” (din ∈ Din, dout ∈ Dout). We can simply express the data dependency relationship set as “DP = Din ↠ Dout”. The data dependency similarity between a candidate model M and DP is defined as follows:

$$SIM_{dep} = \frac{Distance_{DP,M}(M)}{Distance_{DP,M}(M) + \beta \cdot (Distance(D_{in}, M) + Distance(M, D_{out}))}$$
Similar to the definition of data set similarity, data dependency similarity focuses on the common part between the data dependency relationship sets of user requirements and candidate models. The variable $\beta$ ($\beta \geq 0$) denotes the weight of the uncommon data dependency set and should be predefined in practice. Similar to $\alpha$, $\beta$ is assigned with an initial value of 1 in this study, and the most appropriate value could be obtained after experiments.

We evaluate the similarity between user requirements and candidate models from two aspects: data set and data dependency. The overall similarity should be a synthesis of the above two similarity metrics, which is defined next.

**Definition 9.** (Synthesized similarity) The synthesized similarity is the weighted sum of data set similarity and data dependency similarity:

$$SIM = \gamma \cdot SIM_{data} + (1-\gamma) \cdot SIM_{dep}$$

For $0 \leq \gamma \leq 1$, $\gamma$ and $1-\gamma$ represent the weights of $SIM_{data}$ and $SIM_{dep}$ respectively, and should be predefined. We assign 0.5 to $\gamma$ in practice, which means the importance of data set similarity and data dependency similarity is treated equally. In the Loan Application example, we assume that the candidate individual model is the Credit Payment Application model shown in Fig. 3. The user query is described. The common data set between user requirements and the model data set is $\{(\text{Application form, approved})\}$, and the uncommon data set is $\{(\text{Application form, initial})\}$. The distance between the model and $D_{in}$ and $D_{out}$ are 1 and 0 respectively. The distance of the model under the condition of user requirements is 4. The data dependency similarity between the model and user query is $SIM_{dep} = 0.8$ ($\beta = 1$). Finally, the synthesized similarity between user query and the candidate model is 0.65 ($SIM = 0.5 \times 0.5 + 0.8 \times 0.5 = 0.65, \gamma = 0.5$).

**3.4.2. Group model matching**

When individual model match returns no result or the similarities of returned models cannot reach the threshold value, we can conduct an additional search for model groups that may satisfy the requirements after composition. The user requirement for matching multiple models is described below.

**User query requirement template for group model match**

**User query:** $D_{in} \rightarrow D_{out}$.

**Expected result:** model groups containing “$d_{in} \rightarrow \ldots \rightarrow d_k \rightarrow \ldots \rightarrow d_{out}$”. (i.e., a model set $S = \{W_i \mid i = 1, 2, \ldots, n\}$, in which $W_i$ contains “$d_{in} \rightarrow d_{out}$”, Model $W_2$ contains “$d_{in} \rightarrow d_{out}$”, Model $W_n$ contains “$d_{in} \rightarrow d_{out}$”, $(d_{in} \in D_{in}; d_{out} \in D_{out})$).

As illustrated above, the user requirements can be decomposed to a certain level. After finding a model $W_1$ which contains $d_{out}$, and when the initial similarity between $W_1$ and user requirements exceeds a predefined threshold, we scan $W_1$ to get a data set (denoted as $D_{as}$) where all data items determine $d_{out}$. For every data item in this data set, e.g., $d_x$, the second model $W_2$ with a data item $d_{x-1}$ that satisfies “$d_{x-1} \rightarrow d_x$” can be derived. Then a new data set (denoted as $D_2$) can be found from $W_2$, in which all data items determine the previous data item ($d_x$). The above procedure can be repeated until $d_{as}$ appears in a new data set. Because the complexity of this procedure increases sharply when more models are added to the group, a model group size threshold should be predefined to limit the search space. When the number of models in a group exceeds the threshold and the requirement is still unsatisfied, the group should be abandoned with no result returned. We propose a formal algorithm for model searching, shown in Fig. 7.

**3.5. Model composition**

After identifying the candidate model groups, we need to compose the models. In the Loan Application Process design example, two candidate models are the main path of the Transfer Application model (Model I) and the Credit Payment Application model (Model II) shown in Figs. 2 and 3.

In this example, the user requirement is expressed as the data dependency relationship “{(Loan Application form, initial)} $\rightarrow$ ((Loan Application form, approved))”. It means that: by using the input of data item “(Loan Application form, initial)”, the composed model should produce data item “(Loan Application form, approved)” as output. In the individual model search phase, using the flooding algorithm, we cannot find any individual model that fulfills this user requirement. After group model search, the dependency relationship “(Transfer Application form, accepted) $\rightarrow$ (Transfer Application form, approved)” and “(Credit Payment Application form, initial) $\rightarrow$ (Credit Payment Application form, accepted)” in Model I and Model II are identified, where $d_a$ has the same data fields as $d_c$, i.e., “$d_a = d_c$.” The condition of building a model group of Model I and Model II is satisfied. The resulting model should be a composition of the two models that contain “$d_1 \rightarrow d_4$” and “$d_4 \rightarrow d_7$” respectively ($d_4 = d_c$).

**Definition 10.** (Linking node and linked node) Assume dependency “$d_i \rightarrow d_j$” ($i \neq j$) exists in model $W_1$ and dependency “$d_p \rightarrow d_q$” ($p \neq q$) exists in model $W_2$. If $d_i = d_p$, then the dependency “$d_i \rightarrow d_q$” can be achieved after composition of $W_1$ and $W_2$. We call $d_i$ and $d_p$ a Linking Node Pair. $d_j$ and $d_q$ are denoted as Linking Node and Linked Node respectively. For example, $d_i$ is called linking node and $d_j$ is called linked node in Fig. 8.

The Data-centric Model Composition algorithm contains three major steps:

Step 1 (Pruning): taking every linking node, e.g., $d_4$ in Model I in Fig. 8, and the user required output node, e.g., $d_7$ in Model II in Fig. 8, as the ending nodes, we search the nodes that are not included in the data dependency relationships of the ending nodes. Delete data items that do not determine the output nodes or the linking nodes, i.e., $d_5, d_6, d_7, d_8$, and the associated data dependency relationships. Delete data items that determine linked nodes, i.e., $d_5, d_6$, and the associated data dependency relationships. Delete the direct data dependency relationships that end with linked nodes.

Step 2 (Combination): after pruning the unnecessary data set and data dependency relationships, merge the linking node pair with the same data fields as a single node ($d_5$ and $d_7$). Replace the linking nodes and linked nodes in the dependency relationships with the merged node. The data dependency structure after combination is shown in Fig. 9.

Step 3 (Mapping): the composition of the data dependency structure helps us design a reference control flow model. We describe the mapping approach simply as follows: first, decide the relationship between any two tasks in the workflow model based on their data dependency relationship. For example, tasks $t_1$ and $t_2$ produce data $d_1$ and $d_2$ respectively. If $d_1 \rightarrow d_2$, then $t_1$ should be executed earlier than $t_2$, they should be in sequence relationship. If $d_1$ and $d_2$ have no dependency relationship, then $t_1$ and $t_2$ should be in parallel; second, because multiple data items can be produced by one task, different relationships can be detected between tasks; third, build the workflow model without parallel first; and finally, add parallel relationships to the model without interrupting the existing sequence relationships. Due to limited space, the formal mapping algorithm is not in the scope of this paper. Please refer to [46], which introduces a formal approach to building workflow models.
based on their data flow structure. After the model mapping, if a path of a workflow model is involved in the composed model, the other paths should be composed to keep the model complete.

We formulate the model composition algorithm in Fig. 10.

4. Case study

We evaluate our DWMR approach in two ways. First, we assess the feasibility of our approach by applying it to a business case. Second, we invite human experts and apply the existing control flow based approach to the same problem and compare those with the result generated by our approach. The performance of our approach will be evaluated by cross-comparison.

4.1. Case description

Business process designers are required to design a new process that illustrates budget planning for a company. This process begins with setting the annual business goals in the directors’ board meeting, continues across multiple departments and ends with determination of the budget in the financial department. It is extremely difficult for an external designer to design this process model because domain knowledge from multiple departments is required. Without sufficient domain knowledge, designing such a model is often difficult, time consuming and error prone.

In this case, Oracle Best Practice Processes are adopted as the model repository. We show how the data-centric model reuse approach can be used to aid process design by utilizing these existing models. Oracle Best Practice Processes is a package of more than 400 process reference models provided by Oracle. The models are all derived from past experiences and practices, and cover all business functions in an organization. This repository can be considered to be very similar to a private model repository of an organization.

We use our model storage approach to input these models into our repository. According to definition 5, we get the data dependency structure of the model “Defining business strategy”. Due to limited space, we only show an example model from our repository, which are referred to by the results of our approach and human experts (Fig. 11).

4.2. Results of DWMR approach

The user query is expressed as “(Business Goal Report, approved) → (Budget Report, approved)”. First we conduct Individual Model Matching. We show results of the top seven models with the highest similarity scores in Table 2. We set the thresholds for data similarity, data dependency similarity and
synthesized similarity as 0.1, 0.1 and 0.2 respectively. \( M_2, M_1 \) and \( M_3 \) are the results of individual model matching using DWMR approach. There is not any individual model that possesses a similarity of 1.0, which means the user requirements cannot be fulfilled by reusing only one existing model.

Next, since individual model matching cannot directly fulfill the user requirements, we conduct group model matching. As shown in Table 3, group 1 (\( M_1, M_2 \) and \( M_3 \)) has reached the similarity of 1.0, which is accepted as the candidate result. Although group 6 (\( M_1, M_2, M_3 \) and \( M_4 \)) has also reached the highest similarity, it is detected as a superset of

![Fig. 8. Prune the unnecessary data and dependency relationships.](image)

![Fig. 9. Composition of data dependency structures and workflow models.](image)
group 1, which means that the addition of M4 did not increase the similarity, and group 6 does not contain more useful information than group 1. Therefore, group 6 is rejected as a duplication of group 1. For the same reason, group 5 is rejected as a duplication of group 2. Group 4 (M1 and M3) possesses a high data similarity and it is rejected. This is because M1 and M3 do not have a common data set, which means that the two models are not connected and cannot be composed directly. It cannot serve as a candidate group for model composition. Using

Algorithm 2: Model Composition Algorithm

Input:
1. Model group set $S^1 = \{G_i\}$, where $G_i = \{M_{ij}\}, i,j = \{1,2,\ldots\}$ //Model groups and the models in the groups
2. Model set M, where $M = \bigcup G_i = \{M_k\}$, where $M_k = \{D_k, P_k\}$, data set $D_i$ and data dependency set $DP_i$. //The model set and the data set and data dependency set of every model
3. Output:
   a. Combined model set $S^2 = \{M_i\}$, where $i = \{1,2,\ldots\}$ //The set of the combined data flow models
   b. Combined workflow model set $W^2 = \{W_i\}$, where $i = \{1,2,\ldots\}$ //The set of the combined control flow models
4. for each model group $G_i \in S^1$
5.   for each model $M_j \in G_i$ where $D_i \cap D_j \neq \emptyset$ //Start with searching the models related to the data output of user requirements
6.     for each model $M_k \in G_i - M_j$
7.         if $D_i \cap D_k \neq \emptyset$ then //If the model is connected to the previous model
8.             define $D_i = D_i \cap D_k = \{d_1, d_2, \ldots, d_n\}$
9.             create a new model $M_{i-k} = \{D_i, DP_i\}$, where $D_i = D_i, DP_i = \emptyset$ //Combine the model with previous model to create a new model
10.        for each $d_j \in D_j$
11.           for each $d_k \in D_k$
12.             if $(d_j, d_k) \in DP_i$, then add $d_j$ and $d_k$ to $D_i$ and add $(d_j, d_k)$ to $DP_i$ end for
13.        end for
14.    end for //Update the data set and data dependency set
15.    add $M_k$ to $G_i$, delete $M_j$ and $M_k$ from $G_i$. //Update the model group with the combined model
16. if $G_i \cap M_i = \emptyset$, then add $M_i$ to $S^2$. //When all the models in group are combined, return final model
17. end for
18. end for
19. end for
20. for each model $M_j \in S^2$
21. define $W_j = mapping(M_j)$ //Mapping the data flow model to control flow model
22. add $W_j$ to $W^2$

Fig. 10. Model composition algorithm.

Fig. 11. Define business strategy (M1).

TASKS
1. Establish Business Goals;
2. Define Enterprise Performance Measure;
3. Formulate Planning Discipline

DATA ITEMS
1. $d_1$: (Business Goal Report, approved); $d_2$: (Enterprise Performance Report, recorded);
2. $d_3$: (Performance Measure Form, generated); $d_4$: (Planning Discipline Report, recorded)
First, they needed to select the most useful models for the requirements provided for them. The experts were asked to rank the usefulness of the models based on the similarity scores. The human experts were given the same requirements as the DWMR approach. The experts agreed on 4 models that are considered most useful for the requirements, which were then used in the repository and rank the usefulness of the models based on their similarity scores.

Next, we assess the performance of our approach by comparing our results with the results of human experts. Five senior PhD students from the Information Systems Department were invited to serve as human experts. The experts were asked to complete the task in two steps. The first task for the experts is to transform the user requirements to the control flow-based workflow modeling algorithms (e.g., [46]), the composed process model of M1, M2 and M3 is shown in Fig. 12.

The experts agreed on 4 models that are considered most useful for the requirements, which were then used in the repository and rank the usefulness of the models based on their similarity scores.

The purpose of the cross-comparison is not to prove that our approach outperforms the human experts. The most critical issue in model reuse is the selection of appropriate models (or model groups). By adopting our approach, the similarity of models (or model groups) and the choices of models (or model groups) can be done in an automatic manner. The results of human experts are used as a cross reference to prove the correctness of our results. In the comparison, there is a high consistency between the results of our approach and the decisions made by human experts. Thus, we conclude that the approach can assist the process designers in improving decision making efficiency, with little loss on correctness.

### 4.4. Comparison with existing approaches

We also applied existing approaches to this case. Through a comparison of the results, the advantage of our approach can be demonstrated. Most existing approaches are control flow-based, which can be further classified into two categories: structure-based and content-based. These approaches are not capable of processing data flow information, which distinguishes the main advantage of our approach. We manually transform the user requirements to the control flow based form. The precision and recall of our results are 100% and 75% respectively. The experts selected Model M7 which was not in our result. M7 is ranked with lowest usefulness by experts and it is not adopted by any model group for model composition. The reason that M7 is selected by experts in the model search step is that the experts are significantly influenced by the semantic information of the models. The words “Strategy” and “Budget” in M7’s name are very close to the expressions in user requirements, which lead the experts to overestimate usefulness of this model.

The results of human experts can be considered as a cross reference to prove the correctness of our results. In the comparison, there is a high consistency between the results of our approach and the decisions made by human experts. This indicates that our approach automatically selected almost the same models (or model groups) as the human experts.

The second task for the experts is to find appropriate compositions of the candidate models to fulfill the requirements. The model groups that at least 4 experts agreed on are included in their final results. The comparison of the results of our approach and the experts is shown in Table 5. The results generated by experts and our approach are almost the same, which makes the precision and recall both 100%. The only difference is the ranks of model groups 2 and 3, which can be explained by the different perspectives adopted by experts and our approach.

Models in bold font are selected by the DWMR approach based on the similarity scores.

<table>
<thead>
<tr>
<th>Group no.</th>
<th>Model</th>
<th>Data similarity</th>
<th>Dependency similarity</th>
<th>Synthesized similarity</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>M1, M2, M3</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>G2</td>
<td>M2, M1</td>
<td>1.0</td>
<td>0.889</td>
<td>0.944</td>
<td>2</td>
</tr>
<tr>
<td>G3</td>
<td>M1, M2</td>
<td>0.667</td>
<td>0.750</td>
<td>0.709</td>
<td>3</td>
</tr>
<tr>
<td>G4</td>
<td>M1, M3</td>
<td>1.0</td>
<td>--</td>
<td>--</td>
<td>Rejected</td>
</tr>
<tr>
<td>G5</td>
<td>M2, M3, M4</td>
<td>1.0</td>
<td>0.889</td>
<td>0.944</td>
<td>Rejected</td>
</tr>
<tr>
<td>G6</td>
<td>M1, M2, M3, M4</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>Rejected</td>
</tr>
</tbody>
</table>

Models in bold font are selected by the DWMR approach based on the similarity scores.

### 4.3. Comparison with human experts

Next, we assess the performance of our approach by comparing our results with the results of human experts. Five senior PhD students from the Information Systems Department were invited to serve as human experts. The experts were given the same requirements as the DWMR approach. The experts agreed on 4 models that are considered most useful for the requirements, which were then used in the repository and rank the usefulness of the models based on their similarity scores.

First, they needed to select the most useful models for the requirements provided for them. The experts were asked to rank the usefulness of the models based on the similarity scores. The human experts were given the same requirements as the DWMR approach. The experts agreed on 4 models that are considered most useful for the requirements, which were then used in the repository and rank the usefulness of the models based on their similarity scores.

The experts agreed on 4 models that are considered most useful for the requirements, which were then used in the repository and rank the usefulness of the models based on their similarity scores.

The purpose of the cross-comparison is not to prove that our approach outperforms the human experts. The most critical issue in model reuse is the selection of appropriate models (or model groups). By adopting our approach, the similarity of models (or model groups) and the choices of models (or model groups) can be done in an automatic manner. The results of human experts are used as a cross reference to prove the correctness of our results. In the comparison, there is a high consistency between the results of our approach and the decisions made by human experts. Thus, we conclude that the approach can assist the process designers in improving decision making efficiency, with little loss on correctness.

### Table 2

<table>
<thead>
<tr>
<th>Model no.</th>
<th>Data similarity</th>
<th>Dependency similarity</th>
<th>Synthesized similarity</th>
<th>Rank according to synthesized similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>M2</td>
<td>0.667</td>
<td>0.625</td>
<td>0.646</td>
<td>1</td>
</tr>
<tr>
<td>M1</td>
<td>0.667</td>
<td>0.167</td>
<td>0.417</td>
<td>2</td>
</tr>
<tr>
<td>M3</td>
<td>0.333</td>
<td>0.429</td>
<td>0.381</td>
<td>3</td>
</tr>
<tr>
<td>M4</td>
<td>0.667</td>
<td>0</td>
<td>0.333</td>
<td>Rejected</td>
</tr>
<tr>
<td>M5</td>
<td>0.667</td>
<td>0</td>
<td>0.333</td>
<td>Rejected</td>
</tr>
<tr>
<td>M6</td>
<td>0.667</td>
<td>0</td>
<td>0.333</td>
<td>Rejected</td>
</tr>
<tr>
<td>M7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Rejected</td>
</tr>
</tbody>
</table>

### Table 3

<table>
<thead>
<tr>
<th>Group no.</th>
<th>Model</th>
<th>Data similarity</th>
<th>Dependency similarity</th>
<th>Synthesized similarity</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>M1, M2, M3</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>G2</td>
<td>M2, M1</td>
<td>1.0</td>
<td>0.889</td>
<td>0.944</td>
<td>2</td>
</tr>
<tr>
<td>G3</td>
<td>M1, M2</td>
<td>0.667</td>
<td>0.750</td>
<td>0.709</td>
<td>3</td>
</tr>
<tr>
<td>G4</td>
<td>M1, M3</td>
<td>1.0</td>
<td>--</td>
<td>--</td>
<td>Rejected</td>
</tr>
<tr>
<td>G5</td>
<td>M2, M3, M4</td>
<td>1.0</td>
<td>0.889</td>
<td>0.944</td>
<td>Rejected</td>
</tr>
<tr>
<td>G6</td>
<td>M1, M2, M3, M4</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>Rejected</td>
</tr>
</tbody>
</table>

Models in bold font are selected by the DWMR approach based on the similarity scores.

### Table 5

<table>
<thead>
<tr>
<th>Group no.</th>
<th>Model</th>
<th>Data similarity</th>
<th>Dependency similarity</th>
<th>Synthesized similarity</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>M1, M2, M3</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>G2</td>
<td>M2, M1</td>
<td>1.0</td>
<td>0.889</td>
<td>0.944</td>
<td>2</td>
</tr>
<tr>
<td>G3</td>
<td>M1, M2</td>
<td>0.667</td>
<td>0.750</td>
<td>0.709</td>
<td>3</td>
</tr>
<tr>
<td>G4</td>
<td>M1, M3</td>
<td>1.0</td>
<td>--</td>
<td>--</td>
<td>Rejected</td>
</tr>
<tr>
<td>G5</td>
<td>M2, M3, M4</td>
<td>1.0</td>
<td>0.889</td>
<td>0.944</td>
<td>Rejected</td>
</tr>
<tr>
<td>G6</td>
<td>M1, M2, M3, M4</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
<td>Rejected</td>
</tr>
</tbody>
</table>

Models in bold font are selected by the DWMR approach based on the similarity scores.

### Fig. 12

Composed workflow model.
data flow based expression “(Business Goal Report, approved) \rightarrow (Budget Report, approved)” is transformed to “Establish Business Goals (Task1) \rightarrow Approve and Finalize Budget (Task2).” The control flow based approaches are applied to find the models that are similar to the path from Task1 to Task2.

4.4.1. Structure-based approach

The similarity of two structures is defined based on the edit distance between them [42]. The edit distance equals the minimum number of basic actions that need to be taken to transform one structure to the other. The basic actions include insertion, deletion and substitution of nodes. The similarity of Models M1 and M2 is defined as follows:

$$SIMs(M1, M2) = 1 - \frac{\text{Edit distance}(M1, M2)}{\text{Max}(|M1|, |M2|)}$$

4.4.2. Content-based approach

The similarity of two models is calculated based on the similarities of the task labels [40]. The similarity of two labels is simply defined as “Number of common words in two labels/total amount of words in two labels”, where the meaningless function words are not counted. The content-based similarity of Models M1 and M2 is defined as:

$$SIMc(M1, M2) = \frac{2 \sum_{\text{label1} \in M1, \text{label2} \in M2} \text{SIMl}(\text{label1}, \text{label2})}{|M1| + |M2|}$$

The results of the control flow-based approaches are shown in Table 6. Structure-based and content-based methods cannot process the data flow information automatically. It takes human efforts to transform the information into control flow-based forms. Thus, if a user only provides the data flow information like data input and output, control flow-based approaches will not be effective. The advantage of our approach is lowering the requirement for users’ input.

In the results, M2 is ignored by both structure-based and content-based approaches. However, M2 is accepted by our approach and human experts. The reason is that both control flow-based approaches did not take the data flow information into consideration. M2 has no common tasks or labels, but is closely connected with the user requirements, in terms of data flow. DWMR approach detects this connection through data dependency analysis. The control flow-based approaches rely on the accurate estimate of structures or contents of the process model. Our data-centric approach overcomes this shortcoming.

Both control flow based approaches cannot conduct similarity calculations for model groups with user requirements, and cannot perform model composition. This is another unique advantage of DWMR.

5. Discussion

The DWMR framework proposed in this paper aims to support workflow model reuse based on data flow. Our research provides contributions to the research fields of workflow design and data flow modeling. While traditional process design approaches require users to start from scratch [47], our research provides an alternative reuse-based approach. Most of the existing reuse-based approaches provide solutions for matching and modification for individual models [10, 32, 39, 42]. We propose that the connections among individual models can be detected and the groups of connected models can serve as reusable units. Our approach makes important improvement to the existing reuse-based approaches [10, 32, 39, 42].

Our results also enrich the knowledge base of data flow research. Since being introduced to the business process research field, data flow analysis has been mostly applied to error detection and model verification [23, 24]. By considering data flow as the interconnection mechanism between business process models, we propose methods for multiple model matching and data flow-driven model composition. To the best of our knowledge, our paper is the first attempt to discuss the application of the data flow perspective to model reuse and model composition.

This study also suffers from several limitations. First, we utilize the Oracle Best Practice Processes to simulate the model repository of a company. This data set is provided by Oracle based on their consulting and system design experiences in real business. The source of this data set guarantees that to a high degree it can reflect the real situation in a business environment. We admit that we have fewer models than that in a real repository of an organization. Second, some variables used in the model searching and model composition algorithms are not easy to determine. For example, the weights of data similarity and dependency similarity are set as the same when calculating the synthesized similarity. The best values of these variables should be determined by continuous tests in a real data environment. These values may be different for process models from different companies, because of the variety of business contexts and data qualities.

6. Conclusion

Existing workflow models can be used as a knowledge base for workflow model design. However, developing workflow models based on existing models is challenging because of the volume and complexity of business processes. In order to facilitate workflow model reuse, we proposed a data-centric framework for workflow model reuse (DWMR),

<table>
<thead>
<tr>
<th>Table 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparison of individual model search results.</td>
</tr>
<tr>
<td>Model no.</td>
</tr>
<tr>
<td>M1</td>
</tr>
<tr>
<td>M2</td>
</tr>
<tr>
<td>M3</td>
</tr>
<tr>
<td>M7</td>
</tr>
</tbody>
</table>

Precision = 100%, recall = 75%

Models in bold font are selected by the DWMR approach based on the similarity scores.

<table>
<thead>
<tr>
<th>Table 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparison of model group results.</td>
</tr>
<tr>
<td>Group no.</td>
</tr>
<tr>
<td>G1</td>
</tr>
<tr>
<td>G2</td>
</tr>
<tr>
<td>G3</td>
</tr>
</tbody>
</table>

Precision = 100%, recall = 100%

Models in bold font are selected by the DWMR approach based on the similarity scores.
which includes three components: a formal data structure for workflow model repository and indexing, a model searching algorithm and a model composition algorithm. We applied DWMR approach to a business case to show its feasibility. Further, we compared the results generated by our approach and human experts, and found that our approach has a high accuracy in terms of precision and recall. Our approach sets up the foundation for automating workflow model storage, search and composition, which enables effective workflow model reuse. DWMR is data-centric and addresses issues caused by control flow-based approaches, such as data information loss, cross-paradigm model integration and error-prone problems.

We plan to extend this research in two directions. First, we plan to develop a system to facilitate workflow model design based on DWMR approach and conduct a user study to further evaluate our approach. Second, we will look into the semantics of data items and plan to use semantic techniques to enhance model search accuracy and reduce user interventions in the model matching procedure.

Acknowledgements

The work described in this study was partially supported by grants from National Natural Science Foundation of China (No. 71431002, 71421001, 71461023, 71573030, 71610027), the Fundamental Research Funds for the Central Universities (No. DUT15SRC(3)004), and a JP Morgan Chase Fellowship from the Institute for Financial Services Analytics at the University of Delaware.

References

journals, such as Information Systems Research, Decision Support Systems, ACM Transactions on Management Information Systems, and Information and Management. Dr. Wang has served as associate editor, special issue guest editor, and editorial board member for several journals and organized the 2014 Workshop on Business Processes and Service and the 2013 China Summer Workshop on Information Management as a conference co-chair. He has also been a program co-chair, program committee member, and track co-chair for numerous conferences.

J. Leon Zhao is Chair Professor in Information Systems and was Head of Information Systems (2009 – 2015), College of Business, City University of Hong Kong. He was Interim Head (2006-2007) and Eller Professor in MIS, University of Arizona. He also taught at HKUST and College of William and Mary previously. He holds Ph.D. from Haas School of Business, UC Berkeley. His research is on information technology and management with a particular focus on risk management, collaboration and workflow management, and information services for business and finance. His current interests include big data, blockchain, social networks, and FinTech. He is director of Center on Global Internet Finance and Lab on Enterprise Process Innovation and Computing. He has published over 200 academic papers in conferences and journals and held various editorships in several academic journals and chairpersonships in numerous conferences. He received IBM Faculty Award in 2005 and was awarded Chang Jiang Scholar Professorship at Tsinghua University in 2009.