A Data-centric Perspective for Workflow Model Management

Research-in-Progress

Zhiyong Liu
Univ. of Science and Technology of China - City University of Hong Kong
Joint Research Center
Suzhou, P.R. China
liuzy@mail.ustc.edu.cn

J. Leon Zhao
Department of Information Systems,
City University of Hong Kong
Kowloon, Hong Kong
jlzhao@cityu.edu.hk

Harry Jiannan Wang
Department of Accounting and MIS,
University of Delaware
Newark, DE, USA
hjwang@udel.edu

Huaping Chen
University of Science and Technology of China
Hefei, P.R. China
hpchen@ustc.edu.cn

Abstract

With increasingly widespread adoption of workflow technology as a standard solution to business process management, much effort is put on designing appropriate workflow models. The reuse of existing models has been recently suggested to support more efficient workflow design, which requires the storage, search and composition of workflow models. In this paper, we propose a new framework of workflow model management from a data-centric perspective, which provides unique features lacking in existing approaches based on control flow and function description. We develop a data-centric indexing method for organizing workflow models in the repository, design a flooding algorithm for searching workflow models, and present a composition approach for integrating different workflow models to meet specific user requirements.

Keywords: workflow model management, model reuse, data flow modeling, data dependency, data-centric workflow composition
Introduction

With increasingly widespread adoption of workflow technology as a standard solution to business process management (BPM), much effort is put on designing the appropriate workflow models that satisfy the business logic and constraints for the 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 (Stohr and Zhao 2001). Choosing the effective and efficient workflow models is a key factor of BPM success.

Workflow modeling plays a crucial role in BPM, which captures the implicit process knowledge and domain knowledge in organizations. Many modeling methods have been proposed and applied by both academic researchers and business practitioners, such as Petri net (Murata 1989), UML activity diagram (Dumas and ter Hofstede 2001), logic-based method (Bi and Zhao 2004), metagraph (Basu and Blanning 2000), BPMN (Wohed et al. 2006), and data-flow-based approach (Sun et al. 2006). In particular, data-flow-based workflow modeling is a relatively new method, which designs the workflow models by analyzing the input and output data of tasks and their dependency relationships (Sun et al. 2006).

Compared with the other workflow modeling methods, which require business users to provide plenty of information about specific processes, such as description of business function, sequence of tasks and structured representation of business rules, data-flow-based modeling method only needs the users to identify the input and output data in the processes, which is an easier task given those data items are usually contained in the documents they use on a daily basis.

Besides starting from scratch, workflow models could also be developed based on existing similar models. Model reuse will certainly 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, which are named Process Reference Models (PRMs), such as SAP Process Reference Model and Oracle Best Practice Processes (Wang and Wu 2010). Besides the function-based reuse of reference models, researchers also introduced Workflow Model Patterns (or Workflow Model Components) as basic reusable units of models which can accomplish certain decomposed business functions (Zhuge 2003). This kind of model reuse usually mixes function-based reuse and structure-based reuse. In addition, case definitions at schema level and instance level, and case descriptions, can also be sources for reuse. Case repositories have been developed to support case-based model reuse (Madhusudan et al. 2004). Nevertheless, little research has been done about workflow model reuse based on a data-centric approach, which takes the data flow structure of workflow models as the focus of model management.

In this paper, we fill this research gap by proposing a framework of workflow model management, which consists of model storage, model search and model composition via a data-centric approach. We design the data-centric approach for workflow model management and evaluate the approach by applying it to a banking process and by comparing it with similar approaches. Our contributions are as follows: First, we define the formal data structure of a workflow model repository and propose a data-centric indexing method of workflow models based on data dependency relationships. Second, we propose a method for the match of user inquiries and candidate workflow models based on data similarity, and develop a flooding algorithm to search for model groups to satisfy user inquiry. Third, we develop a formal method to compose candidate groups of models to satisfy specific user requirements.

Brief Literature Review

We briefly survey the relevant works about data flow modeling methods and workflow model reuse. Data flow modeling is a relatively new workflow modeling method. Compared with traditional modeling methods that focus on the structure issues (Kiepuszewski et al. 2000), data flow is considered as an important complement to the specification of workflow. Sadiq et al. (2004) identify the importance of data flow issues to workflow research, and introduce the modeling, 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. (2005) describe a series of workflow data patterns to show the different ways of representation and utilization of data flow in workflow system. The analysis of data flow is a useful tool for
workflow verification, which has been used to validate the correctness of control flow (Sun et al. 2004). Sun et al. (2006) 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 (Guo et al. 2008). A formal workflow language based on Petri net and nested relational calculus has been proposed to support data flow modeling (Hidders et al. 2008).

Reusing the existing models is an efficient and effective way to support model design. Wang and Wu (2010) propose a collaborative approach to integrate and manage process reference models (PRMs). Their approach is based on ‘spiral’ model for organizational knowledge creation and leverage Web 2.0 technologies, such as social tagging and classification to maintain PRMs. Besides the reference models, the model patterns (or model components) are also reusable. Zhuge (2003) defines workflow component with four characteristics: independency, encapsulation, completeness and consistency. Thom et al. (2007) classify workflow patterns into nine categories, and Altintas et al. (2005) abstract workflows into several components according to their service functions. Cao et al. (2005) define workflows components, which consist of function, quality of service, control flow model and business domain.

The model design cases are also a reusable resource, which includes more than control flow models but also case descriptions such as information on functions, agents and resource. Madhusudan et al. (2004) propose a framework to reuse two categories of workflow cases, i.e., prototypical cases and instance cases. The indexing structure of the cases is a set of hierarchies, which includes functional, organizational, and task hierarchies. They also introduce a similarity flooding algorithm to support case retrieval. The similarity-based model matching is another important aspect of model reuse. Madhusudan et al. (2004) support case retrieval, by matching the NAME properties of nodes in models through language processing and string matching techniques to get the initial similarities. The final similarities are calculated by iterative fix-point computation. Zhuge (2002) 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. Xiao et al. (2009) define the overall similarity between a given workflow pattern and candidate Petri nets which is expressed as the weighted sum of the three similarities, i.e., vertices, arcs and paths similarities.

A Data-centric Approach for Workflow Model Management

An Illustrative Example

We will illustrate our approach using two simplified banking processes. A process model repository has been built for the bank to store the existing models to facilitate new workflow model design. The two example processes are shown in Figure 1 and 2 using extended UML activity diagram as defined in (Sun et al. 2006).

Transfer Application: After receiving a transfer application, the bank firstly verifies the application to ensure that the information is accurate, and checks the completeness to make certain that all the information needed is provided. When the status data items “Application Verified” and “Application Complete” are received, the “Received Application” data item is transformed to an “Accepted Application”. 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 transfer process finishes.
Credit Payment Application: After “Transaction Information” and “Authentication Information” are received from the client, the bank accepts the credit payment application. Then the “Credit History” and “Liquid Assets” of the customer are checked to create a “Credit Score”. The information of this transaction and the credit score of the customer are analyzed to give an overall evaluation such as risk level and amount adjustment. Finally, the application with evaluation result is approved by the officer.

A new workflow model of Loan Application process needs to be designed, which begins from receiving an application and ends with that application being approved. The requirement can be concluded as modeling a process that produces the data “Application Approved” with input data “Received Application”. Besides starting from scratch, we can reuse the two aforementioned models. The data structure of the models can be used as index. The candidate models are found by similarity-based searching. Considering the possible incapacity of individual models to fulfill the requirements, composition of candidate models is also supported. In following sections, we will explain our approach in detail, which facilitates data-centric model reuse.

Model Storage

Definition 1 (Workflow Model) A workflow model is denoted as W= {Description, Control-flow, Data-flow}: Description contains the business domain, function and information related to the application of the model; Control-flow describes the structure of the control flow, which consists of the task set and set of flows between tasks; Data-flow is a set of relationships between tasks and data, which contains tasks and their input and output data.

The data structure of workflow models stored in the repository can be showed as follow:

\[
\{ \text{Model Name } W; \\
\text{Model Description}; \\
\text{Control Flow Structure } \Gamma = \{ T, F \}; \\
\text{Data Set } D = \{ d_i | i=1, 2, ..., n \}. \\
\text{Task-Data Relationship } TD = \{ (t, D_{\text{in}}t, D_{\text{out}}t) | t \in T \}. \}
\]

W 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 \( W \), \( T=\{t_i | i=1, 2, ..., m \} \), \( t \) is a specific task in \( T (t \in T) \). \( F \) is a finite set of the flows between tasks in \( T \), \( F= \cup (t_i \times t_j), t_i \in T, t_j \in T, i \neq j \). The data items which are used or produced in this model form a data set \( D \). \( TD \) describes the relationship between input/output data and tasks, \( D_{\text{in}}t \) and \( D_{\text{out}}t \) are the sets of input and output data of task \( t \).

For example, the workflow model of Transfer Application shown in Figure 1 can be stored as follow:

\[
\{ \text{Model Name: Transfer Application}; \\
\text{Model Description: Receive, verify, check and accept transfer applications}; \\
\text{Control Flow Structure } \Gamma = \{ T= \{ t_1: \text{Receive Application}, t_2: \text{Verify Application}, t_3: \text{Check Application Completeness}, t_4: \text{Accept Application}, t_5: \text{Check Password}, t_6: \text{Check Balance}, t_7: \text{Transfer} \}, F= \{ (t_1 \times t_2), (t_1 \times t_3), (t_2 \times t_4), (t_3 \times t_4), (t_4 \times t_5), (t_5 \times t_6), (t_6 \times t_7) \} \};
\]
Data Set: $D = \{d_1: \text{Received Application}, d_2: \text{Application Verified}, d_3: \text{Application Complete}, d_4: \text{Accepted Application}, d_5: \text{Password Correct}, d_6: \text{Account Balance}, d_7: \text{Adequate Balance}, d_8: \text{Transfer Finished}\}$

Task-Data Relationship: $TD = \{(t_1, \Phi, \{d_1\}), (t_2, \{d_1\}, \{d_2\}), (t_3, \{d_1\}, \{d_3\}), (t_4, \{d_1, d_2, d_3\}, \{d_4\}), (t_5, \{d_4\}, \{d_5\}), (t_6, \{d_4, d_5\}, \{d_6, d_7\}), (t_7, \{d_4, d_7\}, \{d_8\})\}$. 

**Definition 2** (Direct Data Dependency and Indirect Data Dependency) Data Direct Dependency relationship is defined as: a task cannot produce data B without data A as input, then, we call B directly depends on A, denoted as $A \rightarrow B$. When $A_1 \rightarrow A_2, A_2 \rightarrow A_3, \ldots, A_{n-1} \rightarrow A_n$ ($n=3, 4, \ldots$), we say $A_n$ indirectly depends on $A_1$, which is denoted as $A_1 \Rightarrow \cdots \Rightarrow A_n$.

**Definition 3** (Data Item Description) Data Item Description consists of optional description of the data item, dependency relationships and data-task relationship showing the tasks that use specific data as input or output.

The data structure of data items can be showed as follow:

\[
\{\text{Data Name} \ d; \\
\text{Model Name} \ W; \\
\text{Data Description}; \\
\text{Direct Dependency} \ \text{DP}_d = \{(d \rightarrow d_i) | d,d_i \in D, i=1,2,\ldots\}; \\
\text{Indirect Dependency} \ \text{DP}_i = \{(d \rightarrow \cdots \rightarrow d_j) | d,d_j \in D, j=1,2,\ldots\}; \\
\text{Data-Task Relationship} \ \text{DT} = \{(d, T_i, T_o)\}. \]

\(d\) is the identification of this data item. \(W\) is the name of the model that \(d\) belongs to. \(\text{DP}_d\) and \(\text{DP}_i\) are the sets of direct and indirect data dependency relationships of \(d\) respectively. \(\text{DT}\) records the tasks related to this data item \(d\), \(T_i\) and \(T_o\) are the sets of tasks that use \(d\) as input and output respectively.

In the Credit Payment Application example shown in Figure 2, the data item $d_c$ (“Accepted Application”) can be stored as follow:

\[
\{\text{Data Name: Accepted Application}; \\
\text{Model Name: Credit Payment Application}; \\
\text{Data Description: the information provided by the applicant is accepted and recorded}; \\
\text{Direct Dependency: } \text{DP}_d = \{(d_c \rightarrow d_4), (d_c \rightarrow d_5), (d_c \rightarrow d_6), (d_c \rightarrow d_7)\}; \\
\text{Indirect Dependency: } \text{DP}_i = \{(d_c \rightarrow \cdots \rightarrow d_j), (d_c \rightarrow \cdots \rightarrow d_j)\}; \\
\text{Data-Task Relationship} \ \text{DT} = \{(d_c, t_4, t_5, t_7, t_8)\}. \}
\]

**A Data-centric Workflow Model Indexing Method -- Data Dependency Structure**

**Definition 4** (Data Dependency Structure) A Data Dependency Structure (DDS) is a mapping from the data flow structure of a specific workflow model. DDS consists of the data set and the direct dependency relationship between data, which is denoted as: DDS = \{\(D, \text{DP}_d\)\}.

![Figure 3. Examples of Data Dependency Structure](image-url)
For example, in the model of Transfer Application, the data set \(D = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8\}\), the direct dependency \(DP_d = \{(d_1 \rightarrow d_2), (d_1 \rightarrow d_3), (d_1 \rightarrow d_4), (d_2 \rightarrow d_4), (d_3 \rightarrow d_4), (d_4 \rightarrow d_5), (d_4 \rightarrow d_6), (d_4 \rightarrow d_7), (d_4 \rightarrow d_8), (d_5 \rightarrow d_6), (d_6 \rightarrow d_7), (d_7 \rightarrow d_8)\}\). We represent the data items with ovals and indicate the direct dependency relationship with arrows. The DDS of this model is illustrated in Figure 3(a). From this visual index of this model, we can intuitively identify the direct and indirect dependency relationships. The data dependency structure of model of Credit Payment Application is created similarly, as shown in Figure 3(b).

**Model Search**

We show a sample model search requirement as follows.

- **User Inquiry**: “External Input \(\rightarrow\) External Output”
- **Expected Result**: Matched individual Models or Model Groups

The user describes the possible input and output data for the required model. For the Loan Application process, one search requirement can be expressed as \{'Received Application}\(\rightarrow\)\{Application Approved\}\). The expected result could be an individual model which satisfies the user requirements or several models collectively fulfill the requirement after appropriate composition. According to the different types of expected results, we further classify the data-centric model matching into *Individual Model Matching* and *Multiple Models Matching*.

**Individual Model Matching**

The user requirements for individual model matching are shown below:

- **User Inquiry**: \(D_{in} \rightarrow \rightarrow D_{out}\)
- **Expected Result**: Individual Models contain \("d_{in} \rightarrow \rightarrow d_{out}\)\), \((d_{in} \in D_{in}, d_{out} \in D_{out})\)

\(D_{in}\) (or \(D_{out}\)) is a set of data items required by the user to be input (or output) of the model, \(d_{in}\) and \(d_{out}\) are data items, \(d_{in} \in D_{in}, d_{out} \in D_{out}\). The user expects the target model to produce output data \(D_{out}\) using input data \(D_{in}\). If an individual model satisfies this requirement, then the dependency relationship \("d_{in} \rightarrow \rightarrow d_{out}\) should exist in the data flow structure of this model.

For example, the user requirement for individual model matching of the Loan Application process modeling can be expressed as follows: \(D_{in}=\{\text{Received Application}\}\) and \(D_{out}=\{\text{Application Approved}\}\). The expected result is a set of models \(W=\{w_i\}\). The dependency relationship set of \(w_i\) is denoted as \(w_i.DP\), which includes both direct and indirect dependencies in \(w_i\). The dependency required by user “Received Application\(\rightarrow\rightarrow\) Application Approved” should belong to \(w_i.DP\), which means the returned models are capable to produce “Application Approved” with input “Received Application”.

- **User Inquiry**: \{Received Application\}\(\rightarrow\rightarrow\}\{Application Approved\}\.
- **Expected Result**: \(W=\{w_i|\text{“Received Application}\rightarrow\rightarrow\text{Application Approved” }\in w_i.DP, i=1,2,\ldots\}\)

There may be more than one model that contains “\(d_{in} \rightarrow \rightarrow d_{out}\)”. The most appropriate result should be the one with the highest similarity with user requirement. By extending the definition of similarity measure defined in (Tversky 1977), we give the definition of similarity as follow:

**Definition 5** (Data Set Similarity) The data set described in the user requirement is denoted as \(D_U\) (\(D_U = D_{in} \cup D_{out}\)), data set contained in a candidate model is denoted as \(D_M\), data set similarity is defined as following:

\[
\text{SIM}_{Data} = \frac{|D_U \cap D_M|}{|D_U \cap D_M| + \alpha \cdot |D_U - D_M|}
\]

The data set similarity focuses on the proportion of the common elements between user data set and model data set. The variable \(\alpha\) \((\alpha \geq 0)\) denotes the weight of the data set that includes the data items in user requirements but not in the model. In practice, the variable \(\alpha\) should be predefined. The initial value of \(\alpha\) can be assigned with 1 which means that the common and uncommon data sets between user
We evaluate the similarity between user requirements and candidate models from two aspects: data set similarity and data dependency.

We assign 0.5 to $\gamma$ in practice, which means the importance of data set similarity and data dependency. The overall similarity should be a synthesis of the above two, which is defined next.

**Definition 6** (Dependency Similarity) The data dependency relationship which is described in user requirements is denoted as DP$_U$, which is a set of dependencies like “$d_{in}$$\rightarrow$$d_{out}$” ($d_{in}$$\in$$D_{in}$, $d_{out}$$\in$$D_{out}$). We can simply express that as “DP$_U$ = $D_{in}$$\rightarrow$$D_{out}$”. The dependency contained in a candidate model is denoted as DP$_M$ (DP$_M$ = DP$_d$$\cup$ DP$_p$). Dependency similarity is defined as following:

$$\text{SIM}_{\text{Dep}} = \frac{|\text{DP}_{U} \cap \text{DP}_{M}|}{|\text{DP}_{U} \cap \text{DP}_{M}| + \beta \cdot |\text{DP}_{U} - \text{DP}_{M}|}$$

Similar to the definition of data set similarity, dependency similarity focuses on the common part of dependency sets of user requirements and the candidate model. The variable $\beta$ ($\beta$$\geq$$0$) denotes the weight of uncommon dependency set, and should be predefined in practice. Similar to $\alpha$, $\beta$ is assigned with an initial value of $1$, and the most appropriate value can be obtained by 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, which is defined next.

**Definition 7** (Synthesized Similarity) Synthesized Similarity is the weighted sum of data set similarity and dependency similarity:

$$\text{SIM}_{\text{Syn}} = \gamma \cdot \text{SIM}_{\text{Data}} + (1 - \gamma) \cdot \text{SIM}_{\text{Dep}}$$

For $0$$\leq$$\gamma$$\leq$$1$, $\gamma$ and $1$$-$$\gamma$ represent the weights of SIM$_{\text{Data}}$ and SIM$_{\text{Dep}}$ 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 process modeling example, we assume the candidate individual model is Credit Payment Application model shown in Figure 2. The user requirement is described in Figure 6. The common data set between user requirements and the model data set is $\{\text{Application Approved}\}$, and the uncommon data set is $\{\text{Received Application}\}$. SIM$_{\text{data}}$= 0.5 ($\alpha$$=1$). The user required dependency “Received Application$
\rightarrow$$\rightarrow$$\rightarrow$$\rightarrow$Application Approved” is not contained in the model's dependency set, so the common dependency set is empty and SIM$_{\text{dep}}$=0. Then synthesized similarity between user requirement and the candidate model is 0.25. (SIM$_{\text{Syn}}$=0.5$\times$0.5+0=0.25, $\gamma$=0.5)

**Multiple Model Matching**

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

**User Inquiry:** $D_{in}$$\rightarrow$$\rightarrow$$\rightarrow$$\rightarrow$$d_{out}$

**Expected Result:** Models Groups contain “$d_{in}$$\rightarrow$$\rightarrow$$\rightarrow$$\rightarrow$$d_{out}$”

(i.e. A model set S={$W_i|i=1, 2, \ldots, n$}, in which $W_i$ contains “$d_{in}$$\rightarrow$$\rightarrow$$\rightarrow$$\rightarrow$$d_{out}$”, Model $W_1$ contains “$d_{a1}$$\rightarrow$$\rightarrow$$\rightarrow$$\rightarrow$$d_{in}$”, ..., Model $W_6$ contains “$d_{a6}$$\rightarrow$$\rightarrow$$\rightarrow$$\rightarrow$$d_{in}$”, ($d_{in}$$\in$$D_{in}$, $d_{out}$$\in$$D_{out}$))

After finding a model $W_i$ which contains $d_{out}$, and when the initial similarity between $W_i$ and user requirement exceeds a predefined threshold (denoted as ‘a’), we scan $W_i$ to get a data set (denoted as D$_i$) where all data items are depended by $d_{out}$. For every data item in this data set, e.g., $d_{in}$, the second models $W_2$ with a data item $d_{in}$, which satisfies “$d_{a1}$$\rightarrow$$\rightarrow$$\rightarrow$$\rightarrow$$d_{in}$” can be derived. Then a new data set (denoted as D$_2$) from $W_2$ can be found where all data items are depended by the previous data item ($d_{in}$). The above procedure can be repeated until $d_{in}$ appears in a new data set. Because the complexity of this procedure increases sharply when more models added to the group, a threshold of model group size 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 Flooding Algorithm to support multiple model matching as following:

**Step 1:** Search for the models that contain $d_{out}$ and SIM$_{\text{data}}$ $>$ a, name the returned model set as S$_a$, define a temporary data set as D$_i$, D$_i$ = D$_{out}$. Define a count of numbers of models as k, k=0;

**Step 2:** For every model $W_i$ in S$_k$ (k=1,2,\ldots,), search for the data set D$_{i+1}$, where for every $d_i$$\in$$D_{i+1}$ there exists “$d_{i}$$\rightarrow$$\rightarrow$$\rightarrow$$\rightarrow$$d_{a}”($d_i$$\in$$D_{i}$). If $d_i$$\in$$D_{in}$, add $W_i$ to the result model group G and finish the search.
Step 3: Repeat Step 2, until search finishes or k>=n. (‘n’ is the predefined threshold to limit the size of a model group).

Model Composition

After identification of the candidate model groups, we need to compose the multiple models and delete the unnecessary parts. The candidate models are Transfer Application model (model I for short) and Credit Payment Application model (model II for short) shown in Figure 1 and 2.

In this example, the user inquiry is “{Received Application} - -> {Application Approved}”, which means with the input “Received Application” the composed model returned by search should produce “Application Approved” as output. In the model search phase, using the flooding algorithm, the dependency relationship “Accepted Application - -> Application Approved” (d_{c} - -> d_{h}) in model II is identified, d_{4} in model I has the same description with d_{c}, i.e., “d_{4}=d_{c}”. Then the dependency relationships “Received Application - -> Accepted Application” (d_{1} - -> d_{4}) is identified which satisfies the ending condition of the model search. The resulting model should be a composition of the two models that contain “d_{r} - -> d_{4}” and “d_{r} - -> d_{h}” respectively (d_{r}=d_{r}).

Definition 8 (Linking Node and Linked Node) Assume dependency “d_{i} --> d_{j}” (i≠j) exists in model W_{1} and dependency “d_{p} --> d_{q}” (p≠q) exists in model W_{2}. If d_{j} = d_{p}, then the dependency “d_{i} - -> d_{q}” can be achieved after composition of W_{1} and W_{2}. We call d_{j} and d_{p} a Linking Nodes Pair. d_{j} and d_{p} are denoted as Link Node and Linked Node respectively. For example, d_{4} is called link node and d_{c} is called linked node in Figure 8.

A Data-centric Model Composition Method:

Prune: Taking the link nodes, e.g. d_{4} in model I in Figure 11, and the external output node, e.g. d_{h} in model II in Figure 11, as the ending nodes, we search the nodes that are not included in the dependency relationships of the ending nodes. Delete data items that are not depended by the output nodes or the link nodes, i.e., d_{5}, d_{6}, d_{7}, d_{8}, and the associated dependency relationships. Delete data items that are depended by linked nodes, i.e., d_{a}, d_{b}, and the associated dependency relationships. Delete the direct dependency relationships that end with linked nodes.

Combine: After pruning the unnecessary data and dependency relationships, merge the linking nodes pair with the same description as a single node (d_{4} and d_{c}). Replace the link nodes and linked nodes in the dependency relationships with the merged node.

Figure 4. An Example of Model Composition
Mapping: The composition of the data dependency structure helps us to design a reference control flow model. We propose several principles: (1) After identifying the unnecessary data and data dependency relationships, we can identify the redundant tasks in the workflow. (2) The description of some tasks can be revised after composition which means some of these tasks’ output data are unnecessary and the associated work for producing these data can be omitted. (3) The structural relationships among remained tasks should be kept. (4) The data dependency relationship determines the execution sequences of tasks. (5) The reference model should be revised by incorporating domain knowledge to get the control flow model.

Conclusion and Future Research

In this paper, we propose a new data-centric approach for storing, searching, and composing workflow models. We demonstrate and evaluate our approach via an illustrative example. We are working on several issues to improve on the preliminary results presented in this paper. First, we will refine the proposed approach by fine-tuning some concepts and variables, such as the synthesized model similarity, similarity weight variables, and the computational complexity of key algorithms. Second, we plan to further evaluate our approach in a number of ways, such as a comprehensive comparison with existing approaches, developing a prototype system, and conducting user experiments to test the usability, efficiency, and effectiveness of our approach. Third, we are looking into semantics of data item descriptions and plan to leverage concepts in semantic webs to enhance the model matching procedure presented in this paper.

Acknowledgement: This work was partially supported by an SRG grant from the City University of Hong Kong (No. 7008121).
References


