Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2004 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2004

Data Flow Modeling and Verification in Business Process Management

Sherry Sun
University of Arizona

Leon Zhao
University of Arizona

Olivia Sheng
University of Utah

Data Flow Modeling and Verification in Business Process Management

Sherry X. Sun
Department of MIS
University of Arizona
xiaoyun@bpa.arizona.edu

J. Leon Zhao
Department of MIS
University of Arizona
lzhao@bpa.arizona.edu

Olivia R. Liu Sheng
Accounting and Information Systems
University of Utah
olivia.sheng@business.utah.edu

ABSTRACT

Workflow technology has become a standard solution to managing increasingly complex business processes. Successful business process management depends on effective workflow modeling tools. Recently, researchers have developed a variety of workflow models, focusing mainly on the control and coordination of tasks, i.e. the control flow perspective. However, most of these workflow models found in the literature have paid little attention to the data flow perspective. In this paper, we investigate the data flow issues and propose a data flow modeling approach in the context of business process management. Our research objective is to develop a data flow modeling tool that supports data flow verification in workflow systems. Our contributions include also the classification of data flow operations and data flow anomalies.

KEYWORDS: workflow modeling, data flow modeling, data flow anomalies, data flow verification.

INTRODUCTION

Workflow systems evolved originally from the concept of office automation that intended to eliminate paper work and to automate processes in office environment. Nowadays, workflow applications have been applied to automating complex business processes in various business domains such as supply chain management and customer relationship management. In e-business environment, workflow is used as a business hub that coordinates and controls information flows and business activities through web access (Stohr and Zhao, 2001).

Successful business process management depends on effective workflow modeling tools. Recently, researchers have developed a variety of workflow models, including the widespread use of Petri nets and its variants (van der Aalst, 1998; van der Aalst and van Hee, 2002), activity-based workflow modeling (Bi and Zhao, 2004; Georgakopoulos, Hornick and Sheth, 1995), and metagraphs (Basu and Blanning, 2000). However, it is noteworthy that most of the workflow models focus mainly on the control and coordination of tasks, i.e. the control flow perspective. So far, little attention has been paid to the data flow perspective, and consequently, there is a need for data flow modeling and analysis methods in workflow systems.

In this paper, we propose a novel methodology for data flow modeling and verification. Our primary motivation is to develop a data flow modeling tool that supports data flow verification in workflow systems and, thus, automate the design of workflow applications. By data flow verification, it means detecting data flow anomalies such as missing data, redundant data, and potential data conflicts, which can occur in a workflow with its control flow being syntax error free. Presently, workflow management systems enable the discovery of data flow anomalies via simulation, which is inefficient and inaccurate. Furthermore, data flow verification helps not only maintain data integrity in a workflow, but also validate its control flow. In this work, we also classify data flow operations and data flow anomalies in workflow systems.

Next, we give a brief review of literature relevant to data flow modeling and verification. Then, we present analytical models of the five perspectives of workflow, which is followed by a workflow example and a classification of data flow operations. We also conceptualize three types of data flow anomalies, namely, missing data, redundant data, and conflict data. A high

level algorithm for data flow verification is presented. Finally, we conclude the paper by pointing out the contributions and future research directions.

LITERATURE REVIEW

A workflow model describes a business process at conceptual level and consists of the following elements: roles, actors, tools/applications, tasks and processes, rules, and data/documents (Stohr and Zhao, 2001). The conceptualization of these elements and their interrelations models an organization from five perspectives: the functional perspective, the behavioral perspective, the informational perspective, the operational perspective, and the organizational perspective (Curtis, Kellner and Over, 1992; Stohr and Zhao, 2001). The *functional perspective* describes what a workflow performs. The *behavioral perspective* specifies the conditions for tasks to be executed. The *information perspective* defines what data are consumed and produced with respect to each activity in a business process. The *operational perspective* specifies how a particular task is executed, i.e., what tools and application programs are used to execute the task. The *organizational perspective* describes the personnel that are assigned to various job functions.

In workflow management, modeling data flow is important because activities cannot be executed properly without sufficient information. As pointed out by Basu and Kumar (Basu and Kumar, 2002), data usage analysis, one of the three types of structure analyses in workflow, is needed for prevention of unintentional errors and maintenance of data integrity in workflows. Data usage analysis can be achieved through data flow modeling that models what data are consumed and produced as both intermediate output and final results in a workflow. As an irreplaceable component of workflow modeling, data flow modeling complements control flow modeling and organizational modeling. It can help verify and validate data flow and derive rules for creating data models and forms. However, to our best of knowledge, there are no formal methodologies that can support such functions in workflow systems.

Various modeling methods such as petri nets, metagraph, and activity-based workflow modeling have been proposed for modeling and verifying workflow designs (Basu and Blanning, 2000; Bi and Zhao, 2004; van der Aalst and van Hee, 2002). As formalism for workflow modeling, Petri nets has a solid theory base and therefore can provide rigorous methods for process analysis and verification (van der Aalst, 1998; van der Aalst and van Hee, 2002). The activity-based modeling formalism provides another important tools in workflow modeling paradigm (Bi and Zhao, 2004). Metagraphs are graphical structures that are derived from both directed graphs and hypergraphs (Basu and Blanning, 2000). One distinct feature of metagraphs is that it can represent not only the relationship between individual elements but also the relationship between sets of elements. Another important feature of metagraphs is that it supports formal analysis of workflow from information perspective and answers the questions not only about tasks but also about data and resources. However, none of these modeling methods focus on data flow modeling and therefore cannot support data flow verification.

In Petri nets models, the only component that can represent the data input and output is tokens. A token can represent an insurance claim, a purchase order, or anything that initiate a workflow instance. The problem is that there is no rigorous way to map a token contained in Petri nets to a concrete data object stored in databases and with Petri nets it is not possible to analyze data flow. In the activity-based workflow modeling there is no easy way to capture data flow. Even though metagraphs can help analyze data flow, metagraphs provide neither tools to examine the syntactic and semantic correctness of a model nor methods to direct mapping from the data elements modeled in metagraphs to the concrete data object stored in databases. A recently developed model for workflow management is referred to as State-Entity-Activity-Model (SEAM), in which activities and data objects are integrated in the same model (Bajaj and Ram, 2002). However, this model does not focus on modeling and verification of data flow. Although Data Flow Diagram (DFD) is widely used in system analysis and design (Yourdon and Constantine, 1979), it does not support formal analysis of workflow from the data flow perspective due to the lack of underlying formal theoretical models.

Because there are not sufficient formalisms for data flow modeling and verification in workflow systems, documents and forms are produced manually in practice and questions about data input and output cannot be checked automatically, leading to inefficiencies in workflow design and potential errors in workflow implementation.

FORMAL WORKFLOW MODELS

To effectively modeling all the elements in a workflow, we use the following schema to represent a workflow model.

Definition 1 (**workflow model**). A workflow model is defined as a 11-tuple, $W = (V, E, I(V), O(V), R(E), X, A, L, H(L), S(A, L), F(V, L))$, where V is a set of tasks, E is a set of edges that define the precedence of tasks, $I(V)$ is a set of data input for different tasks, $O(V)$ is a set of data output of different tasks, $R(E)$ is a set of routing rules that define the conditions for the execution of tasks. X is a set of operations the tasks perform on data. A is a set of actors and applications that perform the tasks, and L is a set of roles that defines the responsibilities and job functions to be assigned to actors. H is a set of hierarchical relations among different roles. S is a set of role assignment relations that describe how actors are assigned to play certain roles. $F(V, L)$ is a set of relations between V and L and describes which tasks will be conducted by a role. To

manage the complexity of the workflow model, we decompose it into four parts, a control flow model, a data flow model, an organizational model, and an access control model with different parts focusing on different perspectives.

Definition 2 (**control flow model**) A control flow model is defined as a 3-tuple, $P = (V, E, R(E))$.

A control flow model specifies the tasks, the precedence of the tasks, and the routing rules that describe under what conditions a task is to be executed. The routing rules are defined on E since routing rules determine the path of a workflow instance. A complete control flow model also provides a collection of tools for checking control flow anomalies, i.e. deadlock, lack of synchronization, infinite cycles, and tasks without activation or termination (Bi and Zhao, 2004). According to this definition, Petri Nets and activity-based models can both be considered as control flow models.

Definition 3 (**data flow model**) A data flow model is defined as a 4-tuple, $D = (I(V), O(V), R(E), X)$. Data input $I(V)$ and data output $O(V)$ are further defined as $I(V) = \{ I_0, I_1(V_1), I_2(V_2), \ldots, I_{n-1}(V_{n-1}), I_n(V_n)\}$ and $O=\{ O_0, O_1(V_1), O_2(V_2), \ldots, O_{n-1}(V_{n-1}), O_n(V_n)\}$, where I_i is a set of data items as input for task V_i and O_i is a set of data items as output for task V_i. I_0 is the set of data input and O_0 is the set of data output for the entire process. In addition, $I_0 \subseteq \{I_1(V_1), I_2(V_2), \ldots, I_{n-1}(V_{n-1}), I_n(V_n)\}$ and $O_0 \subseteq \{O_1(V_1), O_2(V_2), \ldots, O_{n-1}(V_{n-1}), O_n(V_n)\}$.

A data flow model specifies how the set of data that are consumed as input are transformed to the set of final output data by the tasks in a process and what data are produced as intermediate output under different conditions defined by a collection of rules. A data flow model should also provide ways for data flow verification, a procedure checking data flow anomalies (discuss later in this paper). In a data flow model, the smallest unit is data item, which is the same as the attribute in database models.

Definition 4 (**organizational model**) An organizational model is defined as a 5-tuple, $G = (A, L, H(L), S(A,L), F(V, L))$.

An organizational model specifies the hierarchical structure that groups job functions into different roles and associates actors with a role.

Definition 5 (**access control model**) An access control model is defined as a 4-tuple, $P = (L, V, I(V), O(V))$.

An access control model specifies what data items are accessible to different users. It connects to both organization model and data flow model in the way that the data set which is input for a particular task is accessible only to the actors who perform that task.

Figure 1 shows the four models and their relations to the five perspectives of a workflow model. These four models and their interrelations represent different perspectives of a workflow. Control flow models are the basis of data flow models because data input and output in a data flow model are defined for tasks and processes specified in a control flow model. Furthermore, data flow models require routing rules $R(E)$ identified in control flow models for data flow verification (see below). The interrelation between control flow models and organizational models are through $F(V, L)$ because the tasks identified in control flow models are assigned to the roles specified in organizational models. Organizational models and data flow models are interrelated through access control models. These four models together form a comprehensive workflow model.

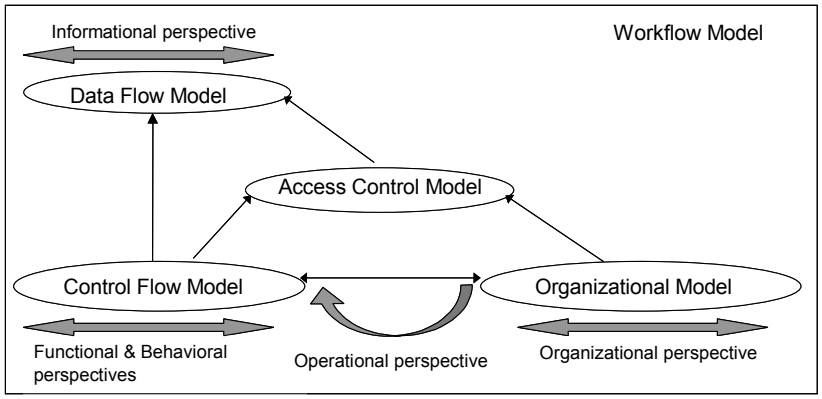

Figure 1. Formal Model of Workflow

AN EXAMPLE OF WORKFLOW

In this section, we describe an example of workflow, i.e. the *clinical trial workflow*, which we use to illustrate data flow problems. A clinical trial is a process of evaluating the effects and risks of promising approaches for disease prevention, diagnosis, and treatment and it is usually the final step of a long medical research process. It requires the collaboration of researchers, nurses, physicians, and statisticians from pharmaceutical companies, research institutions and hospitals, and therefore it involves a significant amount of data processing.

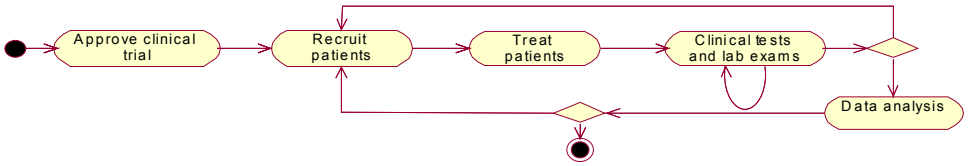

Figure 2. Control Flow Model of Clinical Trials

Figure 2 shows the control flow of a clinical trial at a high level. After a clinical trial protocol has been approved, research nurses recruit patients who meet the requirements. Once the participant patients receive the new treatments that are under investigation, several times of visit to the hospital are scheduled, in which the patients receive different clinical tests and lab exams. When enough data have been collected, the results are analyzed by using statistical procedures. If the conclusion cannot be drawn, then more patients are recruited. Otherwise the process ends.

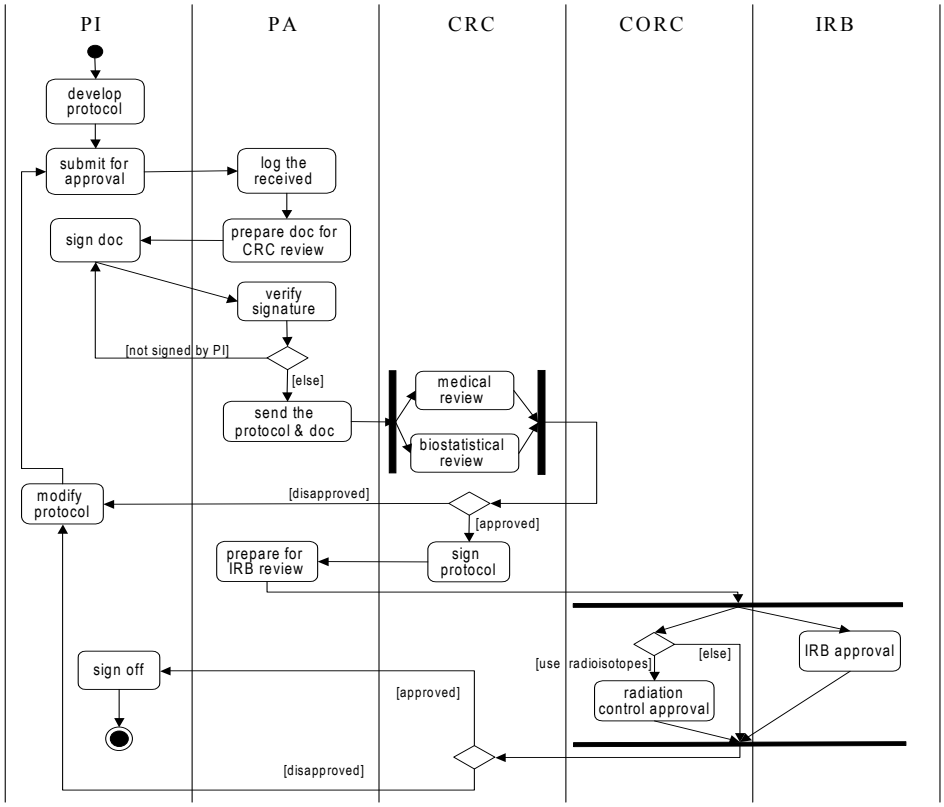

Figure 3. Control Flow Model for Clinical Trial Protocol Approval Sub-process

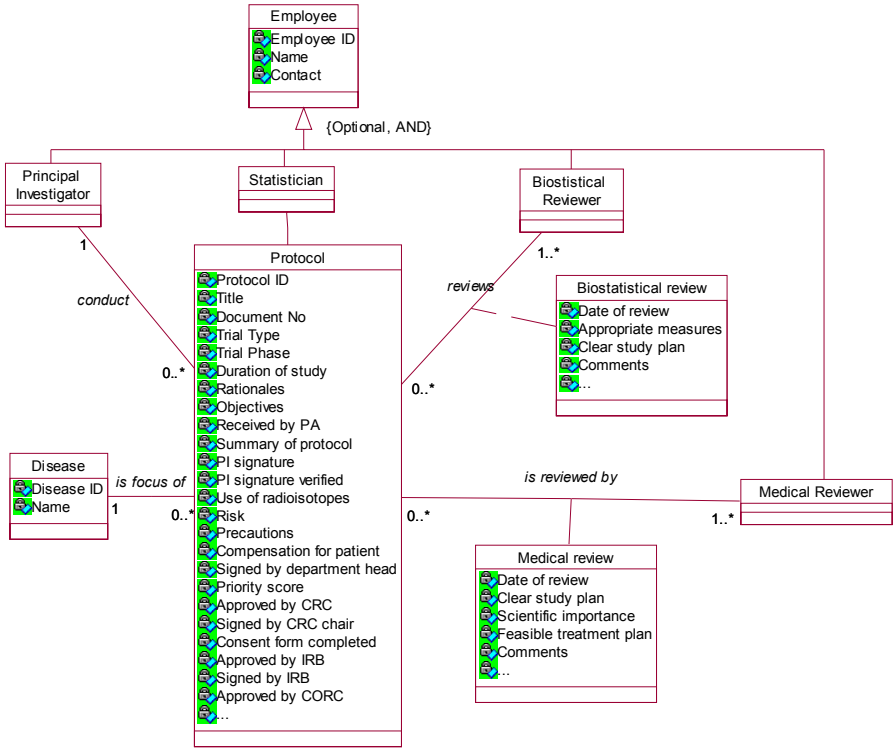

Figure 4. Class Diagram for Clinical Trial Protocol Approval Sub-process

Each task in Figure 2 can be decomposed to a sub-process. For example, the task of "approve clinical trial" in Figure 2 can be decomposed into the clinical trial approval sub-process in Figure 3. In this sub-process, if a principal investigator (PI) decides to conduct a protocol, he/she needs to finish the Clinical Research Committee (CRC) Transmittal Form and submit it with the protocol to a protocol assistant (PA) who helps prepare all required documents for CRC review. When all those documents are ready, the principal investigator needs to sign them. After the protocol assistant verifies the signature, the protocol together with all the required documents is sent to the Clinical Research Committee for review. If the Clinical Research Committee approves the protocol, the protocol is sent to both the Committee on Radiation Control (CORC) and the Institutional Review Board (IRB) for approval. In the case that either the Clinical Research Committee rejects the protocol or the Committee on Radiation Control and the Institutional Review Board disapprove the protocol, the PI needs to modify the protocol and go through the whole process again. Clearly, this sub-process involves a fair amount of data communication among different people.

In Figure 4, the class diagram provides a view of the data model for the clinical trial protocol approval sub-process. Most data transformed in this sub-process are the attributes of the Protocol class.

OPERATIONS ON DATA IN WORKFLOW

Data Flow Operations

Before we can formally define data flow anomalies, we define six types of data operations that a task can perform:

Initialize: this operation generates the initial value for a data item. If task *v* initializes data item *d*, then d is an output of *v* but not input of *v*. In the example of Figure 3, the principal investigator has to fill the name of the statistician who will analyze the data when he/she completes the CRC transmittal form in the task of "develop protocol". That is the first time in the entire process that the statistician is assigned. This scenario provides an example of the *initialize* operation.

Approve: this operation reviews the data input and decides if the data items meet certain requirements. If the data items meet the requirements, then the data items are approved. Otherwise they are rejected. The result of the decision can be the initial value of a new data item (e.g., IsApproved). If task *v* approves/disapproves data item *d*, *d* is the input of task *v*. In a lot of cases, the operation of *approve* also leads to the initialization of a second data item – signature. In Figure 3, the IRB decide whether to approve or reject a protocol. This decision leads to the initialization of two data items, approvedByIRB and signedByIRB.

Update: this operation changes the values of the data items that have been initialized. If task *v* updates data item *d*, *d* is both the input and the output of task *v*. One example from Figure 3 is that the principal investigator modifies the protocol based on the feedback from the reviewers in the task of "modify protocol". The protocol is both the input and the output for the task of "modify protocol".

Read: In some cases, a task uses a data item to determine the value of other data items. This is defined as *read* operation. If task *v* read data item *d*, *d* is the input for *v*. For instance, in the task of "prepare doc for CRC review" in Figure 3, the protocol assistant refers to the protocol for the preparation of other required documents. The protocol is used as a reference in this task and it is the input for creating other documents.

Verify: this operation checks the input and confirms that the data items are correct and the result of the checking is assigned to another data item (e.g., isVerified). If task *v* verifies data item *d*, *d* is the input of task *v*. This operation is different from the operation of *approve* because this operation checks whether a data item has been initialized and it also checks the correctness of the value whereas the *approve* operation is a decision making process. The task of "verify signature" in Figure 3 is a demonstration of this operation where the protocol assistant confirms the signature of the principal investigator on the documents.

Delete: this operation removes the values of data items. After this operation, the value of a data item is set to be null. If task *v* deletes data item *d*, *d* is the input of task *v*. The following scenario from Figure 3 can illustrate this operation. When the protocol is approved by the CRC but is disapproved by the IRB, the protocol has to be modified. Due to the modification, it has to be resubmitted to the CRC for approval. Before the resubmission, the values of some data items such as approvedByCRC and signedByCRC have to be removed.

Data Flow Matrices

We use set D to represent all data items in a workflow. Therefore, D is the union of $I(V)$ and $O(V)$, i.e. $D = I(V) \cup O(V)$. Given the total number of tasks n and the total number of data items z in a workflow, data flow matrix M is a $n \times z$ table where the $(nz)^{th}$ element shows the operation that the task v_n have on the data item d_z. We can represent the data flow matrix with the notation $<v_n | X | d_z>$ where $v_n \in V$ and $d_z \in D$. The tasks that do not involve data processing can be omitted. With a data flow matrix (See Table 1 for an example), we can easily find out how each data item is processed in a workflow.

Table 1 shows what operations each task has on a subset of data items in clinical trial approval sub process. In some cases, a task can take different data sets as input. For example, the task of "modify protocol" can take either the feedback from the CRC or the feedback from the IRB and the CORC. Therefore, in Table 1 we can have different data input under different conditions. Note that the numbers in the first row indicate tasks found in the control flow model in Figure 3, where 1. Develop protocol; 2. Submit for approval; 3. Log the received; 4. Prepare for CRC review; 5. Sign doc; 6. Verify signature; 7. Decision node; 8. Send the protocol & doc; 9.Medical review; 10. Biostatistical review; 11. Decision node; 12. Sign protocol; 13. Modify protocol; 14. Prepare for IRB review; 15 Decision node; 16. Radiation control approve; 17. IRB approve; 18 Decision node. The types of operations in Table 1 include I: initialize; R: read; U: update; A: approve; V: verify; D: delete. Furthermore, R1 refers to the data set in condition 1, and R2 refers to the data set in condition 2.

DATA FLOW ANOMALIES

We define the problems caused by incorrect data flow modeling as data flow anomalies. There are three types of data flow anomalies: missing data anomalies, redundant data anomalies and conflict data anomalies.

Missing Data Anomalies

For a data item $d \notin I_0$, if d is accessed before it is initialized, it causes a missing data anomaly. There are three possible causes of missing data anomalies. One possible case is when $d \notin I_0$ and $d \notin \{O_1(V_1), O_2(V_2), \ldots, O_{n-1}(V_{n-1})\}$ but $d \in I(V)$. In this situation, data item d has never been assigned an initial value within the process whereas some tasks try to use it. For instance, in Figure 3, if the protocol has never been written, no tasks can update, approve, verify, and delete the protocol.

Even if $d \notin I_0$ but $d \in \{O_1(V_1), O_2(V_2), \ldots, O_{n-1}(V_{n-1})\}$, missing data anomalies can still be introduced by incorrect data flow models. In another possible situation, suppose d is the output of task v_i and the input of task v_j. If v_i is designed to be executed after the execution of v_j or the precedence of the execution of v_i and v_j cannot be determined until run time, then missing data anomalies will occur. For instance, in the clinical protocol approval sub process in Figure 3, if the IRB needs to refer to the opinions of the Committee on Radiation Control before it makes decision on a protocol, then there is a missing data anomaly

Data Objects	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Protocol ID	I		R	R	R	R			A	A	R	R	R	R		A	A	
PI	I		R	R	R	R			A	A		R	R					
Protocol title	I		R	R	R	R			A	A		R	U	R		R	R	
Statistician	I		R	R						A			U					
Received By PA			I															
Summary of protocol				I	R				A	A		R						
Date of review				I	R				R	R		R	R1					
PI signature					I	V			R	R						R	R	
PI signature verified						I	R											
Medical reviewer name									R			R	R1					
Biostatistical reviewer name										R		R	R1					
Appropriate measures										I			R1					
Scientific importance									I				R1					
Feasible treatment plan									I				R1					
Priority score									I	I		R	R1					
Approved by CRC				D					I	I	R	R	R1	R			R	
Signed by CRC chair												I					R	
Use of radioisotopes														I	R	R		
Risk														I		R	R	
Signed by department head																R	R	
Approved by IRB													R2	D			I	R
Signed by IRB																	I	R
Approved by CORC													R2	D		I	R	

Table 1. Data Flow Matrix for Clinical Trial Approval Sub Process

because it is possible that the Committee on Radiation Control has not reviewed the protocol at the time when the IRB reviews the protocol.

In the third situation, if

- d is the output of task v_i and the input of task v_j
- v_i is to be executed only under a set of conditions R_i where $R_i \subseteq R(E)$
- v_j is to be executed only under a set of conditions R_j where $R_j \subseteq R(E)$
- $R_i \neq R_j$ and $\exists r$ ($r \in R_j$ but $r \notin R_i$)

then missing data anomalies can occur. In Figure 3, supposing the IRB needs to refer to the opinions of the Committee on Radiation Control even if the trial does not use any radiation, there exist missing data anomalies.

Redundant Data Anomalies

If the data item $d \in \{O_1(V_1), O_2(V_2), \ldots, O_{n-1}(V_{n-1}), O_n(V_n)\}$ or $d \in I_0$ but $d \notin \{I_1(V_1), I_2(V_2), \ldots, I_{n-1}(V_{n-1}), I_n(V_n)\}$ and $d \notin O_0$, a redundant data anomaly occurs. That is, when a data item is produced as an intermediate data output whereas no other tasks use it as input and it is not in the set of the final data output, then there is a redundant data anomaly. Redundant data anomalies may cause inefficiencies in that a process consumes resources to produce useless data. If all the data output from a task do not contribute to the production of the final data output in a process, it is likely that the task can be eliminated. In Figure 3, if the summary of the protocol produced by the protocol assistant in the tasks of "prepare doc for CRC review" is not used by any other tasks in the process and it is neither in the final data output of the entire process, then the summary of the protocol is redundant and can be eliminated.

Conflict Data Anomalies

If two tasks v_i and v_j attempt to initialize data item d at time T_i and T_j respectively, then it causes a conflict data anomaly no matter whether $T_i = T_j$ or $T_i \neq T_j$ as long as $v_i \neq v_j$. Any data item cannot be initialized more than once before it is deleted in a data flow. Conflict data anomalies lead to the problem that it is not possible to decide conceptually which value should be taken when those initial values do not agree with each other. Physically, the second initialization overwrites the first initialization.

In Figure 3, the tasks of both "medical review" and "biostatistical review" make decisions on whether to approve or reject the protocol. If medical reviewers approve the protocol whereas biostatistical reviewers reject the protocol, then it is impossible to make the final decision unless we have appropriate rules that define the way to draw a final conclusion when a conflict occurs.

DATA FLOW VERIFICATION

Data flow verification is the process of analyzing data flow and detecting data flow anomalies in workflow systems. In this section, we present the verification algorithms that consist of three parts as shown in Figure 5.

The following notations are used in the algorithm:

- P: control flow model
- s: start vertex
- e: end vertex
- v_i: task
- d: data item
- p: path;
- M: data flow matrix
- $I(V)$: the set of data input i.e. $I = \{I_0, I_1(v_1), I_2(v_2), \ldots, I_{n-1}(v_{n-1}), I_n(v_n)\}$
- $I_i(v_i)$: the set of data items as input for task v_i i.e. $I_i(v_i) = \{d_{i1}, d_{i2}, d_{i3},\ldots, d_{ip}\}$
- I_0: the original data input for the entire workflow
- $O(V)$: the set of data output i.e. $O = \{O_0, O_1(v_1), O_2(v_2), \ldots, O_{n-1}(v_{n-1}), O_n(v_n)\}$
- $O_i(v_i)$: the set of data items as output for task v_i i.e. $O_i(v_i) = \{d_{i1}, d_{i2}, d_{i3},\ldots, d_{iq}\}$
- O_0: the final data output for the entire workflow
- D_{ei}: the set of data items that have been detected data flow anomalies for task v_i
- D_e: the set of all data items that have been detected data flow anomalies
- V_e: the set of tasks where data flow anomalies happen

When checking the missing data for a task v in a workflow, if the input data are not in the starting data input set, the algorithm finds all the paths from the start node to v and for each path it traverses backward and examines whether the input data for v have been produced by any task executed before v. Checking redundant data anomalies is similar. The algorithm traverses forward from task v to the end node and examines whether the output data of v have been used by any task executed after v if the data are not in the final result. The conflict data checking algorithm really checks whether a data item has been initialized by more than one task in every path. The reason to check every path is that each path represents a workflow instance. The control flow model can help to find paths between two nodes. When we implement the data flow verification algorithms, the path searching algorithms from graph theory such as depth first search and breath-first search can be modified for traversing paths in the control flow models.

Missing data checking algorithm

Input: P, M
Output: D_e, V_e
$D_e = \varnothing$; $V_e = \varnothing$;
for i = 1 to n { /*check every task in a workflow*/
  $D_{ei} = \varnothing$;
  $I_i(v_i)$ = findInput(M, v_i) /*find the data input of task v_i from data flow matrix*/
  for each $d \in I_i(v_i)$ {
    if $d \in I_0$ removeElement($I_i(v_i)$, d); /*if d is in the original data input, then remove it from $I_i(v_i)$ */ }
  if ($I_i(v_i) \neq \varnothing$){ /*when I_0 does not provides every data input required by v_i */
    P_i = FindAllPaths(P, s, v_i) /*find all path from the start node to task v_i */
    for each $p \in P_i$ { /*traverse each path*/
      $I_{temp} = I_i(v_i)$;
      for each $v \in p$ {/*check each task on the path*/
        for each $d \in I_{temp}$ { /*check date input item
          if ($d \in O_j(v_j)$) removeElement(I_{temp}, d); /*remove d from I_{temp} */ } }
        if ($I_{temp} \neq \varnothing$) {
          append I_{temp} to D_{ei};
          if ($v_i \notin V_e$) insert v_i to V_e; } }
    if($D_{ei} \neq \varnothing$) append D_{ei} to D_e; } }

Redundant data checking algorithm

```
Input: P, M
Output: De, Ve
De = ∅; Ve = ∅;
for i = 1 to n { /*check every task in a workflow*/
    Oi (vi) = findOutput(M, vi) /*find the data output of task vi from data flow matrix*/
    Dei = Oi (vi);
     for each d∈ Oi (vi) {
            if d∈ O0 removeElement(Dei, d); /*if d is in the final data output, then remove it from Dei ;*/ }
    if (Dei ≠ ∅){  /*when O0 does not requires every data input produced by vi */
                  Pi = FindAllPaths(P, vi, e) /*find all path from task vi to the end node */
                  for each  p ∈ Pi { /*traverse each path*/
                      for each v ∈ p { /*check each task vj on the path*/
                          for each  d∈ Dei { /*check date output item
                              if (d∈ Ij(vj)) removeElement(Dei, d); /*remove d  from Dei */  } } } }
    if (Dei ≠ ∅) {
                   append Dei  to De;
                   insert vi to Ve; } }

Conflict data checking algorithm
Input: P, M
Output: De, Ve
De = ∅; Ve = ∅;
P_ALL = FindAllPaths(P, s, e) /*find all path from task s to the end node */
for each  p ∈ P_ALL { /*traverse each path*/
    Dtemp= ∅;
    for each v∈ P{/*check each task in path p*/
        for each d ∈ O(V) {/*check each data output item*/
            if (IsInitialization(M, v, d)) {/*check if task v initialize data item d*/
               if(d∉ Dtemp) insert d to Dtemp;
               else {
                    if(d∉  De)  insert d to De;
                    if(v∉  Ve)  insert v to Ve; }}
           else (IsDelete(M, v, d)) {/*check if task v delete data item d*/
                if (d∈  Dtemp) remove d from Dtemp; /*remove d from Dtemp if task v delete d*/ } } } }
```

Figure 5. Data Flow Verification Algorithm

There are several implications of data verification results. If missing data anomalies happen, it is possible that the precedence of tasks is incorrectly defined in the control flow models. If none of the data output from a task contributes to the production of the final data output in a process, it is likely that the task can be eliminated. Conflict data anomalies can be resolved either by applying rules to prioritize the tasks or modifying the control flow.

CONCLUSION

In this paper, we investigated the research issues in data flow modeling and verification. First, we proposed a comprehensive workflow model that consists of four sub-models. Furthermore, to capture the information flow perspective, we proposed a data flow modeling and verification framework to detect data flow anomalies in a syntactically correct workflow. With the proposed approach, it is possible to automate the verification of data flow anomalies. To the best of our knowledge, our work is the first attempt to develop a formal data flow verification methodology in workflow systems.

In the future, we plan to continue our work in two directions. First, we will extend the high-level algorithms for data flow verification presented in the paper by considering advanced issues such as database and form design. Second, we plan to develop and implement a data flow manager in a workflow system so that our research results can be tested in real world applications.

REFERENCES

1. Bajaj, A. and Ram, S. (2002) Seam: A state-entity-activity-model for a well-defined workflow development methodology, *Knowledge and Data Engineering, IEEE Transactions on,* 14, 2, 415-431.
2. Basu, A. and Blanning, R. W. (2000) A formal approach to workflow analysis, *Information Systems Research,* 11, 1, 17-36.
3. Basu, A. and Kumar, A. (2002) Workflow management issues in e-business, *Information Systems Research,* 13, 1, 1-14.

4. Bi, H. H. and Zhao, J. L. (2004) Applying propositional logic to workflow verification, *Information Technology and Management: Special Issue on Workflow and E-business,* Forthcoming.
5. Curtis, B., Kellner, M. I. and Over, J. (1992) Process modeling, *Communications of the ACM,* 35, 9, 75-90.
6. Georgakopoulos, D., Hornick, M. and Sheth, A. (1995) An overview of workflow management: From process modeling to workflow automation infrastructure, *Distributed and Parallel Database,* 3, 119-153.
7. Stohr, E. A. and Zhao, J. L. (2001) Workflow automation: Overview and research issues, *Information Systems Frontiers,* 3, 3, 281-296.
8. van der Aalst, W. (1998) The application of petri nets to workflow management, *The Journal of Circuits Systems and Computers,* 8, 1, 21-66.
9. van der Aalst, W. and van Hee, K. (2002) *Workflow management: Models, methods, and systems,* The MIT Press, Cambridge, Massachusetts, London, England.
10. Yourdon, E. and Constantine, L. L. (1979) *Structured design,* Prentice Hall, Englewood Cliffs, NJ.