Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2005 Proceedings

Americas Conference on Information Systems (AMCIS)

2005

On The Theoretical Foundation for Data Flow Analysis in Workflow Management

Sherry X. Sun
University of Arizona, xiaoyun@eller.arizona.edu

J. Leon Zhao
University of Arizona, lzhao@eller.arizona.edu

Jay F. Nunamaker, Jr.
University of Arizona, nunamaker@cmi.arizona.edu

On the Theoretical Foundation for Data Flow Analysis in Workflow Management

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

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

Jay F. Nunamaker
Department of MIS, University of Arizona
nunamaker@eller.arizona.edu

ABSTRACT

In workflow management, the data flow perspective specifies how data are produced and consumed by activities in a workflow. Data flow analysis can detect data flow anomalies occurring in a workflow while its control flow can be syntactically error-free. Currently, most commercial workflow management systems do not provide the tools for data flow analysis at design time. We have previously proposed a data flow analysis approach and developed the basic concepts and the essential algorithms. As another step forward, this paper examines the issues of data flow anomalies and their verification from a theoretical point of view and validates the correctness of the proposed approach.

Keywords

workflow modeling, data flow verification, data flow anomalies, data flow modeling, business process management

1. INTRODUCTION

Business processes have been considered as invaluable corporate assets, and consequently, corporations are constantly trying to achieve performance improvement through better business process management. As the technology for managing complex business processes, workflow systems enable automatic routing, monitoring, and coordination of business processes. To effectively implement workflow systems, workflow modeling and analysis has become a critical part of corporate information technology.

Workflow models are used to coordinate a collection of activities designed to achieve some business objectives. Current paradigms focusing on modeling the control and coordination of activities, i.e. control flow perspective, include Petri nets and its variants (van der Aalst, 1998; van der Aalst and van Hee, 2002) and activity-based workflow modeling (Bi and Zhao, 2004; Georgakopoulos, Hornick and Sheth, 1995). However, given a syntactically correct control flow model, errors can still occur in the workflow specification due to data flow irregularities. A workflow specification that contains data flow errors may cause process interruption and high cost to debug and fix at the runtime. Hence, data flow analysis is a critical step in the workflow management. The primary purpose of data flow analysis is to prevent unintentional errors or conflicts and to maintain data integrity in a workflow (Basu and Kumar, 2002).

To model and analyze data and data flow in workflow systems, several informal and formal modeling tools have been proposed (Bajaj and Ram, 2002; Basu and Blanning, 2000; Kappel, Lang, Rausch-Schott, and Retschitzegger, 1995; Reuter and Schwenkreis, 1995). However, none of these paradigms has focused on discovering data flow errors in a workflow model. More recently, different types of data flow errors have been investigated in workflow management (Sadiq, Orlowska, Sadiq and Foulger, 2004; Sun, Zhao, and Sheng, 2004). As the first method of analyzing data flow anomalies in workflow, we have previously proposed a data flow analysis approach by developing the basic concepts and the essential algorithms (Sun et al., 2004). However, there is a need for a theoretical foundation of data flow analysis in order to examine the validity of the key concepts and algorithms.

In this paper, we extend our previous work by theoretically proving the correctness of our approach. In order to lay the foundation, we also introduce a number of important new concepts such as activity dependency, decision variables, and decision constraints. Next, we introduce a workflow example that is used to illustrate the critical issues in data flow analysis

In section 3, we define various data flow anomalies. Section 4 presents a dependency–based approach for data flow analysis. Finally, section 5 concludes the paper by summarizing the contribution of this paper and indicating the future research directions.

2. AN EXAMPLE OF WORKFLOW

This section introduces a property loan approval workflow, which is used to illustrate the data flow approach we propose. Next, we examine the key steps of this workflow as shown in the UML activity diagram in Figure 1.

To determine the qualification of the applicants, the financial service first verifies the applicants' employment status, credit history, and liquid asset after it receives a loan application. If the applicants are qualified for the loan application, the current interest rate is locked for a certain period of time as required by the applicants. After the appraisal information is received, the loan application is evaluated again. Given the applicants' credit score, the appraised value of the property, and the loan amount, the level of risk associated with the loan is calculated. If the risk is higher than a threshold, the loan amount must be adjusted. Eventually, the financial service contacts the applicants and discusses with them the necessary adjustment and other conditions such as property insurance options. In case the applicants disagree with the conditions and adjustment, the workflow ends. If the applicants agree on everything, then the application is forwarded to the loan officer for signature after the applicants sign the applications. When the loan amount is more than half million, the manager's signature is required. Table 1 contains the symbols for the activities and data items in this workflow.

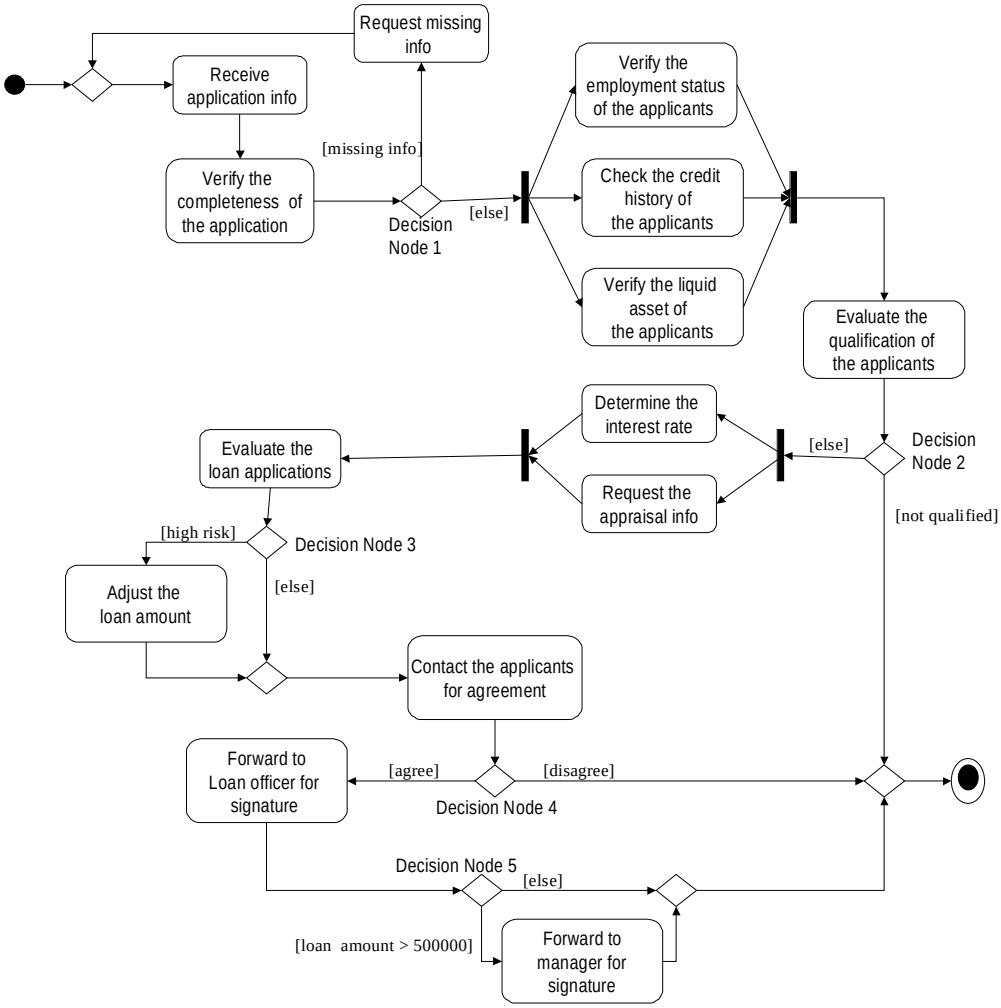

Figure 1. Property Loan Approval Workflow

As shown in Table 2, we use a data flow matrix (Sun et al., 2004) to specify the data flow in the property loan approval workflow. Basically, a data flow matrix is a two-dimension table that records the data flow operations each activity performs

on different data items in a workflow. From database implementation point of view, all data flow operations can be considered as *read* or *write* operation regardless of their semantic meanings. When we analyze data flow to discover data flow irregularities, we focus on the input and output data for each activity. Categorizing the data flow operations into *read* / *write* operations help identify the input and output data for each activity. It is intuitive that when an activity *v* reads a data item *d*, *d* is the input data for *v*, and when *v* performs a write operation on *d*, *d* is the output data from *v*.

Moreover, there are two types of *write* operation, *write* the initial value, also called initialization, and overwrite. It is possible that a data item is overwritten after it is initialized. However, for the purpose of simplicity, the current paper only focuses on the initial *write* operation because initialization is critical for discovering data flow anomalies (see Section 3). Note that under different routing conditions, a data item can be initialized by different activities.

Data items		
d_1: Applicant name	d_{11}: Credit score	d_{21}: Appraised value of the property
d_2: Applicant's address	d_{12}: Bank name	d_{22}: Risk
d_3: Social Security No.	d_{13}: Account balance	d_{23}: Amount adjusted
d_4: Date of birth	d_{14}: Account balance verified	d_{24}: Agreed by applicants
d_5: Loan amount	d_{15}: Applicant qualified	d_{25}: Property insurance
d_6: Employer name	d_{16}: Closing date	d_{26}: Signed by applicants
d_7: Employer phone No.	d_{17}: Type of the loan	d_{27}: Signed by loan officer
d_8: Annual incoming	d_{18}: Interest rate	d_{28}: Signed by manager
d_9: Application complete	d_{19}: Interest lock-in time periods	
d_{10}: Employment status verified	d_{20}: Property address	
Activities		
v_1: Receive application	v_8: Evaluate the qualification of the applicant	v_{15}: Contact applicants for agreement
v_2: Verify the completeness of application	v_9: Decision node 2	v_{16}: Decision node 4
v_3: Decision node 1	v_{10}: Determine the interest rate	v_{17}: Forward to loan officer for signature
v_4: Request missing info	v_{11}: Request appraisal info	v_{18}: Decision node 5
v_5: Verify the employment status	v_{12}: Evaluate the loan application	v_{19}: Forward to manager for signature
v_6: Verify the credit history	v_{13}: Decision node 3	*s*: Start node
v_7: Verify the liquid asset	v_{14}: Adjust the loan amount	*e*: End node
Operations		
R: Read	*W*: Write	

Table 1. Symbols Used in the Property Loan Approval Workflow

3. DATA FLOW ANOMALIES

A workflow can start with a set of initial input data items and end with a set of data items as final output. The activities in the workflow contribute to the production of final output data by generating either some intermediate output data set or a subset of the final output data. If the data flow is not specified correctly in a workflow system, errors and conflicts can occur. The errors and conflicts caused by an incorrect data flow specification are referred as data flow anomalies (Sadiq et al, 2004; Sun, et al., 2004).). Data flow anomalies can be classified into three categories: missing data, redundant data, and conflict data. In this section, we discuss the three categories of data flow anomaly.

3.1. Missing Data

When a data item is accessed before it is initialized, a missing data anomaly occurs. Each of the following scenarios can cause missing data anomalies.

Scenario 1. (Absence of Initialization) A data item has never been assigned an initial value within a workflow whereas some activities use it as input or it is required as the final output of the workflow.

Scenario 2. (Delayed Initialization) A data item is used by an activity *v* as input whereas it is initialized by another activity executed after *v*.

Scenario 3. (Uncertain Synchronization) Two activities *v* and *u* are executed in parallel whereas *v* needs an input data item initialized by *u*. At the time of the execution of *v*, the data item may not be initialized.

Scenario 4. (Improper Routing) Under certain workflow routing conditions, a data item is not initialized whereas it is still used by some activities as input.

Data Objects	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}	V_{19}
d_1: Applicant name	W	R			R	R	R	R		R	R	R		R	R		R		R
d_2: Applicant's address	W	R				R	R	R							R				
d_3: Social Security No.	W	R			R	R	R	R		R		R		R			R		R
d_4: Date of birth	W	R			R		R	R											
d_5: Loan amount	W	R						R							R		R	R	R
d_6: Employer name	W	R			R			R											
d_7: Employer phone No.	W	R			R														
d_8: Annual incoming	W	R			R			R											
d_9: Application complete		W	R	R				R							R				
d_{10}: Employment status verified					W			R											
d_{11}: Credit score						W		R				R							
d_{12}: Bank name	W						R	R											
d_{13}: Account balance	W						R	R											
d_{14}: Account balance verified							W	R											
d_{15}: Applicant qualified								W	R	R		R			R		R		R
d_{16}: Closing date	W									R		R					R		R
d_{17}: Type of the loan										R		R		R			R		R
d_{18}: Interest rate										W		R		R			R		R
d_{19}: Interest lock-in time period										W		R		R			R		R
d_{20}: Property address											R	R							
d_{21}: Appraised value of the property											W	R		R					
d_{22}: Risk												W	R						
d_{23}: Amount adjusted										R				W	R		R	R	R
d_{24}: Agreed by applicant															W	R	R		R
d_{25}: Property insurance												R			R				
d_{26}: Signed by applicants															W	R	R		R
d_{27}: Signed by loan officer																	W		
d_{28}: Signed by manager																			W

Table 2. Data Flow Matrix for the Property Loan Approval Workflow

3.2. Redundant Data

If an activity produces data items that do not contribute to the production of the final output data, then there is a redundant data anomaly. Redundant data cause inefficiency. Either of the following scenarios can cause redundant data anomalies.

Scenario 5. (Inevitable Redundancy) A data item is produced as an intermediate data output whereas no other activities need it as input data and it is not required as the final output data.

Scenario 6. (Contingent Redundancy) A data item is produced as an intermediate data output. However, it is only used under some routing conditions. Under other routing conditions, it is not used by any activities as input.

3.4. Conflict Data

In a workflow instance, if there exist different versions of the same data item, conflict data anomalies occur. The following scenario can cause conflict data anomalies.

Scenario 7. (Multiple Initializations) More than one activity attempts to initialize the same data item in one workflow instance.

There are also other types of data flow anomalies such as mismatched data and insufficient data (Sadiq, et al., 2004; Sun et al., 2004). Within the scope of this paper, we only focus on the anomalies defined above.

4. ACTIVITY DEPENDENCY ANALYSIS FOR DATA FLOW VERIFICATION

In this section, we propose a methodology for analyzing data flow and detecting data flow anomalies in workflow systems based on activity dependency analysis.

4.1. Basic Concepts

A routing constraint defines how a workflow instance is routed according to some business rules. In most cases, a routing constraint can be described based on the values of input and output data items. Next, we define the concept of workflow routing constraint.

Definition 1 (Decision Variable) A routing decision can be made based on a set of input data items. Each of the data item is called a decision variable, denoted as d^c. The set of possible values d^c can have is called its valuation domain.

In the property loan approval workflow, the decision node 1 uses the data item d_9, *application complete*, to route a workflow instance (Table 2). Therefore d_9 is a decision variable. The set of possible values for d_9 is ["*Yes*", "*No*"].

Definition 2. (Routing Constraint): A workflow routing constraint r is defined as a logic formula used to route a workflow instance. In this logic formula, a set of decision variables is quantified.

A decision node can use a set of routing constraints to route a workflow instance, with each routing constraint corresponding to one are leaving from the decision node. For example, in the property loan approval workflow, when d_{24}= "*Yes*" and d_{26}="*Yes*", namely the loan conditions are agreed and signed by the applicants, the decision node 4 routes the application to the loan officer for signature. When d_{24}= "*No*" and d_{26}="*No*" the decision node 4 routes the application to the end of the workflow. Therefore, the routing constraint set used by the decision node 4 is

R={r_1=(d_{24}= "*Yes*" and d_{26}="*Yes*"), r_2=(d_{24}= "*No*" and d_{26}="*No*")}.

Definition 3. (Routing Constraint Set for Workflow): The workflow routing constraint set R for workflow W is defined as the complete set of routing constraints R_w that workflow W uses to route all the instances.

There are a total of seven decision variables in the property loan approval workflow, d_5, d_9, d_{15}, d_{22}, d_{23}, d_{24}, and d_{26}, the combined values of which determine a workflow instance. Therefore, the routing constraint set for this workflow can be written as

R_w={$d_5 \geq 0$, d_9="*Yes*" or "*No*", and d_{15} = "*Yes*" or "*No*", and d_{22} = "*Low*" or "*High*", and $d_{23} < d_5$ or be null, and d_{24}= "*Yes*" or " *No*", and d_{26}="*Yes*" or *No*"}.

Definition 4. (Upstream Routing Constraint Sets for Activities): The upstream routing constraint set R^u for activity v is defined as the set of routing constraints for a workflow W to execute v.

As a simple example, Table 3 shows the upstream routing constraint sets (R^u) for the activities in the workflow shown in Figure 2, which consist of activities *A*, *B*, *C*, *D*, and *E*, decision nodes *1*, *2*, and *3*, and routing constraints r_1, r_2, r_3, r_4, r_5, r_6, and r_7.

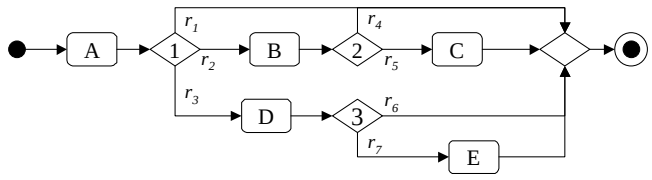

Figure 2. A simple workflow

Activity	R^u
A	∅
B	{r_2}
C	{r_2, r_5}
D	{r_3}
E	{r_3, r_7}

Table 3. Upstream Routing Constraint Sets

In a workflow, an activity depends on data produced by other activities, leading to activity dependencies. Next, we define two basic concepts, data dependency and activity dependency.

Definition 5 (Unconditional and Conditional Data Dependency) An unconditional data dependency for activity v is denoted as $\lambda^u_v(I, O)$. The first element I is the set of data items that activity v requires as input under all routing constraints and I = [i_1, i_2,…, i_n] where i_p, $p \in [1, ..., n]$, is an input data item for activity v. The second element O is the set of data items that activity v produces as output and O = [o_1, o_2,…, o_l] where o_k, $k \in [1, ..., l]$, is an output data item from activity v. There exists an unconditional dependency between I and O for activity v. Similarly, a conditional data dependency for activity v is denoted as $\lambda^c_v(I, O)$. I is referred as the conditional input data set and activity v requires I to produce O only under certain conditions.

For example, there is a conditional data dependency for the activity *Contact the applicants for agreement & signature* in the property loan approval workflow where the input data item *amount adjusted* of the loan may be null depending on the risk level. When the risk level is low, activity *adjust the loan mount* is not activated. Therefore, the data item *amount adjusted* is

not initialized under this condition. However, the activity *Contact the applicants for agreement & signature* can still be activated. Hence, there is a conditional dependency on the data item *amount adjusted*.

Table 4 shows both unconditional and conditional data dependencies in the property loan approval workflow.

Unconditional data dependencies λ^u_{v1} (ϕ, [d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_{12}, d_{13}, d_{16}]) λ^u_{v2} ([d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8],[d_9]) λ^u_{v3} ([d_9], ϕ) λ^u_{v4} ([d_9], ϕ) λ^u_{v5} ([d_1, d_3, d_4, d_6, d_7, d_8], [d_{10}]) λ^u_{v6} ([d_1, d_2, d_3], [d_{11}]) λ^u_{v7} ([d_1, d_2, d_3, d_4, d_{12}, d_{13}], [d_{14}]) λ^u_{v8} ([d_1, d_2, d_3, d_4, d_5, d_6, d_8, d_9, d_{10}, d_{11}, d_{12}, d_{13}, d_{14}], [d_{15}]) λ^u_{v9} ([d_{15}], ϕ) λ^u_{v10} ([d_1, d_3, d_{15}, d_{16}, d_{17}], [d_{18}, d_{19}])	λ^u_{v11} ([d_1, d_{20}], [d_{21}]) λ^u_{v12} ([d_1, d_3, d_{11}, d_{15}, d_{16}, d_{17}, d_{18}, d_{19}, d_{20}, d_{21}], [d_{22}]) λ^u_{v13} ([d_{22}], ϕ) λ^u_{v14} ([d_1, d_3, d_{17}, d_{18}, d_{19}, d_{21}], [d_{23}]) λ^u_{v15} ([d_1, d_2, d_5, d_9, d_{15}, d_{25}], [d_{24}, d_{26}]) λ^u_{v16} ([d_{24}, d_{26}], ϕ) λ^u_{v17} ([d_1, d_3, d_5, d_{15}, d_{16}, d_{17}, d_{18}, d_{19}, d_{24}, d_{26}], [d_{27}]) λ^u_{v18} ([d_5], ϕ) λ^u_{v19} ([d_1, d_3, d_5, d_{15}, d_{16}, d_{17}, d_{18}, d_{19}, d_{24}, d_{26}], [d_{28}]) λ^u_e ([d_{15}], [d_{15}])
Conditional data dependency λ^c_{v10} ([d_{23}], [d_{18}, d_{19}]) λ^c_{v15} ([d_{23}], [d_{24}, d_{26}]) λ^c_{v17} ([d_{23}], [d_{27}])	λ^u_{v18} ([d_{23}], ϕ) λ^u_{v19} ([d_{23}], [d_{28}]) λ^u_e ([d_{22}, d_{23}, d_{24}, d_{25}, d_{26}, d_{27}, d_{28}], [d_{22}, d_{23}, d_{24}, d_{25}, d_{26}, d_{27}, d_{28}])

Table 4. Data Dependencies for Property Loan Approval Workflow

Definition 6 (Activity dependency): Given two activities v and u, if $\exists d$ that $d \in O_u$ and $d \in I_v$, where O_u is the output data set of u and I_v is the input data set of v, then v is dependent on u through d, denoted as $u \Rightarrow v$. Furthermore, activity dependency follows the transitive law, *i.e.* if $x \Rightarrow v$ and $u \Rightarrow x$, then $u \Rightarrow v$. If u provides unconditional input data set for activity v, v is unconditionally dependent on u. If u provides conditional input data set for activity v, v is conditionally dependent on u. If there is no any activity dependency between two activities u and v, we denote the non-dependency between the two activities as $u \infty v$.

Definition 7 (Unconditional and Conditional Requisite set) A set of activities Δ^u is the unconditional requisite set for activity v if for any activity $x \in \Delta^u$, there exists a data item d such that $d \in O_x$ and $d \in I_v$ where I_v is the unconditional input data set of activity v and O_x is the output data set of x. Similarly, a set of activities Δ^c is the conditional requisite set for activity v if for any activity $x \in \Delta^c$, there exists a data item d such that $d \in O_x$ and $d \in I_v$ where I_v is the conditional input data of activity v under all possible routing constraints and O_x is the output data set of x.

Activity dependencies can be derived from data dependencies. Table 5 shows the requisite sets for activities in the property loan approval workflow. From Table 5 we know that most activities depend on activity v_1, *receive application*, to provide necessary input data. Moreover, activities v_{10}, v_{15}, v_{17}, v_{18}, and v_{19} depend on activity v_{14}, *namely adjust the loan amount*, conditionally.

Definition 8 (Instance Set) The set of activities Γ is an instance activity set of workflow W if when a set of workflow routing constraints R from the complete set R_w is satisfied, all the activities in Γ and only the activities in Γ are executed in a specified order from the start activity s of W to the end activity e of W. R is called the routing constrain set of Γ.

Table 6 shows some examples of instance activity set for the property loan approval workflow. Each instance activity set is the set of activities that are executed in one workflow instance. For example, when a complete application is received but the applicant is not qualified, only the activities in the instance set Γ_1 are executed. Therefore the routing constraint set for Γ_1 can be expressed as follows R_1={r_1=($d_5 \geq 0$ and d_9="*Yes*"), r_2=(d_{15}="*No*")}.

Activities	Unconditional Requisite Set	Conditional Requisite Set
v_1	$\varnothing \Rightarrow v_1$	
v_2	$\{v_1\} \Rightarrow v_2$	
v_3	$\{v_2\} \Rightarrow v_3$	
v_4	$\varnothing \Rightarrow v_4$	
v_5	$\{v_1\} \Rightarrow v_5$	
v_6	$\{v_1\} \Rightarrow v_6$	
v_7	$\{v_1\} \Rightarrow v_7$	
v_8	$\{v_1, v_2, v_5, v_6, v_7\} \Rightarrow v_8$	
v_9	$\{v_8\} \Rightarrow v_9$	
v_{10}	$\{v_1, v_8\} \Rightarrow v_{10}$	$\{v_{14}\} \Rightarrow v_{10}$
v_{11}	$\{v_1\} \Rightarrow v_{11}$	
v_{12}	$\{v_1, v_6, v_8, v_{10}, v_{11}\} \Rightarrow v_{12}$	
v_{13}	$\{v_{12}\} \Rightarrow v_{13}$	
v_{14}	$\{v_1, v_{10}, v_{11}\} \Rightarrow v_{14}$	
v_{15}	$\{v_1, v_2, v_8\} \Rightarrow v_{15}$	$\{v_{14}\} \Rightarrow v_{15}$
v_{16}	$\{v_{15}\} \Rightarrow v_{16}$	
v_{17}	$\{v_1, v_8, v_{10}, v_{15}\} \Rightarrow v_{17}$	$\{v_{14}\} \Rightarrow v_{17}$
v_{18}	$\{v_1\} \Rightarrow v_{18}$	$\{v_{14}\} \Rightarrow v_{18}$
v_{19}	$\{v_1, v_8, v_{10}, v_{15}\} \Rightarrow v_{19}$	$\{v_{14}\} \Rightarrow v_{19}$

Table 5. Activity Dependencies for the Property Loan Approval Workflow

4.2. Data Flow Verification Rules

Data flow verification is the process of analyzing data flow and detecting data flow anomalies in workflow systems. In this section, we provide data flow verification rules that are based on dependency analysis and formally prove these rules.

$\Gamma_1=\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, e\}$

$\Gamma_2=\{v_1, v_2, v_3, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}, v_{15}, v_{16}, e\}$

$\Gamma_3=\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{15}, v_{16}, v_{17}, v_{18}, v_{19}, e\}$

Table 6. Instance Sets for the Property Loan Approval Workflow

Lemma 1 (Condition for Absence of Initialization) Given $d \notin I_0$, $d \notin \cup O_i$, i=(1, 2, …, n-1), and $d \in I_i$, i=(1, 2, …, n), a missing data anomalies occur in at least one instance of the workflow W. I_i is the set of input data items for activity v_i and $\cup O_i$ is the union of all the output data items from activities in W. I_0 is the set of initial input data for the entire workflow.

Lemma 2. (Condition for Delayed Initialization) Given data item $d \in O_v$, $d \in I_u$, i.e. $v \Rightarrow u$, and $\exists \Gamma$ such that $v \in \Gamma$, $u \in \Gamma$, and u proceeds v, a missing data anomaly occurs. O_v is the set of output data items for activity v and I_u is the set of input data items for activity u.

Lemma 3. (Condition for Uncertain Synchronization) Given data item $d \in O_v$ and $d \in I_u$, i.e. $v \Rightarrow u$, and the precedence of activity v and activity u cannot be determined until run time, missing data anomalies can occur. O_v is the set of output data items for activity v and I_u is the set of input data items for activity u.

Lemma 4. (Condition for Improper Routing) Given data item $d \in O_x$, $d \in I_y$,, i.e. $x \Rightarrow y$, and $R^u_x \neq R^u_y$, if $\exists \beta$, $\beta \in R^u_x$ but $\beta \notin R^u_y$, a missing data anomaly occurs. O_x is the set of output data items of activity x and I_y is the set of input data items of activity y. R^u_x and R^u_y are the upstream routing constraint sets for x and y, respectively. β is a routing constraint in workflow W.

Discussion: A routing constraint decreases the possibility for an activity to be executed. Given $x \Rightarrow y$, if $\exists \beta$ such that y is executed whereas x is not, missing data anomaly occurs.

Lemma 5. (Finite Instance Sets) A workflow W with a finite set of activities can only have a finite number of instance activity sets.

Theorem 1. (Missing Data Verification) A workflow W is free from missing data anomalies if the following conditions are satisfied: (1) $\forall d$ that $d \in \cup I_i$, i=(1, 2,…, n), $d \in I_0 + \cup O_i$, i=(1, 2,…, n-1), where $\cup O_i$ and $\cup I_i$ are the union of the output and input data sets of all the activities in W, respectively. I_0 is the set of initial input data for the entire workflow, and d is an data item; (2) Given activity dependency $u \Rightarrow v$, $\forall \Gamma$ that $v \in \Gamma$, u$\in \Gamma$ and u precedes v at least once under the routing constrain set R of Γ, where Γ is an instance set of W, u and v are two activities in W.

Proof: We believe Lemmas 1, 2, 3, and 4 are the only conditions under which missing data anomalies cane occur. We use enumeration to prove that a workflow W will avoid these four situations if it satisfies the two conditions: 1) $\forall d$ that $d \in \cup I_i$, i=(1, 2,…, n), $d \in I_0 + \cup O_i$, i=(1, 2,…, n-1), and 2) Given activity dependency $u \Rightarrow v$, $\forall \Gamma$ that $v \in \Gamma$, u$\in \Gamma$ and u precedes v at least once under the routing constrain set R of Γ.

1) When $\forall d$ that $d \in \cup I_i$, $d \in I_0 + \cup O_i$, namely every input data item d has the initial value in the workflow W, either $d \in I_0$ or $d \in \cup O_i$. Therefore, W is free from Absence of Initialization according to Lemma 1.

2) We prove that W is free from Delayed Initialization and Uncertain Synchronization through contradiction. Suppose that W has Delayed Initialization or Uncertain Synchronization when the set of routing constrains R is satisfied. Since W has a missing data anomaly under R, we can find two activities v and u in Γ, such that v depends on activity u for some dataset D, i.e. $u \Rightarrow v$, and the precedence of u and v can be either v precedes u (Lemma 2) or not determined until run time (Lemma 3). In either case, u does not precede v. This contradicts with our assumption. Hence when R is satisfied, W is free from Delayed Initialization or Uncertain Synchronization. By lemma 5, there are a finite number of instance activity sets in W. Since for each instance activity set Γ of W, if there exists an activity dependency between two activities v and u, then u precedes v, we can conclude W is free from Delayed Initialization or Uncertain Synchronization under all the routing conditions.

3) From the condition we also know that for each set of routing constraints, if two activities u and v have dependency $u \Rightarrow v$, then u precedes v. Hence, we cannot find a set of routing constraints under which v is executed whereas u is not when u and v have dependency $u \Rightarrow v$. Therefore, W is free from Improper Routing. Hence, theorem 1 holds. ♦

Lemma 6. (Condition for Inevitable Redundancy) If the following inequality holds: $\cup O_i + I_0 - \cup I_i - O_0 \neq \varnothing$, redundant data anomalies occur in at least one instance of the workflow W. $\cup O_i$, i=(1, 2,…, n-1), and $\cup I_i$, i=(1, 2,…, n), are the union of the output and input data sets of all the activities in W, respectively. I_0 is the set of initial input data for the entire workflow and O_0 is the set of final output data produced by the entire workflow.

Lemma 7. (Condition for Contingent Redundancy) Given $\cup O_i + I_0 - \cup I_i - O_0 = \varnothing$, and $\exists \Gamma$ that satisfies $\cup O_i + I_0 - \cup I_i - O_0 \neq \varnothing$, a redundant data anomaly occurs. $\cup O_i$, i=(1, 2,…, n-1), and $\cup I_i$, i=(1, 2,…, n), are the union of the output and input data sets of all the activities in W, respectively. I_0 is the set of initial input data for the entire workflow and O_0 is the set of final output data produced by the entire workflow.

Theorem 2 (Redundant Data Verification) A workflow W is free from redundant data anomalies if $\forall \Gamma$: $\cup O_i + I_0 - \cup I_i - O_0 = \varnothing$. $\cup O_i$ and $\cup I_i$ are the union of the output and input data sets of all the activities in W, respectively. I_0 is the set of initial input data for the entire workflow and O_0 is the set of final output data produced by the entire workflow.

Proof: Since$\forall \Gamma$: $\cup O_i + I_0 - \cup I_i - O_0 = \varnothing$, in each instance every output data item is used as input or required as the final output. Therefore, by Lemma 6, W is free from Inevitable Redundancy and by Lemma 7 W is free from Contingent Redundancy ♦

Lemma 8. (Condition for Multiple Initializations) Given $O_x \cap O_y \neq \varnothing$, and $R^u_x \subseteq R^u_y$ or $R^u_y \subseteq R^u_x$ a conflict data anomaly occurs, where O_x and O_y are the output data from the two different activities x and y, respectively, and R^u_x and R^u_y are the upstream routing constraint set for x and y, respectively.

Discussion: Given that two activities x and y attempt to initialize the same data item d and when x is executed, y is also executed or *vice versa*, a conflict data anomaly occurs.

Theorem 3. (Conflict Data Verification) A workflow W is free from conflict data anomalies if the following condition holds: given $O_x \cap O_y \neq \varnothing$, $\forall \Gamma$ if $x \in \Gamma$, then y$\notin \Gamma$, where O_x and O_y are the output data from the two different activities x and y, respectively, and Γ is an instance set of W.

Proof: Given $O_x \cap O_y \neq \varnothing$, x and y initialize the same data item. However, x and y are executed in different Γ since $\forall \Gamma$ if $x \in \Gamma$, then y$\notin \Gamma$. By Lemma 8, we know W can avoid conflict data anomalies. ♦

We use the property loan application workflow to intuitively explain how data flow anomaly theorems work. We can detect missing data in the property loan application workflow for the following reasons. First, as shown Table 2, no activity initializes d_{17}, *type of the loan*, but activities v_{10}, v_{12}, v_{14}, v_{17}, and v_{19} use d_{17} as input, i.e. $d_{17} \in \cup I_i$, i=(1,…,19), whereas $d_{17} \notin \cup O_i$, i=(1,…,18), which violates Theorem 1. Second, Table 5 shows that $v_{14} \Rightarrow v_{10}$ whereas the control flow in Figure 1 shows activity v_{10} is executed before activity v_{14} in the instance set Γ_2 (Table 6). This is also a violation of Theorem 1. The examination of Table 2 and all the instance sets shows that there is no violation of Theorem 2, that is for each instance set each output data item is either used by some activities or generated as the final output. The examination also shows no more than one activity in each instance set initializes the same data item. Therefore, this workflow does not violate Theorem 3. According to Theorems1, 2 and 3, the property loan approval workflow is free from redundant data anomalies and conflict data anomalies whereas it contains missing data anomalies.

5. CONCLUSION

Data flow analysis is an important step in workflow management. However, few commercial workflow management systems provide tools for discovering the data flow errors and conflicts in a workflow model. Our previous work has proposed the basic concepts and essential algorithms for data flow analysis in workflow management (Sun et al., 2005). In this paper, we extend our previous work by presenting a theoretical framework based on data and activity dependency analysis. Moreover, we theoretically proved the key concepts and applied them to the case of property loan approval workflow. The result of our work should help make workflow analysis and design more rigorous and more efficient by eliminating data flow anomalies systematically, thus leading to more efficient business process management.

We are currently continuing our work in a number of directions. First, we plan to develop a prototype of data flow manager in a workflow system so that our research results can be tested in real world applications. Second, we will develop a formal methodology for correcting data flow anomalies. Third, we intend to extend our work toward a new workflow design methodology based on data flow analysis (Sun and Zhao, 2004).

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, *Info. 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*, , 5, 3-4, 293-318.
5. 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
6. Kappel, G., Lang, P., Rausch-Schott, S., Retschitzegger, W. (1995) Workflow management based on objects, rules, and roles, *IEEE Data Engineering Bulletin*. 18,1, 11-18
7. Reuter, A. and Schwenkreis, F. (1995) Contracts: A low level mechanism for building general purpose workflow management systems, *IEEE Data Engineering Bulletin*, 18,1, 41-47
8. Sadiq, S., Orlowska, M., Sadiq, W., Foulger, C. (2004) Data Flow and Validation in Workflow Modeling. *Proceedings of The Fifteenth Australasian Database Conference*, Jan.18 -- 22, Dunedin, New Zealand, ACM Press, 207-214
9. Sun, S.X., J.L. Zhao, O.R. Sheng. (2004) Data flow modeling and verification in business process management. *Proceedings of AMCIS 2004*, Aug. 6-8, New York, NY, 4064-4073.
10. Sun, S. X. and J.L. Zhao (2004) A data flow approach to workflow design. *Proceedings of WITS 2004*, Dec 11-12, Washington D.C., 80-85
11. van der Aalst, W. (1998) The application of petri nets to workflow management, *The Journal of Circuits Systems and Computers*, 8, 1, 21-66.
12. van der Aalst, W. and van Hee, K. (2002) *Workflow management: Models, methods, and systems*, The MIT Press, Cambridge, Massachusetts, London, England.