A Tool for Minimizing Update Errors for Workflow Applications: the CARD Model

Dr. Jangha Cho* and Dr. Cheng Hsu**

Final Version March, 2005

Computers and Industrial Engineering

* Senior Manager, Technical Center, Samsung Electronics Corporation, Suwon City, Korea

** Professor, Decision Sciences and Engineering Systems Department, Rensselaer Polytechnic Institute, Troy, NY 12180

Abstract

Product Data Management, the ASP (application service provider) model of e-business, and other information enterprises involve complex database-driven processes that change frequently. Workflow management for these enterprises requires analysis of the causality of updates on the control logic of the processes **and** the semantics of the databases involved. Previous workflow analysis tends to focus on the former and leave the details of the database control to application developers. This practice could lead to inconsistent applications when the processes evolve and/or the underlying data and rules change. The field still needs an **integrated** causality analysis tool to facilitate the update problem. In this research, we develop a Control-Activity-Rule-Data (CARD) model of a **decision support tool** that helps workflow administrators analyze the effects of update in processes and/or databases on both control flow and data/rules flow. The CARD model contributes a new Workflow Information Resources Dictionary which represents data, rules, activities and control in a mutually independent but collectively integrated way to achieve the goal. An extended Workflow Causality Graph capable of representing workflow integrity rules enables the CARD model for implementation. An empirical validation of the model using three representative Product Data Management workflow cases at Samsung Electronics Corporation shows its correctness and relevance for practical applications.

Keywords: workflow management, causality analysis, Metadatabase

Acknowledgement: The authors wish to thank the anonymous referees for their deliberate and helpful comments. The comments have guided the authors to revise the manuscript and made it a better paper.

I. The Update Problem in Workflow Management: Need to Analyze Both Processes and Databases

Workflow management has its origin in the office automation (Bracchi and Pernici 1984). The attention in the early 1990s on business process re-engineering has further developed workflow management systems into an enterprise integration technology built on a database engine (see, e.g., Derungs, Volga and Österle 1997). Today, the trend has encompassed other information enterprises including ERP (Enterprise Resource Planning), PDM (Product Data Management), Supply Chain Management, Customer Relationship Management, and the emerging practice of ASP (application service provider) in e-business (an example of similar observation is given in Sheth, Aalst, and Arpinar 1999). Workflow management technology has become a major tool for the rapid development and evolution of complex business processes (Cho 2002).

In essence, workflow management systems first recognize the fundamental workflow tasks (i.e., ***activities***) that constitute a business process, and then define the sequencing of these activities (i.e., ***control***) for particular instances of the process. Information flow (i.e., ***data***) and workflow control knowledge (i.e., ***rule***) are encapsulated in the logic of both activities and control and stored in a database. Thus, Activity, Control, Data, and Rule are the four basic elements of workflow we recognize formally in this research (Workflow Management Coalition 1996, 1999). ***The problem of workflow updates is concerned with the changes initiated on any of these elements and their immediate cascading effects on other elements.*** This definition is narrower than some of the previous works by, especially, Ellis, Keddara and Rozenberg (1995); however, it represents a common and frequent problem in practice. The solution of this problem could facilitate the investigation of the larger problems. We refer to this definition everywhere in the paper when we mention dynamic changes, update effects, and the like.

Despite the advancements in the field (Casati, Ceri, Pernici, and Pozzi 1996, Nutt 1996, Kamath and Ramamritham 1996, Aalst, Basten, Verbeek, Verkoulen, and Voorhoeve 1998, Aalst, Hofstede, Kiepuszewski and Barros 2003, and Bi and Zhao 2003), system administrators still lack sufficient tools to

analyze update effects and deal with the problem. The reason is simple: while traditional design methods tend to ignore data semantics and leave database structuring to implementation, large-scale and fluid systems such as the above examples nevertheless feature frequent workflow updates that are subject to the encumbrance of the database. In these cases, they need to **analyze the effects of updates on both processes and databases in order to minimize errors**, not just to analyze processes.

The workflow update problem has two basic technical concerns: how to achieve or retain the ***soundness*** of the control of ***processes*** (i.e., initiation, termination, and consistent flow logic), and how to maintain the ***integrity*** of the ***database*** objects and inter-relationships (i.e., unique primary keys or object identifications, consistent foreign key values or class inheritances, and consistent data values). We refer to the achievement of both soundness and integrity the ***causality analysis*** of workflow updates. The reason that updates need integrated causality analysis while the analysis for new system design may not is two-fold. First, updates take place during operation and have to minimize disruption to the operation. Second, updates could take many forms and on any elements of workflow. In contrast, workflow designers could and would follow a hierarchical structure and analyze workflow elements only in sequence: they focus on processes (analyzing activities and control) and leave the detailed analysis of their database implementation (concerning data and rules) to workflow application developers to perform at a later stage. The latter, in turn, would typically hard-code the details into the applications (embedding data and rules in individual activities and control software) without also considering the global interactions of data flow and control flow at the database level. Update analysis at run time, on the other hand, does not have the same distinction of stages. Control flow and data flow would have to be analyzed together in the same short time frame. Changes could be initiated on data and data flow rules contained in the database, and then cascade to activities and controls in a bottom-up manner. In practice, such as PDM, the center of integrated analysis is often the databases, where a single update of

database elements could trigger multiple iterations of analysis with increasingly broader range of redesign on the process elements. Thus, the solution of the integrated causality analysis problem hinges on integrating the database elements into the traditional focus of process elements.

The literature has provided good foundations towards comprehensively considering the inter-dependencies among the four basic workflow elements. Some analyzed the soundness of control flow definition (i.e., activities and control flows) using high-level control graph (e.g., Joosten 1994, White 1998, Aalst and Hofstede 2000, Sadiq and Orlowska 2000, Voorhoeve 2000). Some studied workflow data and rules for data initialization and inconsistency problems at design time and for the consistency problems between workflow rules and event history (e.g., Alonso, Reinwald and Mohan 1997, Kumar and Zhao 1996 and Choi and Zhao 2002). Some considered control flow analysis and data flow (e.g., Reichert and Dadam 1998 and Adam, Atluri and Huang 1998). They all contribute to developing a new integrated framework. **Problems** still remain in, especially, the **integration** of database elements with process elements to support simultaneous analysis of update effects on both processes and databases. Workflow practitioners would manually "integrate" some of the previous tools when they need causality analysis on updates.

Here exists a gap between what the field needs and what the scientific results provide. We contend that a **tool** helping the practitioners to apply the otherwise disjoint scientific results in a **decision support** way is a necessary solution, and represents a significant contribution to the betterment of workflow management.

In this research, we develop such a new tool to facilitate workflow administrators performing integrated causality analysis to minimize workflow update errors on both processes and databases. The tool has to be grounded in a sound conceptual model; therefore, the bulk of the work is the development of the conceptual model to which we refer as the Control-Activity-Rule-Data (CARD) model of workflow, because it recognizes data and rules as basic workflow elements at the same level as the traditional activities and

control. The model, consistent with the Workflow Management Coalition's definitions, provides causality analysis algorithms based on the new formulation of these four workflow elements. The corner stone, and a major contribution, of the new model is an integrated representation method for database and process elements, called in this paper the Workflow Information Resources Dictionary (WIRD). The WIRD representation method extends the Metadatabase model (Hsu, Bouziane, Ratner, and Yee 1991) to support extended workflow causality graphs and analysis algorithms, for performing integrated analysis of control flow and data flow. The WIRD, therefore, is simultaneously a meta-model (workflow data and process definitions) and a repository of workflow metadata (data and process models). The CARD model, based on these results, promises to enable a tool capable of facilitating the update analysis problem discussed above. We substantiate this claim in the rest of the paper.

We develop the CARD model (the WIRD method and an extended work causality graphs), in Section 2, followed by a discussion of how the model may be applied to previous workflow analysis algorithms and integrate them for use in practice, in Section 3. Some of the algorithms add conspicuously to database integrity analysis for workflow updates. Section 4 discusses the prototyping of the CARD model and its empirical testing at Samsung Electronic Corporation, Korea, in the context of the manufacturer's Product Data Management enterprise. We conclude the discussion and suggest some future research in Section 5.

II. The Control-Activity-Rule-Data (CARD) Model

The CARD model consists of the new WIRD method and repository, and the extended workflow graphs as its means for user interface. To implement the model into a tool, it has to include analysis algorithms which, in general, could either be developed expressly for certain classes of applications intended, or be adopted from the literature. Either way, the analysis algorithms would use the meta-information WIRD provides to assure process soundness and database integrity. The analysis algorithms, working together

through the tool, check if a workflow schema or a workflow instance terminates correctly; if two activities and/or rules interfere with each other; and if an activity or a rule executes with all required data provided. The model also derives integrity rules directly from WIRD to control the propagation of workflow data along the association hierarchies, validate integrity of workflow schema, and examine the interference among multiple updates. These two types of analyses are simultaneously and recursively applied to workflow updates. The unique capability of the model is the repository of workflow metadata in WIRD.

II.1 Workflow Information Resources Dictionary (WIRD)

The Workflow Information Resources Dictionary provides, first, definitions of workflow elements expressly developed for causality analysis on updates. The definitions also include the semantic inter-relationships among these elements in the tradition of data models. In this sense, the WIRD is a particular meta-model, as shown in Figure 1. However, when the definitions are applied to workflow models in actuality, they also represent the collection of workflow metadata in a tightly structured database where the meta-model in Figure 1 becomes the schema of the metadata. In this sense, the WIRD is a repository, or a Metadatabase for workflow systems. Therefore, every construct (icon) in Figure 1, such as Activity and Sub-Process, represents a relation or an object class of metadata of certain type in the repository. The lines that connect these constructs – called meta-entity and meta-relationship – in Figure 1 represent structural inter-relationships among relations or object classes; or, in database terms, the integrity rules to maintain the consistency of data semantics and inheritances. The repository capability of the WIRD method distinguishes it in the field, and uniquely enables the CARD model to support causality analysis for workflow updates.

The first task of developing the WIRD method is to determine the sufficient information resources that update causality analysis requires. To achieve this, we follow the process definition standards of Workflow Management Coalition (1999) with some extensions proposed in the literature. In particular, we

incorporate the additional elements required for modeling workflow control knowledge and re-categorize the workflow elements according to their semantics in the causality analysis. For instance, we add software, hardware/network and human resources into the concept of workflow resources. We also term the concepts of workflow-relevant-data and workflow-control-data respectively the referred-data and the generated-data, to reflect the information processing perspective of integrity analysis. We differentiate the aggregate concept of workflow-transition-information into control-flow (for activity execution), data-flow, and workflow-control rules, corresponding to the transition conditions of WfMC. The control flow specifies the from-to execution relation between activities and control. The data flow shows the information flow between activities and control. The workflow control expresses the logic for the execution of an activity (e.g., pre-condition, post-condition, and temporal condition) and the routing of processes (e.g., parallel routing condition and exit condition for a iteration). The result is a basic recognition of four workflow elements: activity, control, data, and rules; of which rule is a major addition. We adopt the rulebase technology (Bouziane and Hsu 1997) to represent workflow control knowledge into rules and analyze them as such. Table 1 shows the fundamental syntax of the rule of the rulebase model.

We now discuss the semantic representation of the Workflow Information Resources Dictionary as shown in Figure 1. The representation uses the modeling primitives of the Two-Stage Entity-Relationship (TSER) method (a complete documentation is given in Hsu, Tao, Bouziane, and Babin 1993). They include the Entity (rectangular shaped, singular primary key) and the three classes of Relationship: Plural (diamond shaped, many-to-many association), Functional (broken diamond, many-to-one), and Mandatory (double diamond, fixed one-to-many). The WIRD unifies four separate but synergistically related meta-models: *control flow definition* specifying workflow enactment sequences, *workflow control knowledge definition* specifying process control and business rules, *data definition* specifying type and flow, and *workflow resources definition*

specifying participants, application, and software and hardware. Common and related contents of these models - i.e., data items, facts and rules - consolidate them into a whole. Another view of the WIRD is its support for a **rule model** (via rules, conditions, actions, events and facts), a **data model**, a **process definition model**, and a **process information model** for instances through data items and facts. Therefore, *the WIRD allows derivation of information from it to substantiate the requirements and implications of updates on process instances (run time) as well as process schema.* This is a contribution to the field.

II.2 Workflow Causality Graph (WCG)

The WIRD method leads immediately to a method for causality analysis visualization, the Workflow Causality Graph (WCG). The WCG shows the execution sequences between activities and rules in a manner similar to the triggering graphs of Baralis, Ceri, and Paraboschi (1995). However, the WCG also shows how a process *control flow* or a *data flow* affects the outcome of the execution of activities and workflow *control rules*, and how the control rules relate to activities and workflow data. The WCG consists of two levels of workflow models: process-oriented WCG and semantics-oriented WCG. The former defines the high-level execution precedence between activities and workflow control rules, and the latter the low-level specifications of *semantic* relations such as existence, complementary, and reference among activities, workflow data and workflow control rules. They correspond respectively to soundness and integrity analyses.

The process-oriented WCG could employ any appropriate graphical language such as Petri nets and state diagrams. However, for unified presentation with the semantic-oriented WCG, we opted to use a directed-graph consisting of four building blocks: Workflow Activity, Workflow Rule (set), Process Split-Join Operator, and Process Control Flow (see Figure 2). The workflow activity and the workflow rule set are comparable to the nodes (e.g., transitions in Petri-nets) of the previous control flow analysis approaches. A workflow rule set

consists of two subsets, activity rule subset and transition rule subset. An activity rule set is composed of pre-activity rules, post-activity rules, in-activity rules and data flow rules. The first three determine and control activity states, while the last controls information flow. A process control flow is a directed edge representing the precedence in executing workflow activities and workflow rules. An activity (or a rule in the rule set) is reachable from another activity (or another rule) if there is a sequence of activities and rules connected by directed edges that link these two activities (or rules). The process split-join operators (i.e., OR-Split, OR-Join, AND-Split, AND-Join, and Sequential), together with the process control flow, describe possible workflow routings. However, the actual process routing of a workflow is dynamically determined by transition rules, which refer to the control flow states and workflow data (referred and generated). More sophistication could be added later when additional causality analysis is required in practice.

The Semantics-Oriented WCG specifies the informational cross-reference among workflow elements. It uses five basic building blocks: Workflow Activity, Workflow Rule, Workflow Data, Data Flow, and Activity-Rule Relation. They are consistent with the Process-Oriented WCG, but provide finer granularity. Figure 3 shows the data flow view of a workflow causality graph. We define two types of workflow data: generated and referred. The generated data represent the information inserted, updated and deleted during the execution of an activity or a rule, while the referred data are the information consumed. A directed edge that connects a generated data and a referred data corresponds to information flow between activities and rules.

Figure 4 shows a unique value of the semantics-oriented WCG, the rule dependency view of workflow. It describes the inter-dependencies between a pair of rules or between a rule and an activity. We introduce three different types of rule dependencies for the analysis: (1) *existence* relation, (2) *complementary* relation and (3) *reference* relation. While the next section discusses the analysis in details, an overview of these relations is provided here. The existence relation represents cascade-delete relation. That is, when an activity or a rule

(on the from-side) has existence relation with other activities and/or rules (on the to-side), then deletion of this activity or rule will cause deletion of its corresponding activities and/or rules. A typical example of the existence relation is the inter-dependency between an activity and its pre-/post-activity rules. A complementary relation between a pair of rules, which do not precede each other, exists when the modification to the conditions of one rule affects the conditions of the other, in order to maintain the exclusiveness of the conditions. For example, the pair of transition rules for an exclusive OR-Split has complementary relation between their conditions. The reference relation exists when the data items used in one rule have referential relationships (via foreign keys or inheritances) with data items in other rules according to the semantic structure of their representation in the databases. This relation is necessary for maintaining the integrity of databases.

All WCGs are essentially just different uses of the WIRD method for different tasks of causality analysis. They are all inter-related, all draw from the same WIRD to produce instances of the graphs, and hence all are able to maintain their integration with others. We now turn to show how the WIRD and WCG support causality analysis.

III. Analysis: The Application of WIRD and WCG

The concept of the CARD model is reduced to practice if we can show that the WIRD method, through WCG, supports causality analysis in an integrated way required of minimizing update errors, as we discussed in the first section. Therefore, we select several representative analyses from the literature and represent them in algorithms corresponding to some WCGs. These algorithms feature the use of rules as well as data, and analyze database integrity as well as process soundness. We wish to stress that the analysis algorithms selected are not part of our claim of contribution, nor are they the testimony to the correctness of our model; rather, they just illustrate the use of, and the feasibility to use, the proposed model. We organize these analysis

algorithms according to the following overall logic of causality analysis, as they would in an implementation of the CARD model:

I: Verify workflow correctness and consistency

I.a Determine workflow confluence between a pair of distinct activity and rule:

- test inconsistency between a refer operation and a generate operation.
- test inconsistency between two concurrent generate operations.

I.b Determine workflow termination

- test live lock (infinite cycle), dead lock, and unintended execution

I.c Determine workflow process initialization requirements

II: Verify workflow integrity

II.a Derive integrity rules from workflow meta-model

II.b Determine cascading modifications.

II.c Check the consistency between workflow modification operations.

III.1 Workflow Soundness

The following examples are based on materials in the literature (e.g., Aiken, Widom and Hellerstein 1992; Vaduva, Gatziu and Dittrich 1997; and Sadiq and Orlowska 1997), as well as from practices in the field. The intent here is not to be comprehensive, but to be representative of how WIRD and WCG work, and how the CARD model would be applied to existing results in the field.

Example 1: How to Represent Workflow Confluence Algorithms

Workflow confluence analysis examines the concurrency of a pair of distinct workflow activities and/or workflow control rules to determine whether they are eligible for concurrent execution. It verifies whether concurrent activities and rules, starting at any states, produce the same final database and workflow states regardless of the execution sequences chosen between them. If so, then *workflow confluence* is guaranteed for them. We use the concept of workflow confluence to investigate the inter-dependencies of

workflow activities caused by data as well as the inter-dependencies of workflow control rules. Guaranteeing workflow confluence based on data is beyond concurrency control in standard database technology. For example, the common rejection-and-re-working strategy for resolving conflicts in concurrent database transactions may not be feasible to workflow management when a rejected activity lasts long time or re-working on the rejected activity is costly. In addition, the write-after-write conflicts of non-overlapping concurrent activities require imposing control flow connectors or inter-dependency constraints on the execution sequences of database operations to avoid. Standard database engines do not offer these application-based facilities. It has been workflow application developers' responsibility to verify that proper connectors and constraints are in place for concurrent accesses to the shared data.

Figures 5 and 6 illustrate the use of WCG to analyze workflow confluence concerning data. Figure 5 shows an inconsistent process execution caused by concurrent activities that are independent in control flow but are related in data flow. In Figure 5, both activities Act1 and Act2 have write accesses to the same data D_4, but do not succeed each other in the flow of process control. The workflow control rules, Rule3 and Rule4, determine a succeeding activity (either Act3 or Act4) to them, and both are predicated on D_4 (i.e., the conditions of Rule3 and Rule4 refer to the same data). The rules may not be triggered before Act1 terminates successfully; but may be triggered before Act2 is activated. This situation could lead to unintentional execution because D_4 could be updated by Act1 or Act2 and cause the conditions of Rule3 and Rule4 to be evaluated differently. As a result, the workflow could be routed inconsistently depending on the states of activities Act1 and Act2. Figure 6 illustrates the different database states that Rule3 may access.

Any of the following four conditions will assure workflow confluence for a pair of activities and/or rules. The first is that they have a fixed precedence-succession sequence and can never be eligible for concurrent execution at the same time. Second, they are on exclusive OR-Split branches. Third, they do not

have write accesses to the same data, and the data modification operations of one have no effect on the data referred by the other. Fourth, they have mutually exclusive conditions. The WIRD method provides sufficient information to check on these conditions since all data and rules are represented and connected to allow cross-referencing. The WCG visualizes the inter-relationships embodied in the instances of WIRD. For completion, we present a high level algorithm to formalize these ideas. The algorithm uses elements (e.g., data and rules) that are consistent with the WIRD method, and hence show how the WIRD method is sufficient and pertinent to the analysis. The algorithm maintains four data structures. RefData stores referring data of the selected activity or rule, and GenData the generated data. ConcActRule stores all activities and rules that are eligible for concurrent execution with the selected activity or rule, and EffActRule a list of activities and rules yet to be considered for modification propagation.

I. Initialize data structures

- Select an activity or a rule AR_i to analyze.
- Create data structures to store referred- and generated- data for the selected node.
  - if $AR_i \in A$, then copy D_{ref}^{ARi} and D_{gen}^{ARi} to RefData and GenData, respectively.
  - if $AR_i \in R$, then copy all data in condition clause and the referred data in its action clause to RefData, and copy generated data in action clause to GenData
- Create an ConcActRule for storing activities and rules eligible for concurrent execution.
- Create an EffActRule for storing activities and rules affected by the selected activity or rule.

II. Determine all concurrent activities and rules eligible for concurrent execution.

-While $N_i \neq$ "starting node", do reverse navigate the control graph.

-if $N_i \in L_{AND\text{-}Split}$, do

- For each $N_j \neq$ "end node" navigate the control graph.

-if $N_j \in L_{AND\text{-}Join}$, add to ConcActRule the nodes that neither precede nor succeed node N_i and that exist between N_i and N_j

III Determine all concurrent activities and rules that would interfere with N.

- For each node N_k in ConcActRule,
- if currtype of N_k is "activity",

- if all D_{ref}^{Nk} and D_{gen}^{Nk} do not match RefData or GenData, do remove.

- if currtype of N_k is "rule",

- if all data in condition and action of N_k do not match RefData or GenData, remove.

Where,

A: set of workflow activities R: set of workflow rules

D: set of workflow data L; set of edges (i.e., splits, joins, and synchronizers)

$D_i \in D_{ref} \cup D_{gen}$

$L_i \in L_{AND\text{-}Split} \cup L_{OR\text{-}Split} \cup L_{AND\text{-}Join} \cup L_{OR\text{-}Join} \cup L_{Synchronizer}$

$N_i \in A \cup R \cup L$

$AR_i \in A \cup R$

D_{ref}^{ARi}: referring data of the i_{th} activity or rule (AR_i)

$D_{gen}^{ARi:}$ generated data of the i_{th} activity or rule (AR_i)

Example 2: How to Represent Process Split-Join Combinations

At the core of workflow soundness is the assurance that workflow terminates properly. The analysis involves evaluation of various combinations of process splits and joins that could lead to problems such as infinite workflow cycle (live lock), workflow deadlock, and unintended multiple executions. The literature, especially Petri nets, has provided numerous results on the analysis of process split-join combinations (e.g., PT/TP handles and other tools); whose logic could be formulated in terms of the WCG language. Using WCG user interface helps enforce the correct use of the modeling constructs and thereby facilitates the valid representation of workflow control and business/operation rules.

For instance, the directed edges of a WCG symbolize that one node may activate the other. Therefore, a cycle in a WCG indicates the *potential* to have non-termination in the execution of activities and rules. In order to avoid infinite execution of a workflow cycle, the cycle must include some activities and/or rules that would provide proper exit conditions. These activities/rules would save the cycle if they either modify workflow data referred to by exit conditions, or eventually change the truth-value of exit conditions to false

after several iterations. To check if such activities/rules exist, the model allows examining each workflow data referred to by exit conditions and then traversing a WCG. If it cannot reach any node or if all nodes reached are outside of the workflow cycle, then the workflow data would not be modified during the iterations. Thus, the truth-value of exit conditions would never change and the workflow cycle would execute infinitely.

It is well-known that improper use of process split-join combinations and workflow control rules may result in workflow halting for no action. Consider these four possible combinations of process split and join: AND-Split/AND-Join, AND-Split/OR-Join, OR-Split/AND-Join, and OR-Split/OR-join. The AND-Split/AND-Join and OR-Split/OR-join are the typical split-join combinations supported by most workflow management systems and will not lead to any workflow termination problem. However, OR-split/AND-Join and AND-split/OR-Join combinations require comprehensive analysis in order to guarantee that a workflow terminates and executes as intended. When a process splits into multiple sub-processes through an OR-Split, and the sub-processes subsequently join via an AND-Join or synchronize using a Synchronizer, then any activities and rules following the join or the synchronizer could never get activated - i.e., resulting in deadlock, because the activities following the split would not terminate together. Another deadlock problem is due to an OR-Split/AND-Join combination with incorrect definition of workflow control rules; i.e., branching conditions. This deadlock problem would occur when a process split by multiple OR-Split's and join later by an AND-Join or by a Synchronizer. It would also occur when the branching rules of the OR-Split's do not activate all activities that are joined by an AND-Join.

These deadlock conditions are detectable by traversing a WCG since it sufficiently represents these problematic combinations. The actual algorithms will follow the many results established in the literature and belong mainly to the implementation effort. In a similar way, the incorrect use of a split-join combination may also lead to unintentional and unexpected workflow executions. When a process splits into multiple sub-

processes through an AND-Split, and the sub-processes later join by an OR-Join, then, the activities and rules following the OR-Join would activate each time the activities following the split terminates – an unintended result. The WCG supports its analysis in the same manner.

Example 3: How to Represent Activity and Rule Initialization

The workflow initialization criteria guarantee that each workflow activity or workflow control rule executes with all required workflow data provided before its execution. That is, all input data an activity uses must be available before it should start execution, and all output data it produces must become available to the activities and rules that use them. Similarly, every data referred in conditions and actions of a rule must be ready to be able to fire the rule. For this analysis, we build *data flow edges* in a Workflow Causality Graph, in additional to the previous control flow edges. The following is a basic logic for using the WCG to check the required conditions: For each activity or rule, the initialization analysis begins with traversing a WCG in the orientation of the control flow edge of the activity (or the rule) until it reaches the start node. For each node (i.e., activity or rule) visited during the traversal, the generated data of the node are compared with the referred data of the activity (or a rule) being analyzed. If the traversal does not reach any node that generates the referred data, then the activity or rule concerned would execute with missing data and facts. If the first node reached in the traversal is on a parallel branch of a process to the original node (i.e., they are eligible for concurrent execution), then further analysis of workflow confluence is required.

I. For each data $D_i \in D_{ref}$, traverse the WCG in the data flow edge's orientation
- if current visited workflow generated data $D_j = D_i$
- if $D_j \in Prec(A(D_i))$*; i.e., if D_j always precedes D_i*
then select another D_i and continue
- else if $D_j \notin Prec(A(D_i))$*; i.e., if D_j and D_i are on parallel branches*

then perform workflow confluence analysis

- if current visited data $D_j \neq D_i$, continue.

III.2. Workflow Integrity Rules Determination and Derivation

In additional to investigating soundness, causality analysis also needs to determine the proper information flows required for propagating the updates, considering all workflow elements affected; which is crucial in order to maintain workflow integrity. The WIRD model includes all workflow elements and their semantic (informational) inter-dependencies, as discussed in Section II.2. Therefore, we derive based on these interdependencies five groups of workflow integrity rules: workflow activity, control flow, workflow control knowledge, workflow resources, and workflow data and data flow. For all these groups, the modification of a basic workflow element could be deletion, insertion, or update of the element; each of which could involve key-attributes or non-key-attributes. As an example, Figure 7 shows the inter-relationships between a workflow activity and five other elements: workflow participant specifications, workflow data, workflow application declaration, workflow control rule and control flow. There are two types of integrity rules necessary for propagating the change of an activity to the five elements. The first is functional relationships (FR) from the entity Activity to the entity Workflow Participant and Software Resources; which implies referential integrity. The second includes four plural relationships (PR) with the entity Workflow Data, Rule, Control Flow, and Workflow Participant; which implies strong referential integrity.

Key attribute deletion

The deletion of a key-attribute from the entity relation Activity corresponds to the deletion of an activity. When one deletes an activity, one should remove its associations with other elements, too. Thus, when an instance of Activity is deleted, the corresponding instances of all many-to-many (PR) relations (i.e., Role, Generate_Refer, Encapsulate, and From To) containing the key of the activity should be deleted as well.

IF changes_in_oepr(,Act_ID←Activity,FALSE,TRUE,FALSE) † AND
act_ID.old←Activity‡ = act_ID←Generate_Refer AND
act_ID.old←Activity = act_ID←Encapsulate AND
act_ID.old←Activity = act_ID←Role AND
act_ID.old←Activity = act_ID←From To
THEN act_ID←Generate_Refer := NIL*;
act_ID←Encapsulate := NIL;
act_ID←Role := NIL;
act_ID←From To := NIL;

(† The parameters of changes_in_oepr() correspond to process name, attribute of a workflow element monitored, update flag, delete flag, insert flag.
The left operand of ← is attribute name, and the right operand table name.
‡ *attribute_name*.old: the old value of attribute_name deleted or updated.
attribute_name.new: the new value
* *attribute_name := NIL*: the deletion of the instances with the key matching *attribute_name*)

Key attribute insertion

The insertion of a key-attribute to the relation Activity corresponds to the definition of a new activity. Since the definition includes the software resources it uses, the software resources must be definable, too; i.e., already existing in the entity relation Software Resources. If not, the new software resource has to be added to the relation before the new activity can refer to it. That is, when a new instance is defined in the entity Activity, its non-key-attribute representing software resources must either be null or matching exactly the key-attributes of some of the instances of the entity Software Resources.

IF changes_in_oepr(,act_ID←Activity,FALSE,FALSE,TRUE) AND
EXIST(appl_ID←Software Resources = appl_ID.new←Activity) †
THEN Act_ID←Activity := Act_ID.new←Activity;
appl_ID←Activity := appl_ID.new←Activity;
(† EXIST(attribute1 = attribute2): returns TRUE if the value of attribute1 matches the value of attribute2)

Key attribute Update

There are two possible ways to handle the update of a key-attribute: direct replacement of a key with new key, and deletion of a key followed by an insertion of a new key. Generally, the update of a key-attribute corresponds to the latter, because it is not usual to change the value of a key (activity identification) without changing other the values of non-key attributes. Hence forth, we assume that the integrity rules for key-attribute update are the sum of the integrity rules for the key-attribute deletion and insertion.

Non-key attribute Delete

The deletion of any values for non-key attributes has no impact on other relations as long as the information model consists of normalized entities and relationships only. The WIRD structure is such a model, thus we do not need to derive special integrity rules for this case.

Non-key attribute Insertion or Update

Insertion of new values for non-key attributes of new activities is a non-issue, except the situation discussed above under key-attribute insertion. The same is true for update, too. When the non-key attributes have any FR relationship with other entities or relationships, such as in this example with Software Resources, then one must make sure that matching instances exist in the relations referenced.

```
IF changes_in_oepr(,appl_ID←Activity,True,FALSE,FALSE) AND
    EXIST(appl_ID←Software Resources = appl_ID.new←Activity)
THEN appl_ID←Activity := appl_ID.new←Activity
```

We do the same for all other groups and complete the set of workflow integrity rules.

IV. The Implementation and Empirical Testing

We tested the proposed CARD model in an industrial setting. The model was applied to develop a causality analysis tool for workflow developers and administrators when they needed to update the instances

of the workflow management system. The analysis started with a request of changing workflow specification or workflow instances. The request was entered as update to the workflow instances represented in the WIRD to determine the workflow data and the process definitions affected. Then, the proposed workflow elements were represented as workflow causality graphs for analysis of the soundness and integrity, as discussed in Section III. The tool fed the problems detected and any conflicts from cascading the change on information resources back to the requesting party along with recommendations for possible adjustment on the request. The process proceed in an iterative way to provide decision support for the user.

IV.1 System Architecture for the Decision Support Tool Using the CARD Model

The implementation of the CARD model entails an environment supporting the WIRD repository, a WCG generator serving as the user interface, and an analyzer operating the algorithms on workflow updates using WIRD and WCG. The environment needs to interact with the workflow management system and the workflow databases. We have developed a prototype system, coded in C++, to test the CARD model in a Product Data Management context. We illustrate it below with an example of constructing a WCG for confluence analysis. The complete testing and the design of the prototype are available from Cho (2002).

Figure 8 shows the graphic user interface of the prototype. Users could specify the updates intended by directly editing the workflow activities, rules and control connectors. An activity dialog assists the user to modify (revising and inserting) activity descriptions, workflow data (i.e., referred and generated types) and workflow control rules (i.e., pre-activity conditions, in-activity rules, and post-activity conditions). Similarly, a control dialogue allows users to edit transition type (i.e., sequential, AND-Split, AND-Join, OR-Split, and OR-Join) and workflow control rules (i.e., transition conditions). The system can then construct the process-oriented WCG and semantics-oriented WCG as discussed in Section II.3, This process involves decomposition of activities into pre-activity rules, activities, and post-activity rules, and transformation of

controls into rules according to the WCG conventions. Any number of activities and rules can be selected for confluence analysis with the rest of the workflow. The system conducts confluence analysis for them one at a time. Each time, it searches for activities and rules (i.e., target nodes) from all others that are eligible for concurrent execution together with the selected activity or rule (i.e., the source node). Then, it conducts confluence analysis as described in Section III.1. Figure 8 also shows that three target nodes are found concurrent to the source node A1. If any *potential* confluence problem is found, the system would highlight the on screen and display the problem description (i.e., source and target node, error type, and data item causing the problem) in the analysis result dialogue box. Accordingly, the modeler could add additional control connector between the target and source nodes, disallow the target and source nodes to execute concurrently (e.g., replace AND-Split and AND-Join with OR-Split and OR-Join), or modify the data items.

The complete example involves other pages for other analyses in a similar way. The termination analysis appears in the termination analysis page of a workflow analysis dialogue box. The system verifies the existence of infinite cycle, deadlock, and unintended execution and displays results in the termination analysis page and the main system page. It shows the split-/join-type, data items, and workflow cycles that cause termination problems. Again, the modeler could correct termination errors according to the causes diagnosed, through editing the activities and controls of the WCG corresponding to the highlighted activities or rules. The system verifies the availability of data required for execution of an activity or rule concerned. It checks whether the workflow will lead to the generation of each referred data of the activity or rule for its proper execution. The activity or rule chosen for analysis is highlighted on the main window, and the data items that violate the criteria are displayed in a dialog box of the initialization analysis page. To correct the problem, a workflow modeler could, for example, delete the data items from the highlighted node or add the data items as generated data at the preceding activities and rules. Based on the integrity constraints derived in Section

III.2, the system conducts integrity checking on workflow activities, workflow control rules, workflow data, control flow, and/or workflow resources. The workflow modeler uses the check box to select the basic workflow elements interesteing for analysis. The system will display the problems found, if any, in the integrity analysis page and the main page. The integrity analysis page shows the source of errors (i.e., which basic workflow element violated integrity) and error types (i.e., insert, update, and delete), to assist the modeler modify the workflow elements and correct the problems.

IV.2 Empirical Test of the Tool

We applied the above prototype to three representative applications of the Product Data Management workflow system at Samsung Electronic Corporation, Korea. The purpose of the test is to establish the need for integrated causality analysis in practice, and the value and validity of the CARD model as we claimed. The system runs on a workflow development server (Hewlett-Packard K class with HPUX v10.20) with a product database of 2,860,000 records for over 300,000 product parts. The server is related to the divisional servers (Hewlett-Packard N and T classes with HPUX 10.20 and v11.0) containing 3,000,000 ~ 5,000,000 records for 15,000 ~ 80,000 product parts and 300 ~ 6,000 files.

The workflow models of the selected workflow applications had different degrees of complexity. The model of the corporate component information (CIS) workflow is the simplest, while the project management (PRJ) system the most complex (see Table 2). They cover the range of complexity existing at the workflow system studied. For testing purposes, nine engineers of varying degrees of technical skills and experience with workflow systems from three divisions of the corporation participated in the test. Each group mixed three engineers from different divisions who had similar level of skills and experience (see Table 3).

The test provided each engineer an update request for each of the above applications, and asked them to update manually the workflow models of these applications as they would do with the previous

technology. They next used the prototype system to analyze the updates and figure out the correct answers according to the results of the analysis (which, in a way, shows the recommendations of the system). The tester then evaluated the improved solutions with the engineers in a comprehensive manual examination of the cases, to very the results recommended by the system. All three groups of engineers made significant errors in their manual updates, and found the analyses provided by the system correct, practical and useful. Although the least skillful group really needed the system to guide their updating, as expected, the most skillful group still relied on the system to correct some critical mistakes they made initially. The level of complexity of workflow applications did not matter too much, either. Errors in manual updates were cross the board, found almost equally in the simple application and the more complicated ones. All engineers agreed that the system caught virtually all the errors built in their manual updates in an efficient and effective manner. There were some residual errors, however, that the system did not catch. They found these remaining errors were due essentially to misunderstanding of the workflow models, or the lack of information about the complete intent of the original modelers of the workflow applications. Without sufficient metadata in the workflow system, the analysis can only achieve precision, not necessarily accuracy.

The testing result seems to confirm our expectations of the CARD model, including both its correctness and feasibility. The test is also consistent with our assertion about the update problem in Section I.

V. Conclusion

Contemporary workflow management systems are built on Control Technology and Information Technology; however, previous causality analysis methods do not sufficiently consider the information side of workflow updates. We combine results from both fields to bring about a new model integrating all four basic workflow elements - Activity, Control, Data, and Rule - to minimize update errors on both processes and

databases. This model enables an integrated causality analysis tool for workflow administrators, who until now had to rely on manual effort to perform this task when updating workflow logic. The main contribution of the paper is the new integrated method for representation of workflow elements that connects processes with databases and thereby facilitates integrated causality analysis. This integrated method makes it more efficient and effective to analyze update effects due to changes in database objects as well as in activities and control. The promises of the CARD model are observable in an empirical testing on Product Data Management workflow systems at Samsung, one of the largest conglomerates in the industrial world.

The WIRD method results in a particular meta-model of workflow elements optimized for causality analysis **and** a repository of workflow metadata which reveals the impact of updates on all workflow elements. The repository provides meta-information for simultaneous analysis of the soundness of processes (control and activities) and the integrity of the database (data and rules pertaining to control and activities). Its integration of data and rules with activities and control recognizes the interrelationship between processes and databases. It uniquely supports update causality analysis since updates on any part of workflow activities, control, data, and rules at run time require the analyses on other parts be performed simultaneously in order to review the cascading effect to the entire workflow system. Therefore, this model adds the ability to reduce workflow update errors due to changes in data and data flow rules, as well as changes in other workflow elements, to the literature. It also has the potential to serve as a reference model for workflow elements in a way similar to a precedence discussed in Hsu, Cho, Yee, and Rattner (1995).

The new model also includes a Workflow Causality Graph and database integrity rules to analyze workflow soundness and integrity; both results are rich in data semantics. The WCG, therefore, is an extension to previous results in its incorporation of data flow for the update analysis. The prototype system adopted algorithms from the literature which were adequate for the testing, but do not represent the state of the art in

the field. This can be improved in production systems as long as we have shown how WIRD could be applied to many analysis algorithms. The algorithms so adopted address data and rules directly. For example, the confluence analysis algorithm allows analyzing concurrency problems between data accessible by an activity and those by a rule. The integrity analysis algorithm verifies the consistency between workflow data and rules, as well as the soundness of control flow definitions. Together, the integrated causality analysis developed is immediately beneficial to updating workflow processes; however, it would be beneficial to the design of a new workflow system that changes frequently, as well. Although the initial investment for the implementation of the CARD model could be non-trivial, the formal representation of rules and data in the WIRD promises to make the investment paid-off in the long term when updates are frequent.

The present result could benefit from some extensions in the future. First, it could enhance its analysis algorithms by adapting the best results from the literature in a way that takes advantage of the WIRD. Second, it could use a methodology such as how to create and maintain the WIRD for particular application domains. The prototype used existing workflow databases to derive the necessary meta-information about data and rules to construct the WIRD. This reverse engineering would be unnecessary if the developers had followed a methodology to create a WIRD in the first place. Third, the results could expand into analysis for distributed workflow updates. The following issues need to be addressed: correctness criteria for distributed workflow updates and concurrency control on distributed workflow specification updates. Finally, we expect the proposed approach to be beneficial to managing workflow for service operations, where business processes tend to be less rigid than the production systems intended for the research. In particular, we believe the emerging practice of ASP (Application Service Provider) in e-business presents a vital need for something similar to the CARD model, with sufficient extensions.

References

Aalst, W. M. P. van der, Hofstede, A. H. M. ter, Kiepuszewski, B. and Barros, A.P. (2003). Workflow Patterns. *Distributed and Parallel Databases*, 14(1), 5-52

Aalst, W. M. P. van der and Hofstede, A. H. M. ter (2000). Verification of Workflow Task Structures: a Petri-Net-based Approach, *Information Systems*, 25(1), 43-69.

Aalst, W.M.P. van der, Basten, T., Verbeek, H.M.W., Verkoulen, P.A.C. and Voorhoeve, M. (1998). Adaptive Workflow: On the interplay between flexibility and support*, Proc. of the first International Conference on Enterprise Information Systems*, Setubal, Portugal, 353-360.

Adam, N.R., Atluri, V., and Huang, W.-K. (1998). Modeling and Analysis of Workflows Using Petri Nets, *Journal of Intelligent Information Systems*, 10(2), 131-158.

Aiken, A., Widom, J.and Hellerstein, J.M. (1992). Behavior of Database Production Rules: Termination, Confluence and Observable Determination, *Proc. of the ACM SIGMOD Intl. Conf. On Management of Data*, San Diego, CA

Alonso, G., Reinwald, B. and Mohan, C. (1997). Distributed Data Management in Workflow Environments, In: Seventh International Workshop on Research Issues in Data Engineering, Birmingham, England.

Baralis, E., Ceri, S. and Paraboschi, S. (1995). Improved Rule Analysis by Means of Triggering and Activation Graphs, *Proc. of 2nd Intl. Workshop on Rule in Database Systems*, Athens, Greece

Bi, H. H. and Zhao, J. L. (2003). Mending the Lag between Commerce and Research: A Logic-based Workflow Verification Approach, *Proc. 8th INFORMS Computing Society Conference*, Chandler, Arizona.

Bouziane, M and Hsu, C. (1997). A Rulebase Management System Using Conceptual Rule Modeling, *International Journal on Artificial Intelligence Tools*, 6(1), 37-61.

Bracchi, G. and Pernici, B. (1984). The Design Requirements Office Systems, *ACM Transactions on Office Information Systems*, 2(2), 151-170.

Casati, F., Ceri, S., Pernici, B. and Pozzi, G. (1996). Workflow Evolution, *Proc. of 15th International Conference on Conceptual Modeling*, Cottbus, Germany, 438-455.

Cho, J-H, *Two-Stage Causality Analysis for Managing Updates,* unpublished Ph.D. dissertation, Engineering Science, Rensselaer Polytechnic Institute, Troy, NY 12180-3590.

Choi, Y. and Zhao, J. L. (2002). Matrix-based abstraction and Verification of E-Business Processes, *Proc. WEB 2002: The First Workshop on e-Business*, Barcelona, Spain.

Derungs, M., Volga, P. and Österle, H. (1997). From BPR Models into Workflow Applications, In: P. Lawrence P, *Workflow Handbook 1997* (PP. 49-60), John Wiley & Sons Ltd.

Ellis, C.A., Keddara, K. and Rozenberg, G. (1995). Dynamic Changes Within Workflow Systems, *Proc. of the*

International Conference on Organizational Computing Systems COOCS'95, Milpitas, CA, 10-21.

Hsu, C., Bouziane, M., Rattner, L. and Yee, L. (1991). Information Resources Management in Heterogeneous, Distributed Environments: A Metadatabase Approach, *IEEE Transactions on Software Engineering*, 17(6), 604-625.

Hsu, C., Tao, Y., Bouziane, M. and Babin, G. (1993). Paradigm Translations in Integrating Manufacturing Information Using a Meta-Model: The TSER Approach, *Journal of Information Systems Engineering*, 1(3), 325-352.

Hsu, C., Cho, J., Yee, L. and Rattner, L. (1995). Core Information Model: A Practical Solution to Costly Integration Problems, *Computer and Industrial Engineering*, 28(3), 523-544.

Joosten, S. (1994) Trigger Modeling for Workflow Analysis, *Proc. CON '94: Workflow Management*, R. Oldenbourg Publishers, Vienna, Munchen, , 236-247.

Kamath, M. and Ramamritham, K. (1996). Correctness Issues in Workflow Management. In: Distributed System Engineering 3, 213-221.

Kumar, A. and Zhao, J. L. (1996) "A Framework for Dynamic Routing and Operational Integrity Control in a Workflow Management System," *Proc. of the 29th Hawaii International Conference on systems Science*, pp 492-501.

Nutt, G.J. (1996) "The evolution toward flexible workflow systems," In: *Distributed Systems Engineering*, 3, 276—294.

Reichert, M. and Dadam, P. (1998). ADEPTflex-Supporting Dynamic Changes of Workflows Without Loosing Control, *Journal of Intelligent Information Systems*, Special Issue on Workflow Management Systems, 10(2), 93-129.

Sadiq, W. and Orlowska, M. E. (1997). On Correctness Issues in Conceptual Modeling of Workflows, *Proc. of the 5th European Conference on Information Systems*, Cork, Ireland.

Sadiq, W. and Orlowska, M. E. (2000). Analyzing Process Models Using Graph Reduction Techniques, *Information Systems*, 25(2), pp.117-134.

Sheth, A., Aalst, W.M.P. van der, and Arpinar, I. "Process Driving the Networked Economy," *IEEE Concurrency*, 7(3) 18-31.

Vaduva, A., Gatziu, S. and Dittrich, K.R. (1997) Investigating Termination in Active Database Systems with Expressive Rule Languages, Proc. 3rd. International Workshop on Rules, In: Database Systems, 149--164, Springer-Verlag, Berlin.

Voorhoeve, M. (2000). Compositional Modeling and Verification of Workflow Processes, In *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of Lecture Notes in Computer Science, 184--200. Springer-Verlag, Berlin.

White, G.M. (1998). Towards the Analysis and Synthesis of Adaptive Workflow Systems, *Proc of the Conference on Computer-Supported Cooperative Work*, Seattle, WA.

Workflow Management Coalition (1996) *Workflow Management Coalition: Workflow Standard - Interoperability Abstract Specification*, WFMC-TC-1012.

Workflow Management Coalition (1999), *Workflow Management Terminology and Glossary*, WFMC-TC-1011, Issue 30, http://www.aiim.org/wfmc/standards/docs/glossy3.pdf.

FIGURES

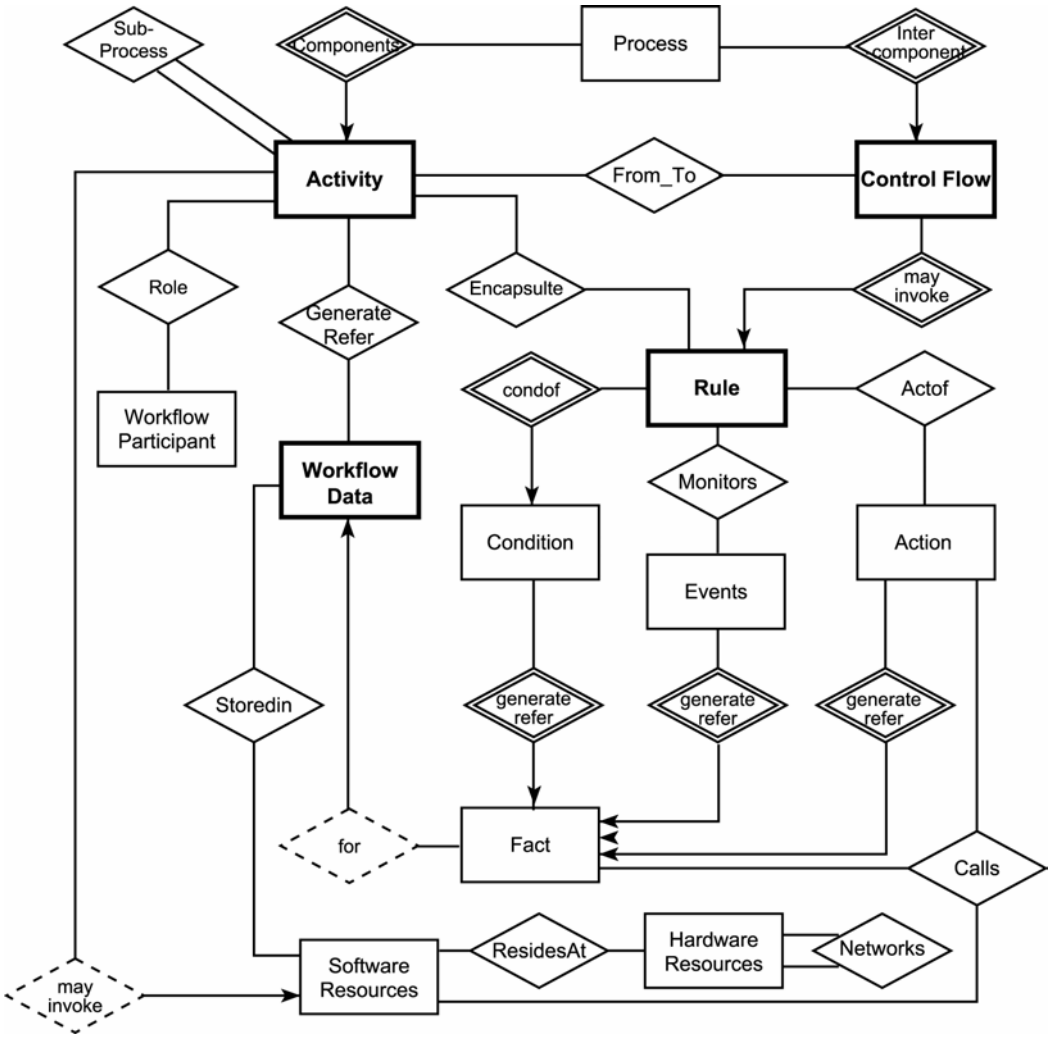

Figure 1: Workflow Information Resources Dictionary

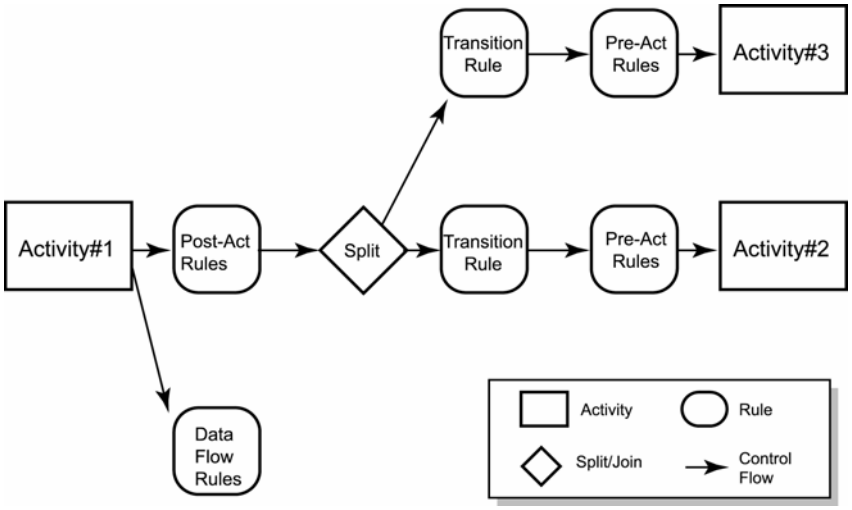

Figure 2: Process-Oriented Workflow Causality Graph

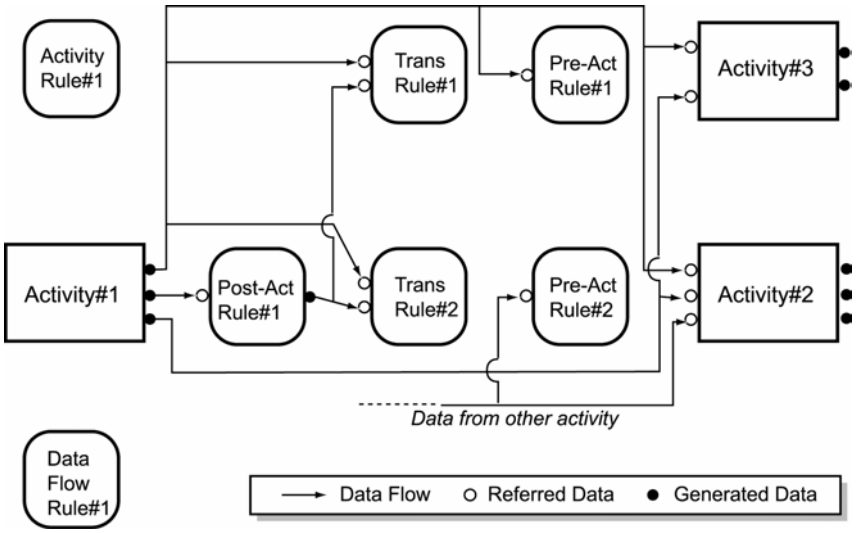

Figure 3: Semantics-Oriented Workflow Causality Graph: Data Flow

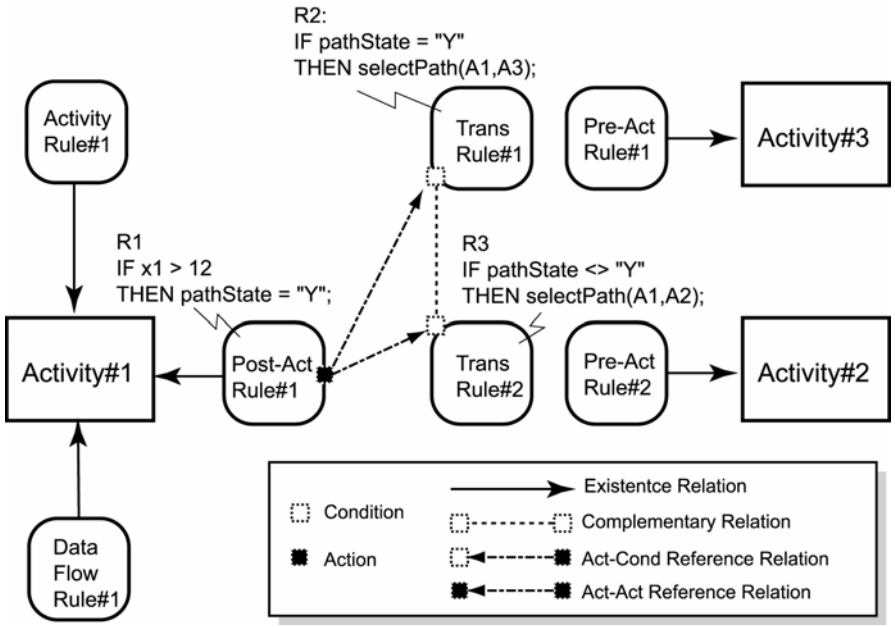

Figure 4: Semantic-Oriented Workflow Causality Graph: Rule Dependency

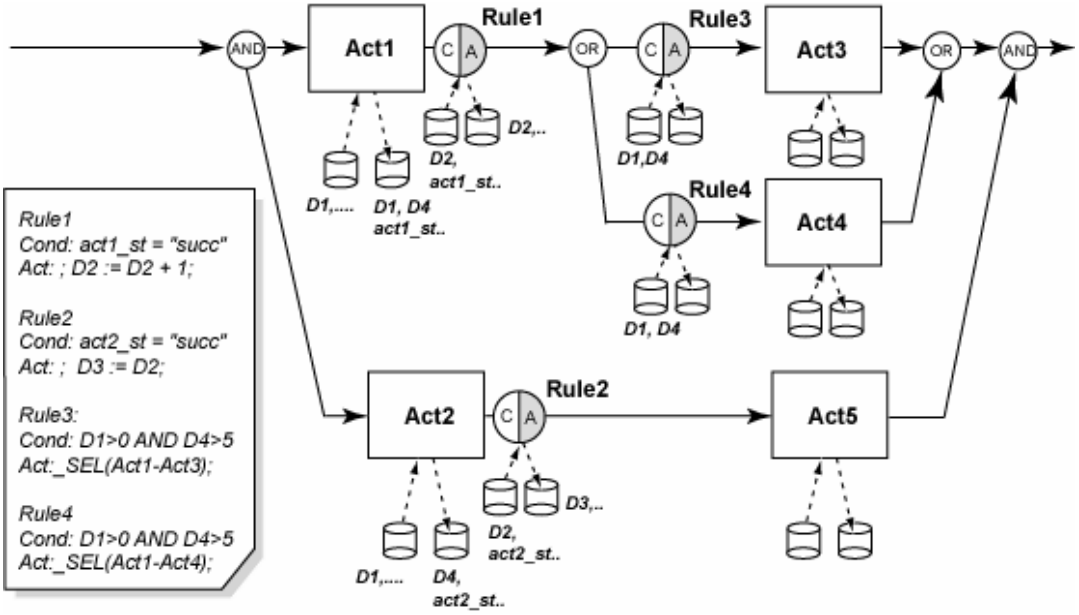

Figure 5: An Example of Concurrent Activities and Rules

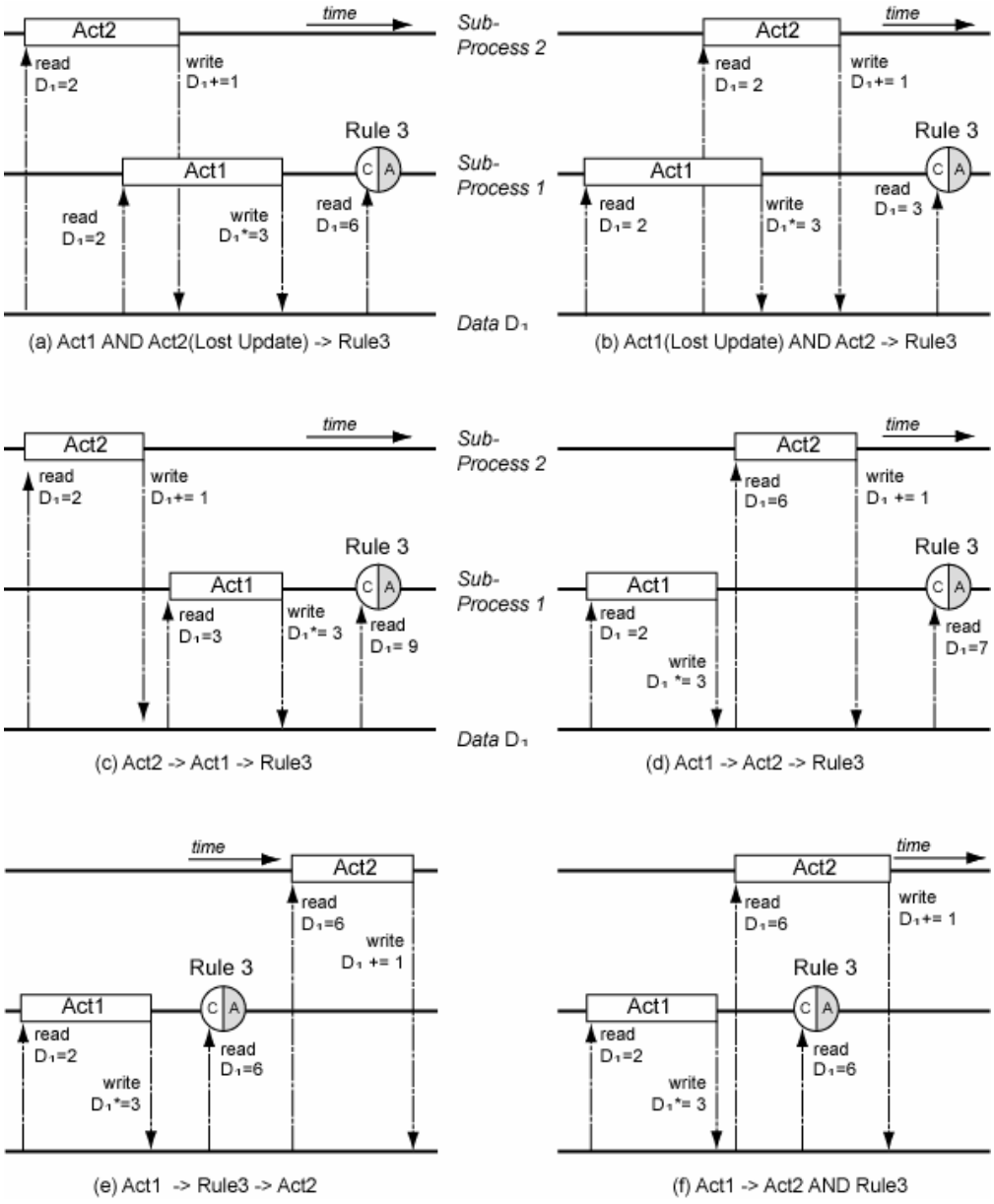

Figure 6: Time Diagrams of Generate- Generate Type Problems

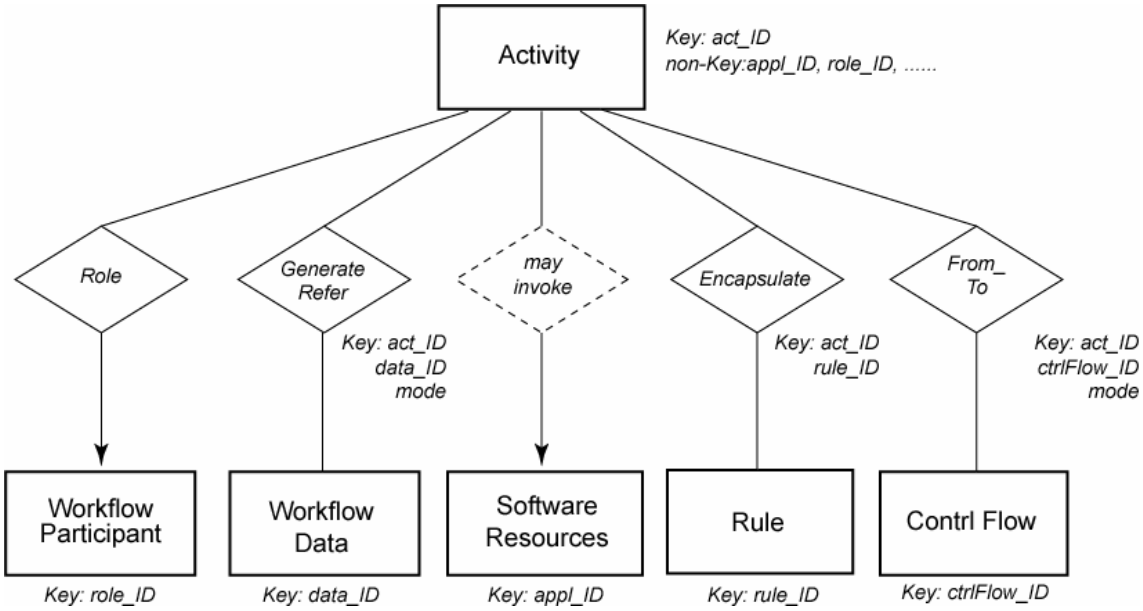

Figure 7: Workflow Causality: Workflow Activity Modification

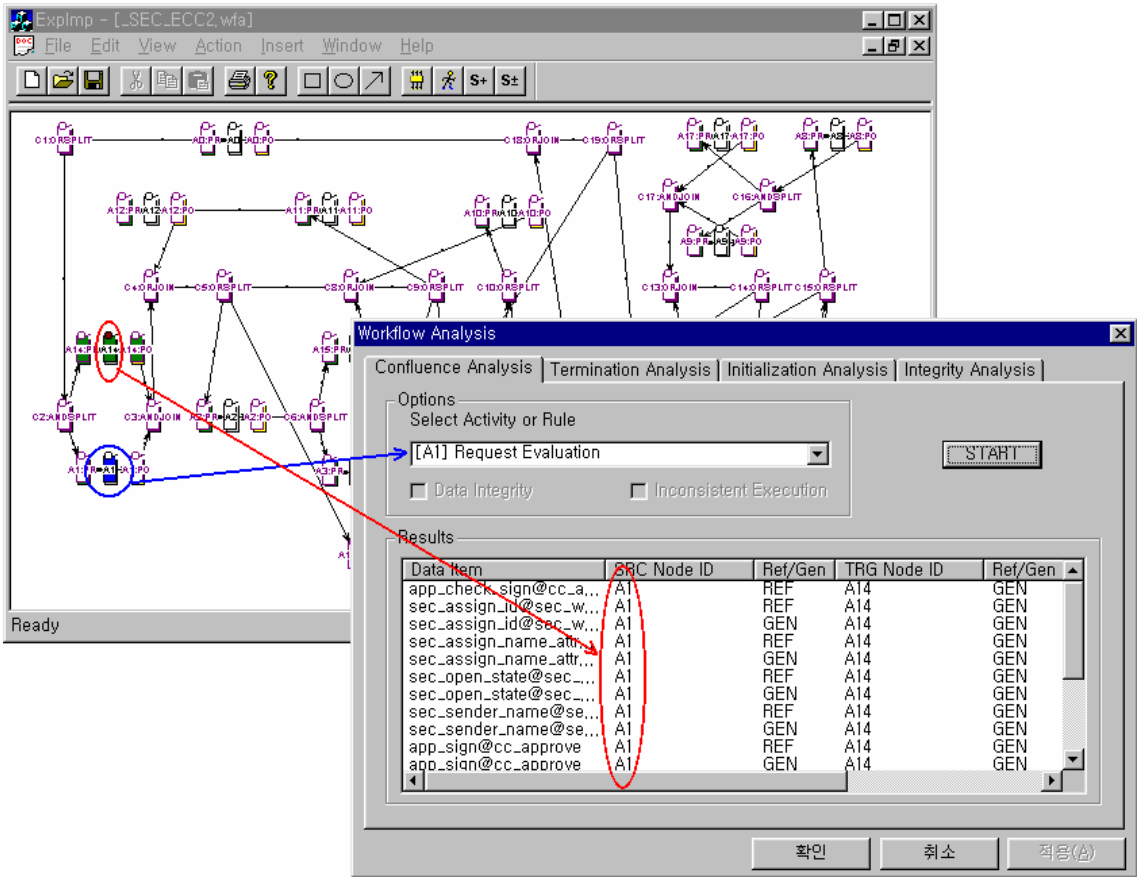

Figure 8: The User Interface (Workflow Confluence Analysis Results)

TABLES

<Rule> ::= [<Event>] IF <Condition> THEN [<Action>]+
<Event> ::= Item <Operation> | <Rule-Ident>
<Condition> ::= <Expression>
<Action> ::= <Assignment-Statement> | <Procedure-Call>
<Assignment-Statement> ::= <Action-Ident> := <Evaluated-Fact>
<Procedure-Call> ::= Procedure-name([<Parameter-List>]*)
<Expression> ::= <Fact> <Operator> <Fact> | <Fact>
<Fact> ::= <Evaluated-Fact> | <Declarative-Statement>
<Evaluated-Fact> ::= <Simple-Fact> | <Composed-Fact>
<Simple-Fact> ::= Constant | Item | <User-Ident> | <Action-Ident>
<User-Ident> ::= Identifier
<Action-Ident> ::= Identifier
<Composed-Fact> ::= <Function-call> | <Expression>
<Function-call> ::= Function-name([<Parameter-List>]*)
<Parameter-List> ::= <Parameter>[, <Parameter-List>]*
<Parameter> ::= <Fact>, parameter_type

Table 1: Syntax of the Rule Language

Workflows	Activities	Data Items	Control Rules	OR-Split	OR-Join	AND-Split	AND-Join
CIS	9	43	19	1	1	3	3
PRJ	90	431	77	2	2	13	14
ECM	20	287	42	7	4	4	4

Table 2: Workflow Applications at Samsung

Group	Description	Primary workflow skills
1	General workflow administrators at different divisions	Database administration
2	Master workflow administrators and software application developers	Database administration and workflow application development
3	Master workflow application developers for product data management	Workflow application development, modeling, and workflow engine development

Table 3: Groups of Engineers with Different WF Modeling Skills